

**ΟΙΚΟΝΟΜΙΚΟ  
ΠΑΝΕΠΙΣΤΗΜΙΟ  
ΑΘΗΝΩΝ**



**ATHENS UNIVERSITY  
OF ECONOMICS  
AND BUSINESS**

**ΣΧΟΛΗ  
ΕΠΙΣΤΗΜΩΝ &  
ΤΕΧΝΟΛΟΓΙΑΣ  
ΤΗΣ  
ΠΛΗΡΟΦΟΡΙΑΣ**  
SCHOOL OF  
INFORMATION  
SCIENCES &  
TECHNOLOGY

**ΜΕΤΑΠΤΥΧΙΑΚΟ  
ΕΦΑΡΜΟΣΜΕΝΗ ΣΤΑΤΙΣΤΙΚΗ**

**MSc IN  
APPLIED STATISTICS**

## **Microarray Data Analysis**

**By**

**Alina Kasian Panagiotopoulou**

**A THESIS**

Submitted to the Department of Statistics  
of the Athens University of Economics and Business  
in partial fulfilment of the requirements for  
the degree of Master of Science in Applied Statistics

Athens, Greece  
February 2025





## **DEDICATION**

This thesis is dedicated to my family, whose unwavering support and encouragement have been invaluable throughout my academic journey. Their belief in me has been my greatest motivation. I also dedicate this work to my professors and mentors, whose guidance has shaped my understanding of statistical data analysis and its applications. Their insights and expertise have been instrumental in developing my analytical skills and deepening my appreciation for rigorous statistical methodologies.



## ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Mr. Papastamoulis, for his invaluable guidance and continuous support throughout the development of this thesis. His expertise in microarray data analysis and advanced statistical methodologies has been instrumental in shaping my understanding of the subject. His insightful feedback and rigorous approach have significantly contributed to the quality and depth of this work.

I deeply appreciate his patience, encouragement, and willingness to provide thoughtful discussions that helped refine my analysis and methodology. His dedication to excellence has been a great source of inspiration.

This thesis would not have been possible without his expertise, and I am truly grateful for his mentorship.





## VITA

I pursued my undergraduate studies in Mathematics at the National and Kapodistrian University of Athens. During my studies, I specialized in Applied Mathematics, Statistics, and Business Management, focusing on courses such as probability theory, statistical modeling, computational statistics, econometrics, and actuarial mathematics. My academic projects involved statistical computing using R.

Following my undergraduate degree, I continued my education with a Master's in Applied Statistics at the Athens University of Economics and Business. My postgraduate studies have provided me with a strong foundation in biostatistics, big data analytics, computational statistics, and applied probability. My research interests center on high-dimensional data analysis, particularly in multiple hypothesis testing, differential expression analysis, and clustering techniques in microarray data analysis. Through my thesis, I have explored statistical methodologies tailored to the challenges of microarray experiments, applying advanced modeling techniques to extract meaningful insights from genomic data.





## ABSTRACT

Alina Kasian

### **Microarray Data Analysis: Statistical Approaches for High-Dimensional Gene Expression Data**

February 2025

Microarray technology has revolutionized the field of genomics by enabling quantitative measurement of gene expression levels for thousands of genes in a single experiment. This high-throughput technique has been instrumental in the understanding of biological processes, identification of biomarkers for diseases, and exploring molecular pathways of many conditions. Microarray data analysis, though extremely powerful, is riddled with serious statistical problems due to the high dimensionality of the data, inherent variability, and requirement of robust methodologies to extract meaningful biological information.

One of the most important challenges in microarray data analysis is that of multiple hypothesis testing. With a thousand genes being tested for differential expression at once, the application of traditional statistical methods leads to an increased rate of false positives, hence the need to apply corrections that balance the false discoveries vs. statistical power trade-off. The Benjamini and Hochberg (1995) False Discovery Rate (FDR) approach is one popular method to regulate the false positives and attain high sensitivity. In this thesis we examine different multiple hypothesis testing procedures from highly conservative traditional methods such as Bonferroni correction to cutting-edge state-of-the-art FDR-based strategies and assess their application to microarray data.



Another central aspect of microarray data analysis is the identification of differentially expressed (DE) genes. The limma package, which employs linear models and empirical Bayes variance moderation, has become a standard for differential expression detection in microarray research. In this thesis, limma is compared to the traditional t-tests, and their performances are evaluated under various settings, e.g., varying effect size, sample size, and noise. By using different simulation scenarios and applying the methods on real microarray data sets, we demonstrate the advantages of empirical Bayes methods in reduction of variance estimation uncertainty and increase in statistical power.

In addition to differential expression analysis, clustering plays a crucial role in identifying patterns in gene expression data. Traditional approaches such as k-means and hierarchical clustering are used regularly but suffer from drawbacks such as sensitivity to noise and the inability to determine the number of clusters. For solving these challenges, model-based clustering techniques such as Gaussian Mixture Models (GMMs) and advanced Bayesian techniques such as PUMA-CLUST provide a mathematical framework for microarray data cluster analysis. This thesis investigates various clustering techniques, assessing their performance on different situations, and shows how probabilistic modeling can be used to improve stability and interpretability of clusters.

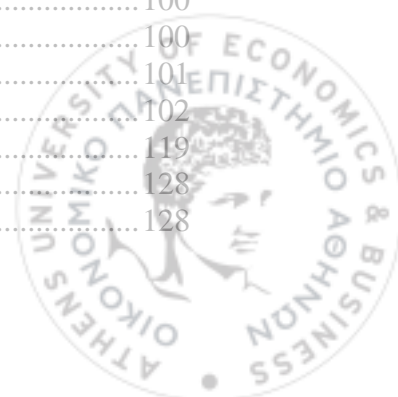
Overall, the thesis presents an extensive review of statistical and computational methods for the analysis of microarray data, from multiple hypothesis testing, differential expression analysis, to clustering. With theory and practice combined, we aim to explain the advantages and disadvantages of various methods, offering a structured framework for analysis of high-dimensional gene expressions. The findings contribute to the ongoing development of better and robust methodology for revealing biologically interpretable information from microarray experiments.





## TABLE OF CONTENTS

	<b>Page</b>
CHAPTER 1 .....	1
1.1 Introduction.....	1
1.2 Classical Approaches to Multiple Hypothesis Testing .....	2
1.2.1 Family-Wise Error Rate (FWER) Control .....	2
1.2.2 Limitations of FWER Control Methods .....	4
1.3 Modern Approaches: False Discovery Rate (FDR) Control.....	4
1.3.1 Introduction to False Discovery Rate (FDR) .....	5
1.3.2 Adaptive FDR and Storey's q-value.....	5
1.3.3 Positive False Discovery Rate (pFDR) .....	6
1.3.4 Advantages and Limitations .....	6
1.4 Extensions to FDR Control .....	7
1.4.1 Local False Discovery Rate (Local FDR).....	7
1.5 Application 1 .....	8
CHAPTER 2 .....	16
2.1 Introduction to Differential Expression Analysis .....	16
2.2 The Statistical Challenge of Differential Expression.....	17
2.3 The Limma package.....	20
2.3.1 Moderated t-statistics and Empirical Bayes Methods.....	20
2.3.2 Linear Models in limma.....	22
2.4 Normalization and Pre-processing .....	25
2.4.1 The importance of normalization.....	25
2.4.2 Normalization methods .....	25
2.5 Application 1: Differential Expression Analysis on Simulated Data .....	27
2.5.1 Part 1: Two Group Comparison .....	27
2.5.2 Part 2: Three Group Comparison (Multiple Contrasts) .....	49
2.6 Application 2: Differential Expression Analysis on Real Data .....	57
2.6.1 Global Tests to identify significant genes.....	58
2.6.2 Pairwise comparison to identify significant genes .....	63
CHAPTER 3 .....	74
3.1 Introduction to Clustering methods .....	74
3.2 Traditional Clustering methods.....	75
3.2.1 Hierarchical Clustering .....	75
3.2.2 K-means Clustering .....	81
3.2.3 K-Medoids Clustering.....	86
3.3 Model-Based Clustering .....	88
3.3.1 Gaussian Mixture Models (GMMS) .....	89
3.4 Advanced Model-Based Clustering Techniques.....	91
3.4.1 Parsimonious Gaussian Mixture Models (PGMMS) .....	91
3.4.2 Puma-Clust.....	93
3.4.3 EMMIX-Gene .....	96
3.4.4 Factor Analytic Bayesian Mixture Models (FABMix) .....	97
3.5 Preprocessing and Dimensionality Reduction .....	100
3.5.1 Preprocessing .....	100
3.5.2 Dimensionality Reduction .....	101
3.6 Application 1 Gene Clustering.....	102
3.7 Application 2 Sample/Tissue Clustering .....	119
APPENDIX.....	128
<b>Tables</b> .....	128



<b>Code</b> .....	131
Application Chapter 2.....	131
Application 1 Part 1 Chapter 3 .....	137
Application 1 Part 2 Chapter 3 .....	156
Application 2 Chapter 3.....	162
Application 1 Chapter 4.....	168
Application 2 Chapter 4.....	177
<b>References</b> .....	184





## LIST OF TABLES

Table 1 Number of Significant Genes Detected by Each Method.....	12
Table 2 Microarray Data simulated Data structure for two group comparison .....	29
Table 3 Metrics Table at different Effect sizes .....	34
Table 4 Metrics Table a different Variance Levels .....	38
Table 5 Metrics Table a different Sample sizes .....	43
Table 6 Metrics Table a different $\alpha$ levels .....	47
Table 7 Data Structure for three group comparison.....	50
Table 8 Summary Table Of DE Gene Counts .....	52
Table 9 Metrics Table .....	52
Table 10 Proportion of Significant Genes for every Alpha .....	62
Table 11 Number of significant genes identified vs NoL comparison .....	65
Table 12 Number of unique genes per comparison .....	71
Table 13 Jaccard Similarity Scores.....	113
Table 14 Internal Metrics per Clustering Method.....	114
Table 15 Adjusted Rand Index (ARI) per method.....	123
Table 16 Internal Clustering Metrics per method .....	124
Table 17 Distance Metrics for clustering .....	129
Table 18 Linkage methods for clustering .....	131





## LIST OF FIGURES

Figure 1 Mean Gene Expression per Sample Boxplot (AML vs CML) .....	10
Figure 2 Mean Gene Expression Scatterplot (AML vs CML).....	11
Figure 3 Number of Significant Genes Detected by each method.....	12
Figure 4 Overlap of Significant Genes Detected by Multiple Hypothesis Testing Methods.....	14
Figure 5 PCA plots for Multiple Hypothesis Corrections .....	15
Figure 6 Number of Significant Genes Detected by Limma vs Ttest at different Effect sizes.....	33
Figure 7 Metrics vs Effective Size.....	34
Figure 8 Mean difference versus standard deviation plots for various Effect size for Limma.....	36
Figure 9 Number of Significant Genes Detected by Limma vs Ttest at different Variance Levels .....	37
Figure 10 Metrics vs Variance Levels .....	39
Figure 11 Mean versus variance plots for different Variance Levels for Limma.....	41
Figure 12 Number of Significant Genes Detected by Limma vs Ttest at different Sample Sizes .....	42
Figure 13 Metrics vs Sample Sizes .....	44
Figure 14 Mean versus variance plots for different Sample Sizes for Limma .....	45
Figure 15 Limma Number of Significant Genes Detected by Limma vs Ttest at different $\alpha$ levels .....	46
Figure 16 Metrics vs $\alpha$ levels.....	48
Figure 17 Number of Significant Genes Detected by Limma vs Ftest.....	52
Figure 18 Roc Curve .....	53
Figure 19 Mean difference vs Variance for Control vs Treatment A .....	55
Figure 20 Mean difference vs Variance for Control vs Treatment B .....	56
Figure 21 Mean difference vs Variance for Treatment A vs Treatment B.....	57
Figure 22 Volcano plot with identified significant genes.....	60
Figure 23 MA plot with average log-expression vs log-fold change.....	61
Figure 24 Proportion of Significant Genes per alpha levels .....	63
Figure 25 Volcano plot ALL vs NoL .....	65
Figure 26 Volcano plot AML vs NoL.....	66
Figure 27 Volcano plot CLL vs NoL .....	67
Figure 28 Volcano plot CML vs NoL .....	68
Figure 29 Mean difference vs Standard Deviation for ALL vs NoL .....	68
Figure 30 Mean difference vs Standard Deviation for AML vs NoL .....	69
Figure 31 Mean difference vs Standard Deviation for CLL vs NoL .....	70
Figure 32 Mean difference vs Standard Deviation for CML vs NoL .....	71
Figure 33 Venn Diagram for unique and common genes identified.....	72
Figure 34 BIC vs. Number of Clusters plot for Puma .....	104
Figure 35 BIC vs. Number of Clusters plot for MClust .....	106
Figure 36 2D PCA of Clustered Data per method .....	115
Figure 37 3D PCA of Clustered Data per method .....	118
Figure 38 PCA and t-SNE visualization plot for Model based Clustering methods.....	127







## CHAPTER 1.

### MULTIPLE HYPOTHESIS TESTING IN MICROARRAY DATA ANALYSIS

#### 1.1 Introduction

In traditional hypothesis testing, researchers examine a single hypothesis at a fixed significance level, such as 0.05. However, when multiple hypotheses are tested simultaneously, applying the same significance level to each test increases the overall Type I error rate (falsely rejecting a true null hypothesis). As a result, several genes may be declared significant by chance, increasing the number of false discoveries.

For instance, in a microarray experiment with 10,000 genes, if the goal is to identify which genes are expressed differently between healthy and diseased tissues, using a significance level of 0.05 for each test would result in approximately 500 false positives purely by chance, even if no truly differentially expressed genes exist. This issue, known as the multiple comparisons problem, can severely compromise the reliability of the study's results.

Several statistical methods have been developed to address the problem of inflated false positives. The Family-Wise Error Rate (*FWER*) is one of the earliest and simplest approaches for controlling false discoveries and is defined as the probability of making at least one Type I error among all the tests performed. Many other methods have also been proposed, primarily modifying the significance level for each test according to the number of tests performed. Although these methods effectively control the *FWER*, they are often overly conservative, particularly for large numbers of hypotheses, as in microarray studies. Due to the conservativeness of these methods, the results may often lead to a high rate of false negatives, where real changes in gene expression are overlooked.



To overcome these limitations, the False Discovery Rate (*FDR*) was introduced by Benjamini and Hochberg in 1995. Whereas *FWER* primarily aims to prevent any false positives, *FDR* focuses on controlling the expected proportion of false positives among the hypotheses selected as significant. This approach allows for a better balance between detecting true effects and minimizing false discoveries and is particularly relevant to exploratory studies like microarray experiments, where the goal is to identify potential targets for further investigation.

This chapter will present various techniques developed to address the multiple comparisons issue, starting with traditional methods such as Bonferroni correction and Holm's method and progressing to more advanced approaches, including *FDR* correction and empirical Bayes methods. Each method will be presented alongside practical examples for microarray data analysis.

## 1.2 Classical Approaches to Multiple Hypothesis Testing

### 1.2.1 Family-Wise Error Rate (*FWER*) Control

The Family-Wise Error Rate (*FWER*) is one of the first statistics developed to address the problem of multiple comparisons. It is defined as the probability of making at least one Type I error among all simultaneously tested hypotheses. Mathematically, it is expressed as:

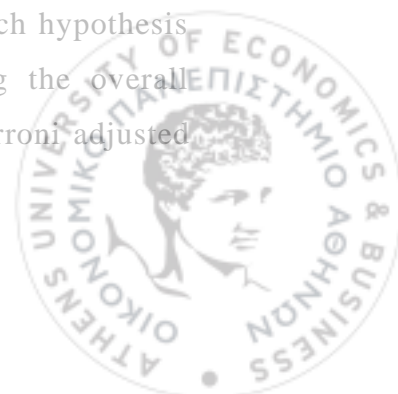
$$FWER = P(V \geq 1),$$

where  $V$  is the number of false rejections (Type I errors).

In the context of multiple testing, controlling *FWER* ensures that the probability of rejecting at least one true null hypothesis is less than the pre-specified level, usually  $\alpha$ .

#### *BONFERRONI CORRECTION*

The Bonferroni correction is one of the most popular methods of *FWER* correction. It functions by adjusting the significance level of each hypothesis test based on the total number of tests conducted. Denoting the overall significance level as  $\alpha$  and the number of tests as  $m$ , the Bonferroni adjusted significance level for each test is given by:



$$\alpha_{Bonferroni} = \frac{\alpha}{m}.$$

For instance, in a microarray study where 10,000 genes are under investigation and the fixed significance level  $\alpha$  is 0.05, the adjusted p-value for each gene would be:  $\alpha_{Bonferroni} = 0.05/10,000 = 0.000005$ .

While effective, this method is often criticized for being overly conservative, especially when  $m$  is large, as in many genomic studies. The conservative nature of the Bonferroni correction can lead to a high rate of false negatives, where true signals are missed due to the stringent significance threshold.

### *HOLM'S SEQUENTIALLY REJECTIVE PROCEDURE*

Holm (1979) proposed a stepwise modification of the Bonferroni correction, known as the Holm-Bonferroni method, which is less conservative while still controlling the *FWER*. Holm's procedure tests hypotheses sequentially, beginning with the smallest p-value, and compares each to an adjusted significance level that becomes less stringent as more hypotheses are tested:

1. Order the p-values from smallest to largest:  $p(1) \leq p(2) \leq \dots \leq p(m)$ .
2. Compare each p-value to its corresponding threshold:

$$\alpha_{Holm}(i) = \alpha / (m - i + 1)$$

3. Start from the smallest p-value. If the p-value is significant, then proceed to the next test. If however the test is non-significant, the testing process should stop and no further hypothesis should be evaluated.

For example, with an overall significance level of  $\alpha = 0.05$  and 10,000 tests performed, the first hypothesis would be tested at:  $\alpha_{Holm}(1) = 0.05/10,000$ .

If this hypothesis is rejected, the second hypothesis would be tested at:  $\alpha_{Holm}(2) = 0.05/9,999$ .

This process continues until a hypothesis is not rejected.

Holm's procedure is less conservative than the Bonferroni correction; it allows for less stringent adjustments as the testing progresses, especially for hypotheses with smaller p-values. This increases the test's power to detect true effects with little or no increase in the Type I error rate.

### *HOCHBERG'S STEP-UP PROCEDURE*



Hochberg's step-up procedure is more powerful than Holm's procedure in the sense that it is more favorable than the latter in controlling the family-wise error rate (*FWER*). Although both procedures use sequential testing with adjusted significance levels, Hochberg's method offers greater statistical power, particularly when test statistics are independent or positively correlated. The procedure is:

Sorting the p-values in ascending order:  $p(1) \leq p(2) \leq \dots \leq p(m)$  of the respective hypotheses  $H(1), \dots, H(m)$ .

Step-up comparison:

- Determine the largest index  $i$  which satisfies the following condition  $p(i) \leq \alpha / (m - i + 1)$ .
- If such an  $i$  exists, and hence the test is significant then all hypotheses with smaller p-values are also rejected; more specifically reject the Null Hypotheses  $H(j)$  for all  $j \leq i$ .

The main difference between Hochberg's and Holm's procedures is that all the previous p-values do not necessarily have to meet their respective criteria, thus more hypotheses can be rejected. This modification increases statistical power without compromising on the strong *FWER* control.

### 1.2.2 Limitations of FWER Control Methods

In high-dimensional data, such as microarray studies, *FWER*-based methods—including the Bonferroni correction, Holm's, and Hochberg's procedures—effectively control Type I errors. However, their conservativeness also leads to very low statistical power, which means that real effects may not be detected. In exploratory research where the aim is to form hypotheses and identify potential targets for further investigation, *FDR* (False Discovery Rate) control is often more suitable. This is because *FDR* control can help researchers detect more true positives without an increase in the false discovery rate.

## 1.3 Modern Approaches: False Discovery Rate (FDR) Control

Traditional methods tend to be overly conservative in high-dimensional data contexts. As a result, many alternative approaches have been developed to address multiple comparisons problems by balancing Type I error control and statistical power. The False Discovery Rate (*FDR*) control is considered one of



the most influential contemporary approaches, developed by Benjamini and Hochberg in 1995. The paper associated with the introduction of the *FDR* presents the *FDR* as a cornerstone of multiple hypothesis testing problems, especially in Genomics and Bioinformatics where large-scale testing is the norm.

### 1.3.1 Introduction to False Discovery Rate (FDR)

The Family-Wise Error Rate (*FWER*) differs from the False Discovery Rate (*FDR*) in that *FDR* controls the expected proportion of false positives among rejected hypotheses. This makes *FDR* particularly useful in exploratory analyses, where the goal is to identify as many true discoveries as possible at the cost of some false positives. Mathematically, the *FDR* is defined as:

$$FDR = E(V/R),$$

$V$  denotes the number of false positives, and  $R$  denotes the total number of rejections (both true and false positives). If no hypotheses are rejected ( $R = 0$ ), *FDR* is defined as zero.

The most common procedure for controlling *FDR* is the Benjamini-Hochberg (BH) procedure. The steps are as follows:

1. Order the  $p$ -values from all tests in increasing order:  $p(1) \leq p(2) \leq \dots \leq p(m)$ .
2. Find the largest  $k$  such that:  $p(k) \leq \frac{k}{m}\alpha$ , where  $\alpha$  is the desired *FDR* level, usually set at 0.05.
3. Reject all null hypotheses corresponding to  $p(1)$  through  $p(k)$ .

This procedure ensures that the expected proportion of false positives among the rejected hypotheses is controlled at the chosen *FDR* level,  $\alpha$ .

### 1.3.2 Adaptive FDR and Storey's $q$ -value

The original Benjamini-Hochberg method has one major disadvantage; that it assumes all hypotheses are equally likely to be true, meaning that  $\pi_0$ , the proportion of true null hypotheses is, 1. However, in many biological studies, not all hypotheses are actually null hypotheses. The adaptive *FDR* of Storey (2002) is an extension of the BH procedure that uses an estimate of  $\pi_0$  based on the data.



The estimate of  $\pi_0$  is based on the p-value distribution, assuming that p-values that are close to 1 are more likely to come from true null hypotheses. After  $\pi_0$  is estimated, then the *FDR* threshold is modified as follows:

$$\alpha_{\text{adaptive}} = \frac{\alpha}{\widehat{\pi_0}}$$

Here  $\alpha$  is the specified *FDR* level (for instance, 0.05), and  $\widehat{\pi_0}$  is an estimate of the number of true null hypotheses. Adaptive *FDR* techniques, including Storey's q-value, are useful in data settings where the majority of hypotheses are non-null, for instance, in microarray data. Therefore, an adaptive *FDR* procedure offers a more dynamic and powerful solution to the multiple comparison problem than a fixed *FDR* procedure that adapts accordingly to the data.

The q-value is closely related to the p-value but is specifically designed for multiple testing. It is the minimum *FDR* at which a given test can be deemed significant. The q-value gives a measure of significance that ranks hypotheses by their probability of being true positives, rather than using a strict threshold.

### 1.3.3 Positive False Discovery Rate (pFDR)

Alongside the q-value, Storey (2003) developed another measure called the Positive False Discovery Rate (*pFDR*). Whereas the conventional *FDR* includes all rejections irrespective of whether no null hypotheses have been rejected (i.e.,  $R = 0$ ), *pFDR* is defined only when at least one hypothesis has been rejected (i.e.,  $R > 0$ ). This refinement gives a more realistic idea of the number of false positives when results are considered significant.

Conceptually, the *pFDR* is defined as:

$$pFDR = E(V/R | R > 0),$$

where  $V$  is the number of false positives and  $R$  is the total number of rejections. This method can be more powerful than traditional *FDR* control for detecting meaningful results in large-scale studies. As a result, *pFDR* is most appropriate for exploratory analyses such as microarray studies, where the objectives are to narrow down the search for potential targets.

### 1.3.4 Advantages and Limitations



The main advantage of *FDR* and q-value procedures is that they capture the balance between discovery and false positive control. This makes them suitable for exploratory analyses such as those commonly implemented in genomics, where the objective is to produce potential hypotheses rather than conclude in definitive findings.

However, there is a limitation to *FDR* control: it requires p-values to be independent or positively dependent. The actual *FDR* may differ from the nominal level in complex dependency structures, which can make the interpretation of results more challenging. In addition, a correct estimation of  $\pi_0$  is essential for the reliability of q-values, and any inaccuracies in the estimation may result in incorrect conclusions.

#### 1.4 Extensions to *FDR* Control

As dataset dimensions and complexities increase in fields like microarray genetics, the need for improved *FDR* control procedures has become evident. The classical *FDR* procedure introduced by Benjamini and Hochberg (1995) serves as a foundational method, though several modern extensions have been proposed to enhance its adaptability and precision in multiple hypothesis testing. Local *FDR* approaches are among one of the few advances in the field that emphasize hypothesis-specific significance testing.

##### 1.4.1 Local False Discovery Rate (Local *FDR*)

The classical and adaptive *FDR* methods control the expected proportion of false discoveries across all rejected hypotheses, while local False Discovery Rate (local *FDR*) offers a more refined approach. Stephens (2017) introduced local *FDR* to estimate the probability that a specific hypothesis is a false discovery while permitting hypothesis-specific significance assessments. This method is particularly useful in research that necessitates the prioritization of certain hypotheses above others based on the individual evaluation of being true positives. Local *FDR* produces a conditional estimate of the probability that each hypothesis is a true positive, given its test statistic, as opposed to just controlling the overall rate of false discoveries across all rejections. This feature is valuable in exploratory studies, as in genomic research, for example, where many hypotheses are tested, but only a few warrant further investigation.



The local *FDR* for a given test statistic  $t$  is calculated as:

$$\text{Local FDR}(t) = \frac{\pi_0 f_0(t)}{f(t)},$$

where  $\pi_0$  is the estimated proportion of true null hypotheses,  $f_0(t)$  is the distribution of the test statistics under the null hypothesis, and  $f(t)$  is the observed distribution of the test statistics.

By focusing on the specific characteristics of each hypothesis, local *FDR* provides a more refined approach to significance testing, particularly in large-scale studies where hypotheses vary in biological relevance.

## 1.5 Application 1

### *DATASET DESCRIPTION*

The dataset used in this study consists of gene expression data from bone marrow samples of patients diagnosed with one of the four primary types of leukemia: Acute Lymphoblastic Leukemia (ALL), Acute Myeloid Leukemia (AML), Chronic Lymphocytic Leukemia (CLL), and Chronic Myeloid Leukemia (CML), along with samples from healthy controls (No Leukemia). The data were generated using the Affymetrix Human Genome U133 Plus 2.0 microarray platform, which measures the expression levels of over 20,000 genes across different patient samples. Mononuclear cells were isolated from the bone marrow using Ficoll density centrifugation and were then profiled for gene expression. The Microarray Innovations in Leukemia (MILE) study used this dataset: The MILE study was an international collaboration that sought to apply gene expression profiling for leukemia diagnosis and classification. The full microarray raw data is available at the NCBI Gene Expression Omnibus (GEO) under accession number GSE13159. The dataset can be found on the following website:

(<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE13159>).

### *HYPOTHESIS TESTING FRAMEWORK*

The main idea of the analysis is to compare the difference in expression of genes in Acute Myeloid Leukemia (AML) and Chronic Myeloid Leukemia



(CML) samples. The multiple comparison problem aims to identify the probability of false discoveries that occur when testing thousands of genes.

For this problem, we used several classical and modern techniques for multiple hypothesis testing, including:

- Family-Wise Error Rate (*FWER*) methods: Bonferroni, Holm, and Hochberg corrections.
- False Discovery Rate (*FDR*) methods: Benjamini-Hochberg (BH) procedure, adaptive *FDR* (Storey's q-value), and local *FDR*.

### *ANALYTICAL APPROACH*

The first step of the analysis was the data preprocessing, where the dataset was filtered to only include samples from AML and CML patients, which reduced the number of patient samples evaluated. Each patient sample's gene expression data was extracted and then used in combination with phenotypic data (leukemia type) to define the two groups for comparison. Gene-wise t-tests were then performed for every gene to compare expression levels between samples of AML and CML groups. The two sample t-tests were then used to produce p-values, which served as input for the multiple hypothesis testing corrections.

To address multiple comparisons, both classical and modern correction approaches were applied. To control Family-Wise Error Rate (*FWER*), the Bonferroni, Holm, and Hochberg corrections were applied to the p-values obtained from the t-tests. For the False Discovery Rate (*FDR*) control, the Benjamini-Hochberg (BH) procedure was employed to set a limit on the proportion of false positives among the rejected hypotheses. The adaptive *FDR* control was also used with Storey's q-value method to estimate the proportion of true null hypotheses and adjust the *FDR* threshold accordingly. Lastly, local *FDR* was computed to provide a hypothesis-specific estimate of false discovery probability.

Finally, the performance of each method was compared by evaluating the number of significant genes detected at a 5% significance level. This comparison provided valuable insight into how the various methods balanced false positive control with the power to detect true gene expression differences, highlighting the trade-offs inherent in each approach.



## Results

To compare AML and CML gene expressions at a broader scale, a boxplot of mean gene expression per sample was created (Figure 1). This visualization provides an overall picture of transcriptional activity across the two leukemia subtypes.

The median gene expression level is slightly higher in the CML group compared to the AML, meaning that, on average, genes are slightly more active in CML samples.

The IQR, representing the central 50% of expression values, is smaller in CML than in AML. This suggests that CML patients have less variation in gene expressions, or, in other words, more homogeneous gene expressions, as compared to AML patients.

Lastly, it appears that there are a few outliers, and most of them are from the CML group, where a few samples have lower mean expression than the majority.

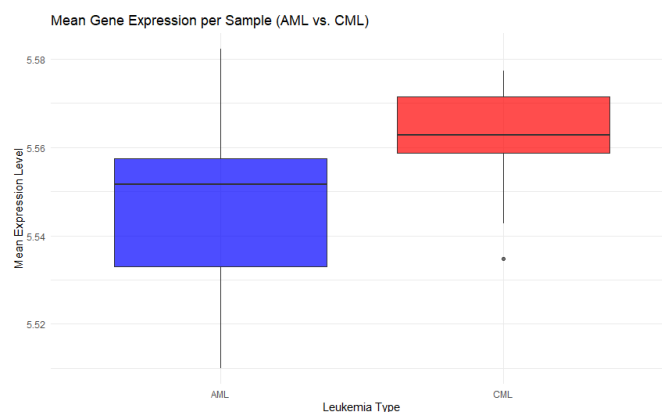


Figure 1 Mean Gene Expression per Sample Boxplot (AML vs CML)

To compare the global gene expression patterns between Acute Myeloid Leukemia (AML) and Chronic Myeloid Leukemia (CML), a scatter plot of mean gene expression per sample was created (Figure 2). This plot helps to determine whether gene expression levels differ between the two subtypes of leukemia. Every point in the scatter plot is a gene and its mean expression value in the AML group is represented on the x-axis and its CML mean expression on the y-axis. The red diagonal line represents the theoretical line where the expression of a gene is the same in both conditions.



As seen in the plot, most genes are located around the diagonal, meaning their expression is similar in AML and CML. This suggests that there is a strong correlation in transcriptional activity between the two types of leukemia, that may indicate that they have similar biological pathways and core regulatory mechanisms. However, some genes are located far away from the diagonal line, indicating that their expression is different in AML and CML groups. The genes above the diagonal indicate that they are overexpressed in the CML group, while those below the diagonal are overexpressed in the AML group. Some genes have large deviations from the diagonal line and hence may be considered as differentially expressed (DEs).

It should be noted that this scatter plot does not determine whether the observed differences are statistically significant. To formally identify DE genes, statistical tests (e.g., t-tests) must be performed together with multiple hypothesis correction methods (such as Bonferroni, Benjamini-Hochberg, or Storey's q-value).

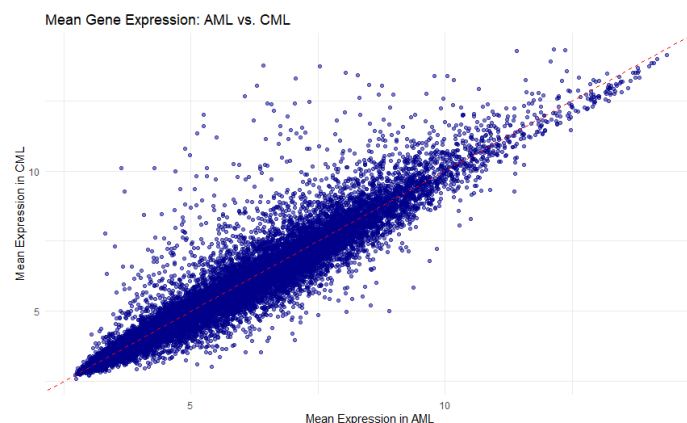


Figure 2 Mean Gene Expression Scatterplot (AML vs CML)

By comparing AML and CML samples' genes expression levels and by applying multiple hypothesis testing methods, the results of the analysis showed considerable variation in the number of significant genes detected. At a significance level of  $p < 0.05$ , the number of significant genes varied across methods, from 221 genes identified by the Bonferroni, Holm, and Hochberg methods to 5,051 genes detected by the adaptive *FDR* (q-value) method as shown in Figure 3 and Table 1.



<b>METHOD</b>	<b>SIGNIFICANT GENES (<math>P &lt; 0.05</math>)</b>
Bonferroni	221
Holm	221
Hochberg	221
Benjamini-Hochberg	3,458
Adaptive FDR (q-value)	5,051
Local FDR	129

Table 1 Number of Significant Genes Detected by Each Method

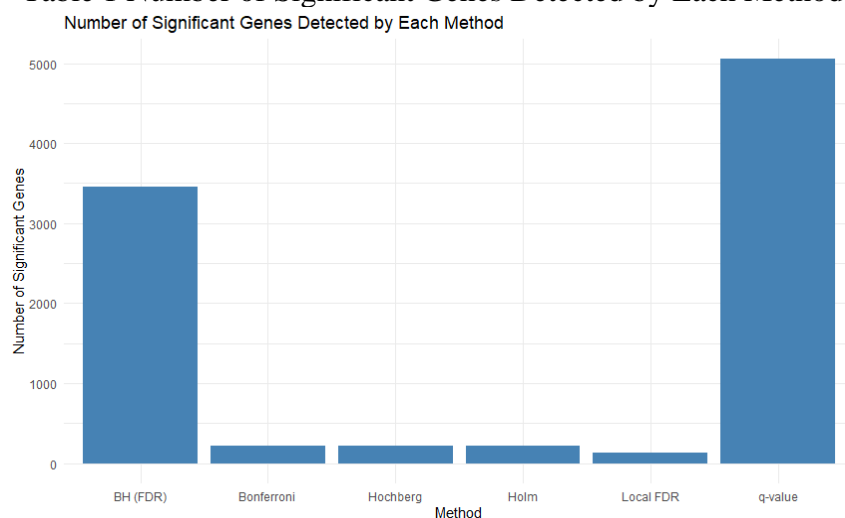
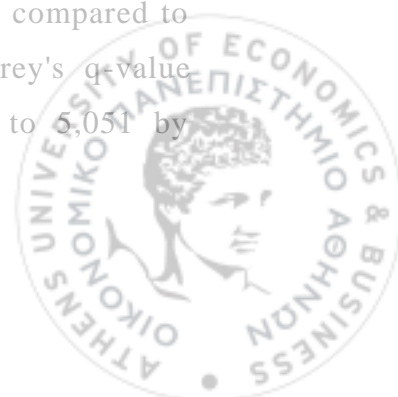


Figure 3 Number of Significant Genes Detected by each method

From results presented in Table 1 and Figure 3, it is evident that classical Family-Wise Error Rate (*FWER*) methods, such as Bonferroni, Holm, and Hochberg, are highly conservative, identifying only 221 significant genes. These methods strictly control the probability of making even a single Type I error, which is reflected in the relatively low number of significant findings. In contrast, False Discovery Rate (*FDR*) methods provide a more balanced approach by controlling the expected proportion of false positives among all rejected hypotheses. The Benjamini-Hochberg method detected 3,458 significant genes, demonstrating a substantial increase in power compared to *FWER* methods. Adaptive *FDR*, as implemented through Storey's q-value approach, further increased the number of significant genes to 5,051 by



estimating the proportion of true null hypotheses ( $\pi_0$ ) and adjusting the *FDR* threshold accordingly.

Local *FDR*, which controls the false positives for individual hypotheses, also detected only 129 significant genes. This result suggests that, under the current dataset's high dimensional settings, the local *FDR* method may provide overly conservative control, making fewer discoveries.

To further assess the consistency of significant gene detection across the methods employed, we used an UpSet plot, to visualize the intersection of significant genes (Figure 4) (Lex et al., 2014).

As can be seen from the Upset plot, the largest intersection, consisting of 3,232 DE genes identified, was common to both the Benjamini-Hochberg (*FDR*) and q-value methods. Furthermore, the q-value method was able to detect 1,593 additional genes that are unique and were not detected by the other methods, thus indicating that the adaptive *FDR* is less conservative and can detect a larger number of DE genes than the BH correction. In contrast, conservative methods such as Bonferroni, Holm, and Hochberg had a very small intersection of 124 genes, indicating that these methods are quite similar yet more restrictive. The strictest consensus set—only 97 genes—was detected by all methods, except local *FDR*. Lastly, the local *FDR* method found a small number of DE genes which only partially overlapped with other approaches, while the smallest intersection of 5 genes corresponded to *FDR* control-based methods (such as BH correction and Storey's q-value).

In conclusion, the UpSet plot displays the balance between sensitivity and specificity for the multiple correction methods employed; *FDR* -based methods identify substantially larger sets of significant genes, while Bonferroni, Holm, and Hochberg methods are more conservative and produce results that have very few overlaps with the other methods.



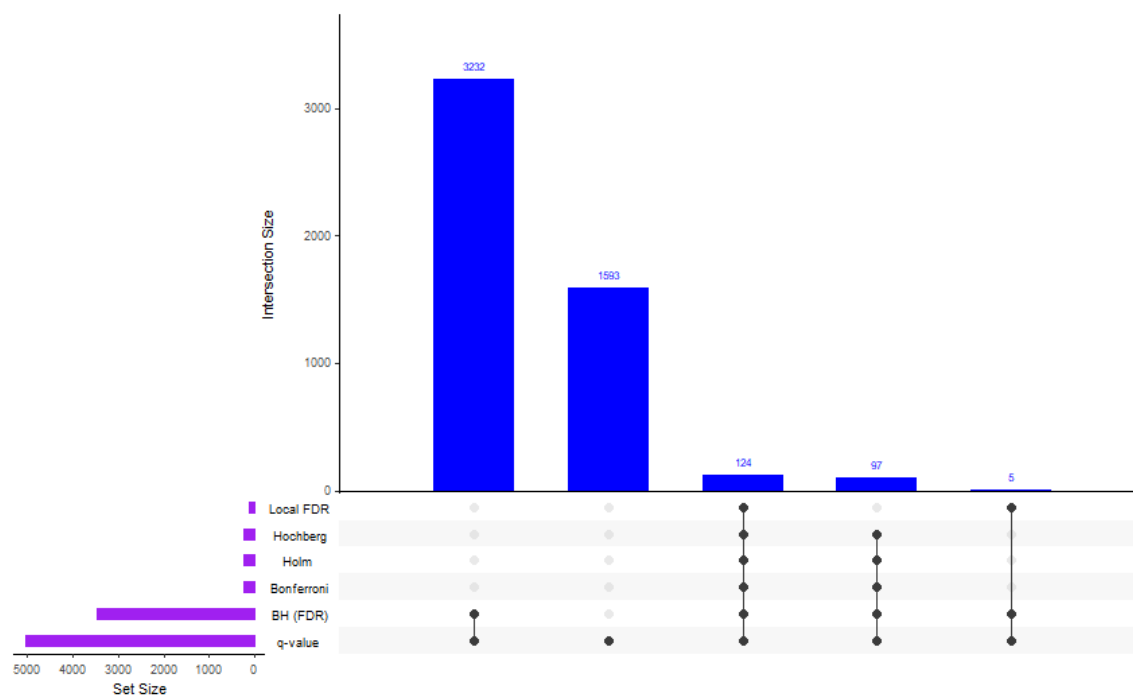


Figure 4 Overlap of Significant Genes Detected by Multiple Hypothesis Testing Methods

The PCA plots illustrate the distribution of genes in the principal component space, highlighting how different multiple hypothesis correction methods influence the selection of differentially expressed (DE) genes (Figure 5). Red points represent differentially expressed (DE) genes while grey points represent non-DE genes.

In the first plot (corresponding to *FWER* control), the distribution of DE genes is sparse. Any identified DE genes are concentrated at the periphery of the main data cloud, especially within regions that have more extreme PC1 and PC2 values. The central area remains almost entirely grey, indicating that most genes are not classified as DE. This suggests that only the most strongly deviating genes are detected by these methods. Moreover, it should be noted that the PCA plots for Holm and Hochberg procedures are not presented, given the fact that both methods identified the same number of DE genes with the Bonferroni method, and hence the illustrations produced are exactly the same in the PCA space.

The number of DE genes shows a clear increase in the second plot (BH method). Red points have expanded their range and extend further into the areas where gene data shows high density (main data point cloud). The identified DE genes



by the Bonferroni method were limited to the outer edge regions of the plot (as seen in the previous plot), while BH method detected DE genes near the main data cloud clusters. The general distribution pattern of DE genes expands throughout broader data sections of the plot.

The q-value plot shows a similar trend to the BH correction, although a more amplified version of it. The red points display extensive distribution throughout the PCA space, covering most areas except the smallest central part. The distribution of DE genes extends further into the main data cloud than the BH plot shows, merging with non-DE gene locations instead of remaining in the outer regions.

The last plot shows a Bonferroni-like pattern with minimal detected DE genes. The red points show limited distribution, with very few appearing within the main dense cluster of genes and extending more on the outer corners of the PCA space. This suggests that only a selective subset of genes, typically those positioned at the most extreme ends of the distribution, are classified as DE.

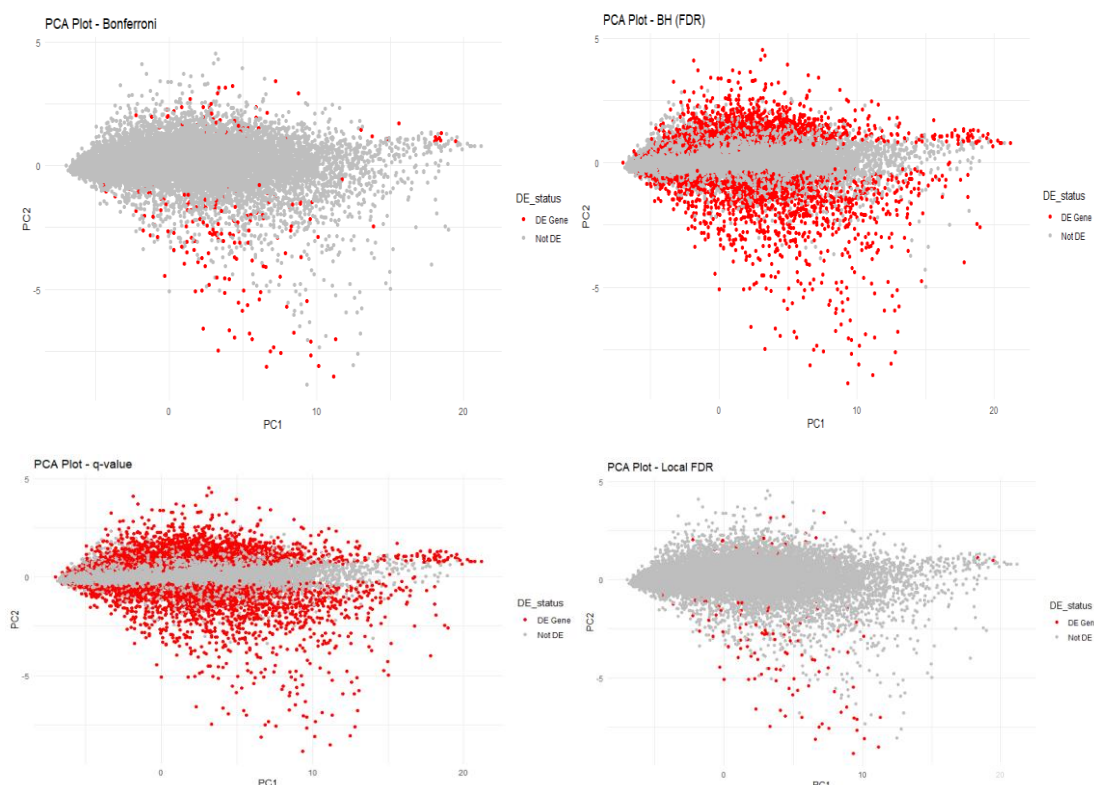


Figure 5 PCA plots for Multiple Hypothesis Corrections



## CHAPTER 2.

### STATISTICAL ANALYSIS FOR DIFFERENTIAL EXPRESSION

#### 2.1 Introduction to Differential Expression Analysis

Following the detailed discussion on multiple hypothesis testing corrections in Chapter 1, this chapter focuses on another vital part of microarray data analysis: differential expression. Microarray data analysis requires identifying genes whose expression levels change between biological conditions, and the difference in expression levels may help reveal the molecular mechanisms causing these phenotypic changes. Microarray experiments perform thousands of comparisons simultaneously with the aim of detecting the genes whose expression differs across experimental groups, while also leveraging the advantages offered by the multiple hypothesis testing correction frameworks. Microarray studies measure gene expression from multiple genes at once which creates challenges in detecting real biological signals due to technical noise and small sample sizes that often introduce random variation. Traditional methods, such as simple t-tests, struggle with this complexity, and often produce unreliable results. Hence, there is a need for more advanced approaches that provide high statistical power and accuracy when detecting differentially expressed (DE) genes.

This chapter describes the limma package, one of the most widely used methods for performing differential expression analysis of microarray data. The limma approach estimates gene expression changes by using linear models and empirical Bayes moderation. More specifically, limma's ability to borrow information from all genes to improve the precision of variance estimation, makes it a more reliable choice in differential expressions analysis than standard tests.



## 2.2 The Statistical Challenge of Differential Expression

In microarray data, differential expression analysis is defined as the comparison of gene expression levels across different biological conditions to identify those that are statistically significant. An important challenge is that of testing thousands of genes at once, while also trying to achieve precise results given the limited sample sizes and noisy data often associated with microarray gene data. This section explains the principles of differential expression analysis, including the importance of variance estimation, and describes the limma package as a state-of-the-art method for tackling these challenges.

As a key part of microarray studies, differential expression analysis is used to identify the molecular mechanisms underlying biological processes, disease states and treatment responses. However, several statistical challenges arise when dealing with the high-dimensional nature of microarray data.

### *HIGH DIMENSIONALITY AND LIMITED REPLICATES*

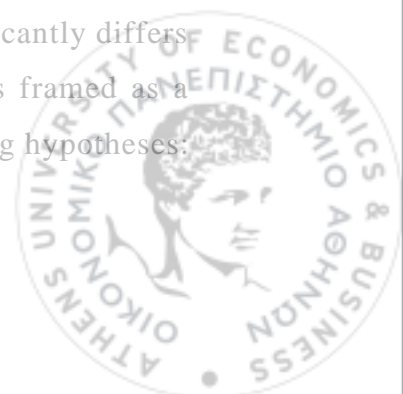
The first challenge in differential expression analysis comes from the nature of microarray data: while gene expression is measured for thousands of genes simultaneously, the number of biological replicates (samples) is often small. This imbalance between the number of genes (typically tens of thousands) and the number of samples (often fewer than 20) leads to a statistical problem known as the "curse of dimensionality".

When the number of replicates is limited, the estimation of gene-specific variances becomes unreliable. For many genes, the sample variance may either be overestimated or underestimated, leading to inflated or deflated test statistics, respectively. As a result, the analysis may either miss truly differentially expressed genes (low statistical power) or incorrectly identify genes as significant due to random fluctuations (high false positive rate).

### *HYPOTHESIS TESTING FRAMEWORK*

For each gene, we aim to determine whether its expression significantly differs between two or more experimental conditions. Formally, this is framed as a hypothesis testing problem. For each gene  $g$ , we test the following hypotheses:

- The null hypothesis  $H_0$ :



$$H_0: \mu_g(1) = \mu_g(2)$$

Where  $\mu_g(1)$ ,  $\mu_g(2)$  represent the mean expression levels of gene  $g$  under the two experimental conditions.

- The alternative hypothesis  $H_1$ :

$$H_1: \mu_g(1) \neq \mu_g(2)$$

which indicates that the expression levels of gene  $g$  are statistically different between the two conditions.

For every gene, a test statistic is calculated, and from this statistic, a p-value is obtained to determine the significance of the difference in expression levels under the null hypothesis. However, given the large number of genes that are being tested at once, these p-values need to be carefully interpreted to avoid the inflation of false positives.

### *MULTIPLE TESTING PROBLEM*

In microarray experiments, the most critical challenge is the multiple testing problem. With thousands of genes undergoing simultaneous assessment of differential expression and when each gene's test maintains an independent standard significance threshold (e.g.,  $\alpha = 0.05$ ), the false positives identified at each step increase significantly.

A critical strategy to manage the excessive false positive findings is applying multiple hypothesis testing corrections like the False Discovery Rate (*FDR*) control (Benjamini and Hochberg, 1995). These corrections adjust the significance thresholds to prevent false discoveries and enable the detection of actual true signals. More details are provided in Chapter 1.

### *VARIANCE ESTIMATION AND THE IMPACT ON T-STATISTICS*

Traditional differential expression analysis employs a t-test to compare mean expression values between two conditions for every gene. The test statistic for gene  $g$  is calculated as:

$$t_g = \frac{\hat{\mu}_g^{(1)} - \hat{\mu}_g^{(2)}}{\sqrt{s_{p,g}^2 \left( \frac{1}{n_1} + \frac{1}{n_2} \right)}}$$



where  $\hat{\mu}_g^{(1)}, \hat{\mu}_g^{(2)}$  are the sample means of gene  $g$  in the two groups,

$s_{p,g}^2 = \frac{(n_1 - 1)s_{g,1}^2 + (n_2 - 1)s_{g,2}^2}{n_1 + n_2 - 2}$  is the pooled variance estimate for gene  $g$ ,

$s_{g,1}^2, s_{g,2}^2$  are the sample variances in the two conditions,  $n_1$  and  $n_2$  represent the corresponding sample sizes.

The modified t-test known as Welch's t-test is applied when the equal variances assumption does not hold:

$$t_g = \frac{\hat{\mu}_g^{(1)} - \hat{\mu}_g^{(2)}}{\sqrt{\frac{s_g^{(1)2}}{n_1} + \frac{s_g^{(2)2}}{n_2}}}$$

where  $s_g^{(1)2}$  and  $s_g^{(2)2}$  represent the separate sample variance estimates from each condition.

While the standard t-test is widely used in microarray studies, it relies on the assumption that the variance estimates for each gene are stable. The small number of replicates in microarray experiments produces unstable variance estimates which lead to inaccurate t-statistics. Genes with low expression levels are especially vulnerable to these issues because minimal changes in their expression values may create artificially high t-statistics that ultimately produce false significance results. Empirical Bayes approaches implemented in the limma package solve these problems by performing variance stabilization.

### *EMPIRICAL BAYES AS A SOLUTION*

To tackle the problem of the instability of variance estimates in high-dimensional data, empirical Bayes methods have been developed, most notably in the limma package (Smyth, 2004). By using empirical Bayes, the t-statistics for every gene are moderated through more stable estimates of variance, obtained by pooling information across all genes. This approach, known as moderation, decreases the effect of outlier variance estimates and produces more accurate results in differential expression analysis, especially when the number of replicates is small.



## 2.3 The Limma package

### 2.3.1 Moderated t-statistics and Empirical Bayes Methods

A significant advancement in the analysis of differential gene expression involves the application of empirical Bayes methods to stabilize variances across thousands of genes. The approach implemented in the limma package (Smyth 2004) addresses unstable variance estimates by ‘borrowing’ information from all genes in the dataset. This section explains the fundamental concepts of moderated t-statistics and how empirical Bayes methods improve both sensitivity and specificity in differential expression analyses.

#### *THE EMPIRICAL BAYES APPROACH*

Unstable variance estimates are addressed through the use of empirical Bayes in the limma package. Empirical Bayes primarily functions on the premise that each gene’s variance is drawn from a shared distribution. By extracting information from all genes, empirical Bayes estimates a common variance component, which is then combined with gene-specific variance estimates to achieve variance stabilization. The process of empirical Bayes produces moderated variances that are proven to be more accurate and consistent than those calculated separately for every gene by using standard approaches.

In the empirical Bayes framework, each gene’s variance estimate  $s_g^2$  is shrunk toward a shared variance estimate  $s_0^2$  that is computed from the overall variance distribution of all genes. The quantity of information available for each gene directs the level of shrinkage in this process. More specifically, genes that have few replicates or high variability will experience stronger shrinkage toward the pooled estimate, while genes with more moderate variance estimates will exhibit less shrinkage. The result of the process is the production of more stable t-statistics that overall enhance the performance of differential expression analysis.

#### *MODERATED T-STATISTICS*

The moderated t-statistic in the limma package is calculated similarly to the standard t-statistic, but with a moderated variance in the denominator:



$$t_g = \frac{\widehat{\mu}_g^{(1)} - \widehat{\mu}_g^{(2)}}{\sqrt{\frac{s_g^2 + s_0^2}{n_1} + \frac{s_g^2 + s_0^2}{n_2}}}$$

Where:

$\widehat{\mu}_g^{(1)}, \widehat{\mu}_g^{(2)}$  are the sample means of gene  $g$  in the two conditions

$s_g^2$  is the sample variance estimate for gene  $g$

$s_0^2$  is the pooled variance estimate (the "prior" estimate from the empirical Bayes framework),

$n_1$  and  $n_2$  are the sample sizes for the two groups.

The use of moderated variances in the t-statistic ensures that genes with unreliable variance estimates (due to low expression or few replicates) are stabilized by the pooled estimate, while allowing genes with reliable variances to retain their original estimates. This leads to more consistent and reliable results across all genes.

### *BAYESIAN SHRINKAGE*

The term "Bayesian shrinkage" refers to the process of shrinking the gene-specific variance estimates towards a pooled estimate. In the context of limma, this shrinkage is performed using the empirical Bayes method, where the variances are treated as random variables drawn from a common distribution. The level of shrinkage depends on the gene-specific variance estimates as well as the amount of data that is available for each gene. The empirical Bayes approach is especially useful in high-dimensional data such as microarrays, where there is usually a limited number of samples compared to the large number of genes.

Shrinking the variance estimates towards a common value increases the stability of the t-statistics and reduces the false positive rate caused by random variability. This is particularly useful for microarray analyses given that the objective of these studies is often to detect hundreds of differentially expressed genes from among many thousands and, as such, the detection of false positives is quite high.



## *PRACTICAL IMPLICATIONS FOR DIFFERENTIAL EXPRESSION ANALYSIS*

The limma package's moderated t-statistics have the following advantages in practical applications for microarray analyses:

1. **Increased Power:** The stabilization of variance estimation improves the ability to detect true DE genes, and this is particularly important for experiments with a limited number of replicates.
2. **Reduced False Positives:** The empirical Bayesian approach reduces false positives by shrinking the inflated variance estimates for genes with low expression or small sample sizes.
3. **Improved Consistency:** The use of moderated variances for all genes leads to more uniform results, which enables researchers to identify real changes without being misled by random noise.

The moderated t-statistics used in the limma package form the backbone of differential expression analysis and offer more robust results compared to conventional approaches, such as traditional t-tests. The empirical Bayes approach ensures that microarray gene analysis provides highly accurate results.

### 2.3.2 Linear Models in limma

Traditional t-tests have the disadvantage of being used only for comparing two conditions, but biological experiments are often more complex including time courses, multiple experimental groups or factorial designs with interactions. The limma package offers a flexible and robust approach to analyzing such experiments by using linear models. As such, researchers can expand on differential expression analysis beyond simple pairwise comparisons and analyze many different experimental setups.

### *THE LINEAR MODEL FRAMEWORK*

The basic assumption of a linear model is that the expression level of a gene  $g$  is a linear function of the explanatory variables (covariates) that describe the experimental conditions. Formally, for each gene  $g$ , the linear model is expressed as:

$$Y_g = X\beta_g + \varepsilon_g$$



where:

- $Y_g$  is a vector of observed expression values for gene  $g$  across all samples,
- the design matrix  $X$  contains the experimental conditions for each sample,
- the vector of coefficients (or regression parameters) for gene  $g$ , denoted by  $\beta_g$ , quantifies the effect of each experimental condition on the expression of gene  $g$ ,
- the vector of random errors (residuals) for each gene is denoted by  $\varepsilon_g$

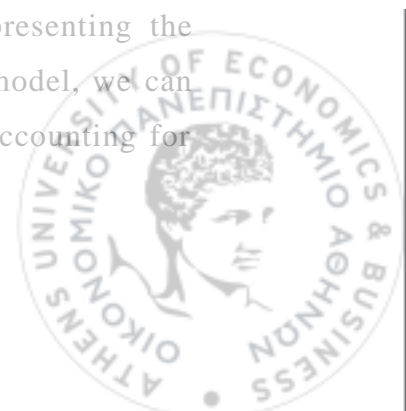
and is assumed to follow a normal distribution with mean zero and variance  $s_g^2$ . The main advantage of using linear models is that they can provide flexible representations of the different types of experimental structures. The linear model can be used to compare treatment means, carry out analyses involving batch effects, or examine interactions between variables by encoding the experiment's structure into the design matrix  $X$ .

### *BUILDING THE DESIGN MATRIX*

The experimental conditions in the linear model are determined by the design matrix  $X$ . Each sample is represented as a row in the matrix, while the experimental factors such as conditions or time points appear as column elements. For example, in a two-condition experiment consisting of control and treatment groups the design matrix  $X$  takes the form:

$$X = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

In this scenario the initial column denotes the intercept, which represents a standard baseline expression level throughout all samples, while the second column denotes the treatment effect, with the value of 1 representing the treatment group and 0 the control group. By fitting this linear model, we can estimate the effect of the treatment on gene expression while accounting for overall baseline expression levels.



This matrix-based approach used in the linear models employed in the limma package, offers flexibility since it can encode numerous experimental configurations, such as time-course experiments or studies with multiple groups. The design matrix can be updated accordingly to include additional columns representing specific factors or interactions.

### *FITTING THE LINEAR MODEL WITH LIMMA*

After the design matrix  $X$  is defined, limma fits the linear model for each gene using ordinary least squares (OLS). The coefficients  $\beta_g$  are estimated by minimizing the residual sum of squares between the observed expression values and the predicted values from the model. Specifically, the coefficients are estimated as:

$$\hat{\beta}_g = (X^T X)^{-1} X^T Y_g.$$

Where  $X^T$  is the transpose of the design matrix. These estimated coefficients reflect the effects of the experimental conditions on gene  $g$ , so that we can determine if the expression of the gene is significantly different across conditions.

After the model is fitted the empirical Bayes method is used to moderate the gene-wise variances as described in Section 2.3.1.

### *ADVANTAGES OF LINEAR MODELS IN LIMMA*

Some advantages of using linear models in limma for the purpose of differential expression analysis are:

1. **Flexibility:** It can model many types of experiments including those that may compare treatments, time course studies and even factorial designs.
2. **Handling Batch Effects:** The design matrix can include batch effects (non-biological factors influencing gene expression). Accounting for these effects in the model reduces their impact on the analysis.
3. **Efficient Analysis of Multiple Comparisons:** Instead of performing separate t-tests for each pairwise comparison, linear models allow all conditions to be compared simultaneously. This reduces the computational burden and simplifies the interpretation of results.



4. Moderation of Variance Estimates: As described previously, limma uses empirical Bayes moderation to stabilize variance estimates, improving the accuracy and reliability of differential expression analysis.

## 2.4 Normalization and Pre-processing

### 2.4.1 The importance of normalization

Microarray experiments generate vast amounts of data, with thousands of genes measured across multiple samples. However, the raw data collected from microarrays often include various technical biases. For example:

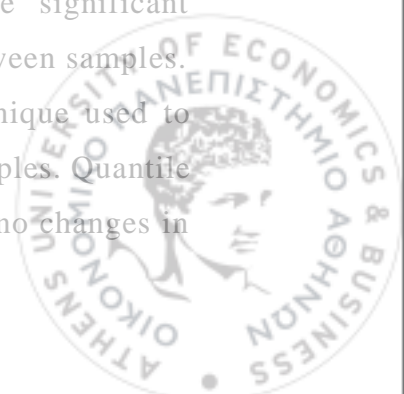
- Batch effects: Differences in experimental conditions (e.g., samples processed on different days or in different labs) can introduce systematic biases that affect the expression levels of multiple genes simultaneously.
- Hybridization efficiency: Variations in the efficiency with which RNA molecules hybridize to probes can lead to inconsistent signal intensities across samples.
- Spotting errors and scanning variations: Errors during the printing or scanning of microarrays can introduce noise into the data.

Normalization is an essential initial data pre-processing step, designed to remove technical biases introduced during data collection. Moreover, it ensures that the observed differences arise from actual biological relationships within the data and not from incorrect conclusions.

### 2.4.2 Normalization methods

In microarray data analysis, multiple normalization strategies exist, which target different types of technical noise. The selection of a normalization procedure directly depends on the research question, the experimental design and data characteristics.

In global mean normalization, the samples' expression values are adjusted so that their overall (mean or median) expression levels become comparable across samples. This approach effectively handles situations where significant variations occur in RNA concentration or labelling intensity between samples. Another widely used method is quantile normalization, a technique used to equalize the distribution of gene expression values among all samples. Quantile normalization is based on the assumption that most genes show no changes in



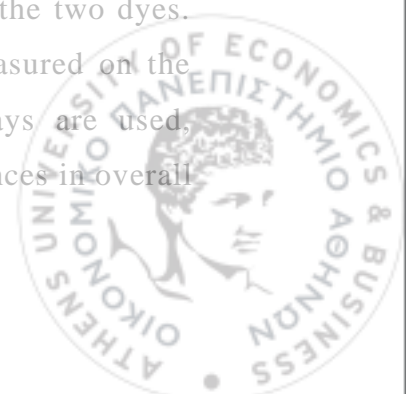
expression between conditions and uses this assumption to remove systematic biases between samples by realigning their distributions to a common reference. The method is particularly useful for eliminating batch effects and it is often used as a standard step in the limma workflow to normalize data before differential expression analysis.

In cases where biases are intensity-dependent, Loess (Locally Weighted Scatterplot Smoothing) normalization is often the preferred approach. This method is particularly useful for two-color microarrays, where variations in signal intensity depend on the expression levels of genes. The Loess normalization technique removes dye biases so that expression comparisons between samples remain authentic and unbiased by signal intensity.

Finally, Robust Multi-array Average (RMA) is a detailed normalization approach that has been developed especially for Affymetrix microarray platforms. The method includes three fundamental operation steps which consist of background adjustment and quantile normalization, together with probe-level data summarization. After applying these adjustments, RMA assumes that the expression data across different arrays are comparable, and hence, it is widely used as a standard method to analyze gene-level data in large microarray experiments.

#### *PRE-PROCESSING IN LIMMA*

The limma package provides an integrated framework for preprocessing microarray data, that includes several normalization methods for improving the quality of the data before performing differential expression analysis. The pre-processing begins with background correction, which removes nonspecific binding and noise that may affect intensity measurements. Background correction removes signals that are not related to the hybridization of the target RNA and thus improves the accuracy of the expression measurements. In two-color microarray experiments, within-array normalization is used to remove technical variation due to differences in labelling efficiency of the two dyes. This ensures that comparisons between the two conditions measured on the same array are bias-free. In experiments where several arrays are used, between-array normalization is performed, to bypass any differences in overall



intensity between the arrays. This is particularly important in multi-array studies, as it guarantees that all the samples have comparable expression values. These pre-processing steps in limma form a strong basis for the subsequent analysis, ensuring accurate differential expression analysis results.

## 2.5 Application 1: Differential Expression Analysis on Simulated Data

### 2.5.1 Part 1: Two Group Comparison

In this first application, the performance of the differential expression analysis methods was assessed using simulated microarray datasets. The simulation-based approach provides a benchmark for understanding the performance of methods such as limma and t-tests, under various conditions, to identify differentially expressed (DE) genes, while also managing false discoveries. These simulations are designed to mimic both the statistical and biological complexities observed in high-dimensional gene expression data analysis.

#### *SIMULATION FRAMEWORK*

The simulation framework was carefully designed to replicate the features of microarray data and assess the robustness of statistical methods under varying assumptions. Data was generated for 10,000 genes, representing the high-dimensional nature of typical microarray datasets. Two experimental groups—control and treatment—were simulated, each with equal sample sizes, to eliminate confounding effects due to unequal group sizes. Ten percent of the genes were designated as truly DE, while the remaining genes served as non-DE.

To introduce biologically plausible differences, expression levels for DE genes in the treatment group were shifted by predefined effect sizes (small, moderate, and large). Normal distributions were used to generate gene expression values, aligning with the assumptions underlying linear modeling and hypothesis testing methods. Although biological data may deviate from perfect normality, normalization techniques often applied in microarray preprocessing help justify this assumption based on the Central Limit Theorem.



Gene-specific variances were modeled using an inverse gamma distribution, which provides a flexible way to generate both highly variable and stable genes, reflecting the heterogeneity observed in real-world datasets. This approach ensured the inclusion of realistic variability maintaining controlled noise levels across scenarios.

The simulated dataset's structure (Table 2) includes unique gene identifiers, gene-specific variances, DE status, and expression values for control and treatment groups.

<i>Column Name</i>	<i>Description</i>
Gene_ID	Unique identifier for each gene (e.g., Gene_1, Gene_2, ..., Gene_n).
Variance	Gene-specific variance, drawn from a Gamma distribution.
Is_DE	Binary indicator (1 = Differentially Expressed, 0 = Not Differentially Expressed).
Control_1	Simulated expression value for Control Group, Sample 1.
Control_2	Simulated expression value for Control Group, Sample 2.
Control_3	Simulated expression value for Control Group, Sample 3.
Control_4	Simulated expression value for Control Group, Sample 4.
Control_5	Simulated expression value for Control Group, Sample 5.
Treatment_1	Simulated expression value for Treatment Group, Sample 1.
Treatment_2	Simulated expression value for Treatment Group, Sample 2.
Treatment_3	Simulated expression value for Treatment Group, Sample 3.
Treatment_4	Simulated expression value for Treatment Group, Sample 4.
Treatment_5	Simulated expression value for Treatment Group, Sample 5.

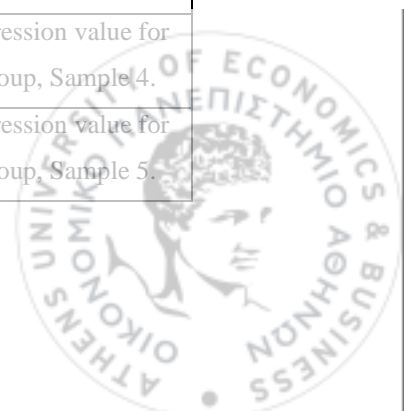


Table 2 Microarray Data simulated Data structure for two group comparison

### *SIMULATION PARAMETERS*

The simulation was designed to evaluate the performance of statistical methods for differential expression analysis under varying experimental conditions, focusing on four key factors:

- **Effect Sizes:** Effect sizes were systematically varied—small (0.5), moderate (1.5), and large (3.0)—to capture a range of biological scenarios from small to more pronounced changes in gene expression. A small effect size (0.5) corresponds to minimal shifts in mean expression levels, challenging the sensitivity of statistical methods to detect weak signals. Moderate (1.5) and large (3.0) effect sizes represent increasingly discernible differences in expression, providing a framework to evaluate the robustness of methods like limma and the t-test in detecting more pronounced biological signals.
- **Variance Levels:** The variances of each gene were drawn from an inverse gamma distribution, since this distribution can model the observed gene expression variability in the biological datasets used for differential expression analysis. The inverse gamma distribution is useful for modeling a broad range of variances, from stable (low variance) to highly noisy (high variance).
  - o **Low Variance:** A high shape parameter and low scale parameter (e.g., *shape* = 8, *scale* = 0.5) were used for one of the simulation scenarios, which produced a distribution of values that is mostly centered around small values. These variances showed little variation and remained small in value. This approach represents genes that show uniform and tightly regulated expression patterns, which could be involved in highly conserved biological processes. Any differences occurring in this specific scenario would most certainly be detected since any remaining noise levels are minimized.
  - o **Moderate Variance:** By using a more balanced shape and scale (e.g., *shape* = 4, *scale* = 0.9), the resulting model produced a broader distribution of variances. In this scenario, the generated variance levels reflect biological variability, where genes exhibit both stable and moderately noisy expression patterns. This model provides a realistic



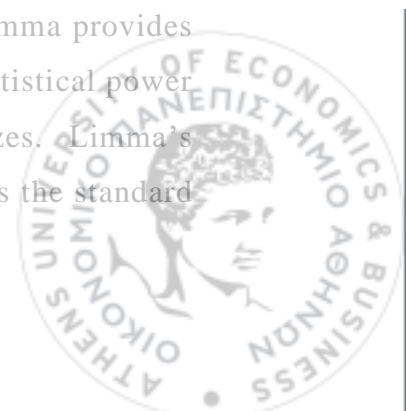
representation of most genes that appear in microarray experiments, where variation emerges from technical and biological sources.

- o High Variance: The combination of a low shape parameter with a high scale parameter produced a distribution that has increased dispersion, with larger variance values in comparison to the previous scenarios, (e.g.,  $shape = 2$ ,  $scale = 3$ ). The described conditions reflect genes with unpredictable expression levels, commonly found in regulatory pathways influenced by external factors. The presence of high variance generates substantial noise, making it difficult for statistical methods to identify true DE genes among false positive results.
- Sample Sizes: The simulations used small (5 samples per group), moderate (10 samples per group), and larger (20 samples per group) sample sizes. The selection of these options aims to demonstrate how statistical methods perform under typical experimental microarray study setups, specifically with increasing sample sizes.
- FDR Thresholds: The analysis focused on varying significance thresholds of 0.01, 0.05 and 0.1, to evaluate how specificity and sensitivity levels are affected by method performance. The different thresholds aimed to provide insights into how different false discovery controls (especially the conservativeness introduced by each threshold) affected each method's performance.

### *METHODOLOGICAL APPROACH*

The methodological framework's main purpose was to compare the performance of two widely used statistical approaches for differential expression (DE) analysis: limma and the standard two-sample t-test. The evaluation process assessed their performance through simulated scenarios to examine their strengths and limitations.

The limma method applies linear modelling together with empirical Bayes variance moderation. By borrowing information across genes, limma provides more accurate and stable variance estimates, leading to better statistical power and improved robustness, especially with small sample sizes. Limma's advantage lies in the empirical Bayes approach, which moderates the standard



errors and provides improved shrinkage estimates for DE analysis, which is critical in high-dimensional datasets like microarrays. In contrast, the two-sample t-test, though more traditional, remains widely applied in microarray studies. To adjust for multiple comparison, the Benjamini-Hochberg method was employed to control the *FDR* at predefined *FDR* thresholds for both limma's and t-test's outputs.

The performance of these methods was evaluated using three key metrics. The statistical power of each method was evaluated through sensitivity, which represents the ratio of correctly identified DE genes. Additionally, each method's ability to correctly classify non-DE genes as non-significant was also measured through specificity, indicating the method's ability to control false positives. Finally, the False Discovery Rate quantified each method's reliability by estimating the proportion of false positives among significant results. Each metric is calculated as seen below:

$$\text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$$

$$\text{FDR} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Positives}}$$

## RESULTS

### *IMPACT OF EFFECT SIZE ON DIFFERENTIAL EXPRESSION ANALYSIS*

The simulation results of varying effect sizes (0.5, 1.5, and 3.0) show the differences in the performance of the t-test and limma in detecting truly DE genes. Results are presented in tabular form and graphically displayed using Venn diagrams for each effect size (Figure 6).

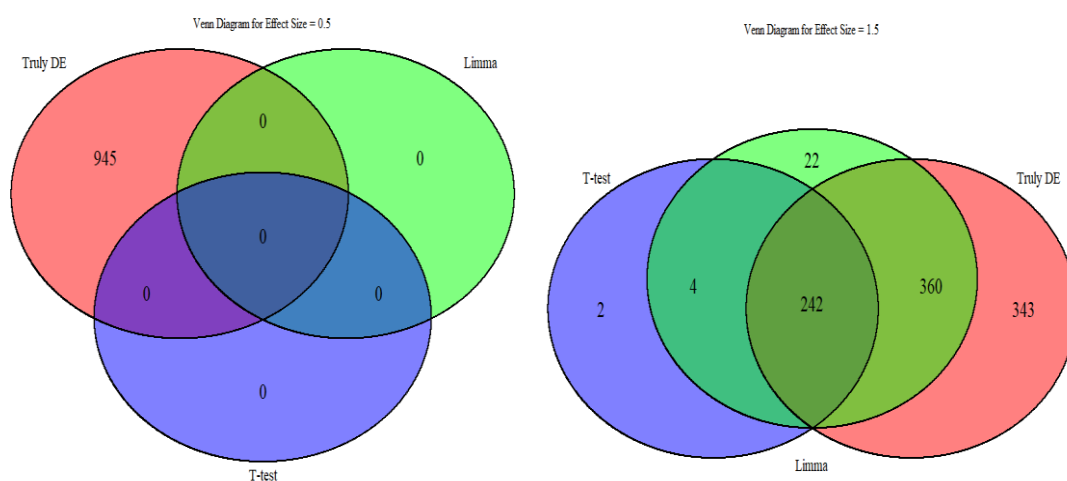
At a small effect size of 0.5, neither limma nor the t-test detected any DE genes. This result is expected because any difference between the control and treatment groups' expression levels was not sufficient to provide strong statistical power for either method to detect significant genes. This condition highlights the shortcomings of both methods when the biological signal is not strong. This result aligns with expectations, as the minimal difference in



expression levels between the control and treatment groups did not provide sufficient statistical power for either method to identify significant genes.

For a moderate effect size of 1.5, both methods began to identify DE genes, with limma detecting 628 genes compared to 248 for the t-test. This significant difference indicates the greater sensitivity of limma, which benefits from its empirical Bayes variance moderation, particularly in smaller sample sizes. Further analysis in the form of a Venn diagram revealed a large overlap, with 242 genes being found by both methods. However, limma demonstrated clear advantages by detecting an additional 360 true DE genes that were missed by the t-test. The t-test, on the other hand, identified only 2 unique DE genes, indicating its limited detection power in this scenario.

At a large effect size of 3.0, both methods performed well, achieving high detection rates. Limma identified 971 DE genes, while the t-test detected 894. The overlap between the two methods increased significantly, with 871 genes being identified as DE by both approaches. Despite this improvement in performance for the t-test, limma maintained its sensitivity advantage, identifying 57 additional true DE genes compared to the t-test, which detected only 11 unique DE genes. These findings highlight limma's robustness and continued superiority in detecting true DE genes.



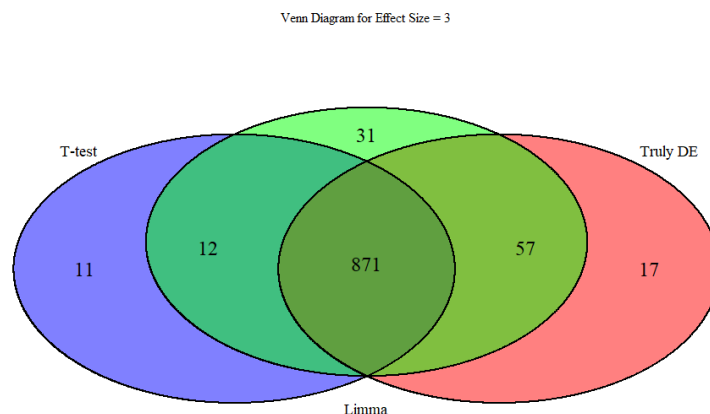


Figure 6 Number of Significant Genes Detected by Limma vs Ttest at different Effect sizes

The performance metrics for sensitivity, specificity, and False Discovery Rate (*FDR*), as presented in Table 3 and Figure 7, provide a detailed overview of trends across varying effect sizes. Sensitivity was consistently greater for limma across all conditions. At a very large effect size of 3.0, limma's sensitivity was 98.2%, which was much higher than that of the t-test at 92.1%. This trend was even more pronounced at a moderate effect size of 1.5, where limma's sensitivity (63.7%) was almost double that of the t-test (25.6%). Both methods had high specificity (>99.5%) for all the effect sizes with great control over false positives under any magnitude of the effect size. For the *FDR*, the t-test exhibited slightly better performance at moderate and large effect sizes, though the differences were marginal. More specifically, at an effect size of 3.0, the *FDR* for the t-test was 2.4%, compared to 4.1% for limma. These results highlight limma's superior sensitivity while maintaining competitive specificity and *FDR*, making it more effective for detecting true DE genes, particularly under moderate to large effect sizes.

<i>Method</i>	<i>Effect Size</i>	<i>Sensitivity</i>	<i>Specificity</i>	<i>FDR</i>
limma	0.5	0	1.0000000	0
T-TEST	0.5	0	1.0000000	0
limma	1.5	0.6370370	0.9971287	0.04140127
T-TEST	1.5	0.2560847	0.9993374	0.02419355
limma	3	0.9820106	0.9952512	0.04428424



T-TEST	3	0.9216931	0.9974600	0.02572707
--------	---	-----------	-----------	------------

Table 3 Metrics Table at different Effect sizes

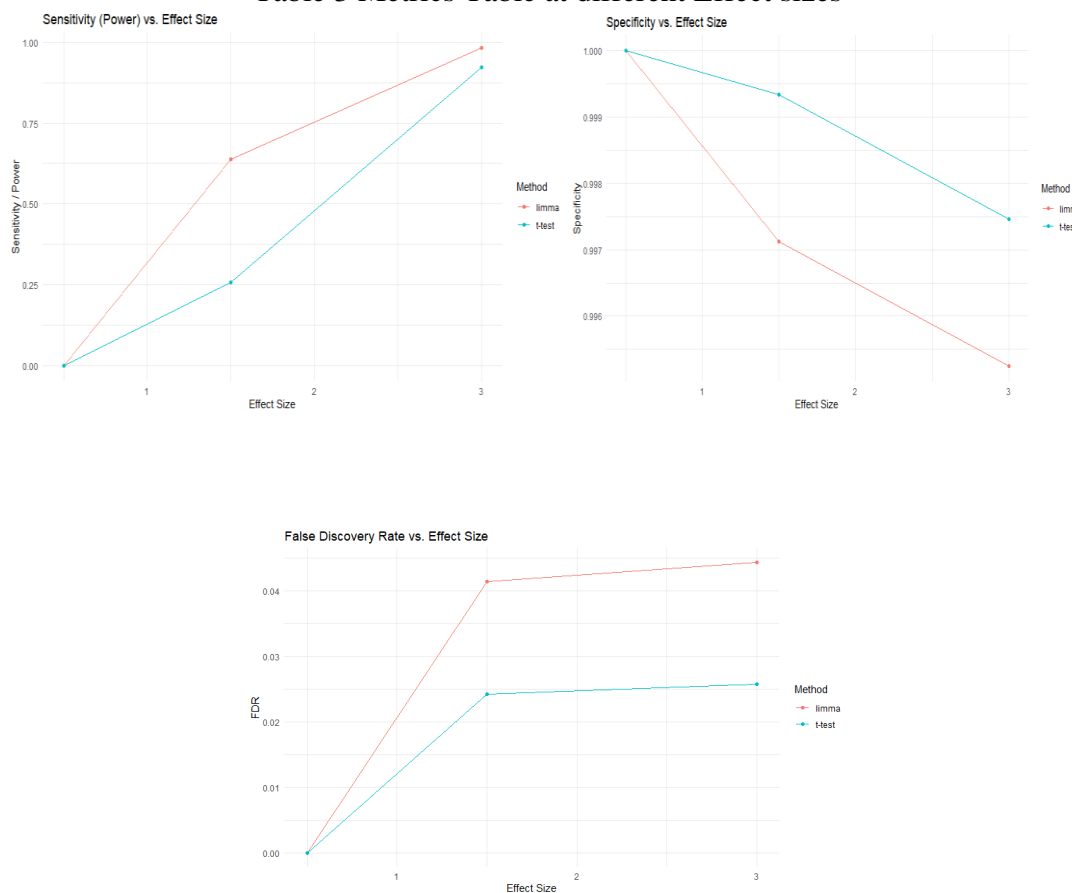
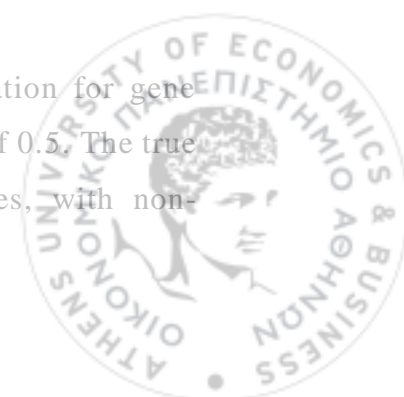


Figure 7 Metrics vs Effective Size

The mean difference vs standard deviation plots (Figure 8) offer important insights regarding the distribution of gene expression against their amount of variation. These plots visually illustrate how gene expression variability affects the detection capability of differentially expressed genes. The true DE genes are represented by red points, while the non-DE genes, by blue points. The significant genes identified are represented by filled circle symbols, while with open circle symbols the non-significant genes, providing a visual distinction between statistically validated results and background noise. The visualisation enhances data interpretability and showcases the strength of limma in identifying true DE genes.

The first plot displays the mean difference vs. standard deviation for gene expression data that was analyzed using limma at an effect size of 0.5. The true DE genes appear in red, blue points represent non-DE genes, with non-

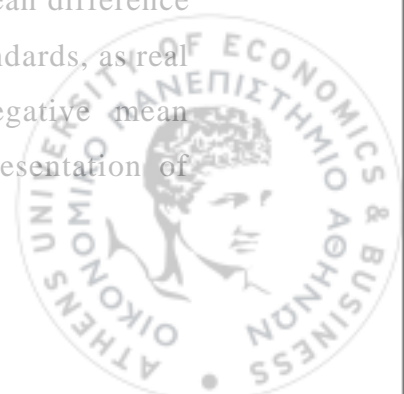


significant genes shown as open circles. As a result of the small effect size, there is very little separation between true DE and non-DE genes, which as it can be seen from the plot leads to a significant overlap between the two groups. This overlap suggests that limma fails to detect DE genes at this effect size, resulting in low sensitivity as no genes pass the significance threshold. Non-DE genes remain tightly clustered around a mean difference of zero, reinforcing high specificity since most non-DE genes are correctly classified as not significant. The spread in standard deviation reflects gene-specific variance, but the absence of significant genes confirms that at an effect size of 0.5, limma lacks power to distinguish DE genes from background noise.

With an effect size of 1.5, the separation between true DE and non-DE genes becomes more pronounced, with DE genes shifting further along the mean difference axis. An increased proportion of DE genes are now detected as significant (filled red circles), reflecting improved sensitivity compared to the smaller effect size of 0.5. The identified non-DE genes remain tightly clustered around zero, with minimal false positives (filled blue circles), indicating strong control over specificity. The plot demonstrates limma's ability to balance sensitivity and specificity at moderate effect sizes, detecting true differential gene expression with minimal false discovery rates.

At an effect size of 3, the separation between DE and non-DE genes is highly pronounced, with DE genes substantially separated along the mean difference axis. Nearly all true DE genes are detected as significant (filled red circles), demonstrating maximum sensitivity. Non-DE genes remain tightly centered near zero, with very few misclassified as significant (filled blue circles), reflecting strong specificity and low false positives. This represents an ideal scenario for limma, ensuring reliable detection of DE genes while maintaining variance control. The structured clustering of DE genes, even at large mean differences, highlights limma's robustness in detecting strong effects while maintaining statistical control.

Lastly, it should be noted that the visualizations pertaining to mean difference vs standard deviation below do not strictly adhere to realistic standards, as real depictions of DE genes would also include some with negative mean differences. However, these plots still provide a reliable representation of limma's performance.



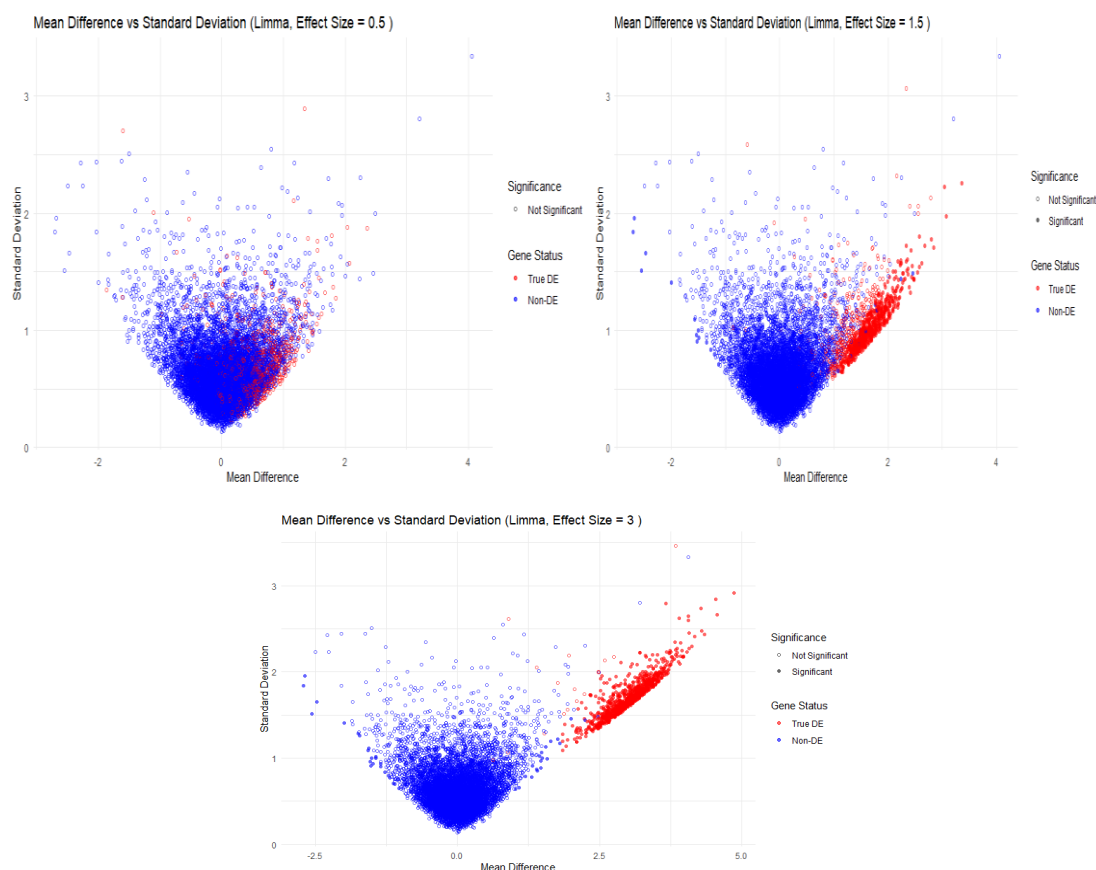


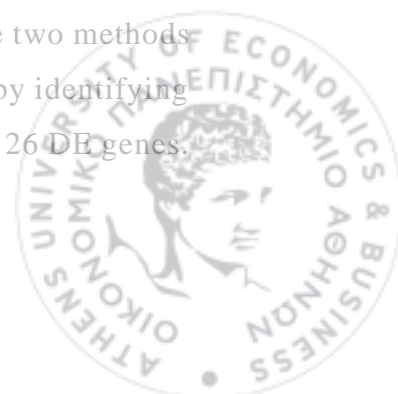
Figure 8 Mean difference versus standard deviation plots for various Effect size for Limma

### *IMPACT OF VARIANCE LEVELS ON DIFFERENTIAL EXPRESSION ANALYSIS*

This section evaluates the performance of limma and the t-test under varying levels of gene-specific variance: low, moderate, and high. These results are summarized and visualized using Venn diagrams (Figure 9).

At low variance, limma identified 980 DE genes compared to 959 detected by the t-test, out of the 945 truly DE genes. The results show that limma identified all the truly DE genes, while the t-test uniquely detected 17 DE genes.

At moderate variance, the sensitivity of both methods declined slightly, with limma detecting 990 DE genes compared to 954 for the t-test. The two methods overlapped on 908 DE genes, but limma continued to outperform by identifying an additional 35 true DE genes, while the t-test uniquely detected 26 DE genes.



Under high variance, the sensitivity of both methods further decreased. limma detected 993 DE genes, while the t-test identified 948. A strong overlap of 907 DE genes was observed between the two methods. limma outperformed again, detecting 28 additional DE genes compared to 14 uniquely identified by the t-test. However, 10 true DE genes remained undetected by either method, emphasizing the challenges posed by high variability in gene expression data. Overall, these results demonstrate that while both methods struggle with increased variance, limma consistently maintains higher sensitivity across all variance levels.

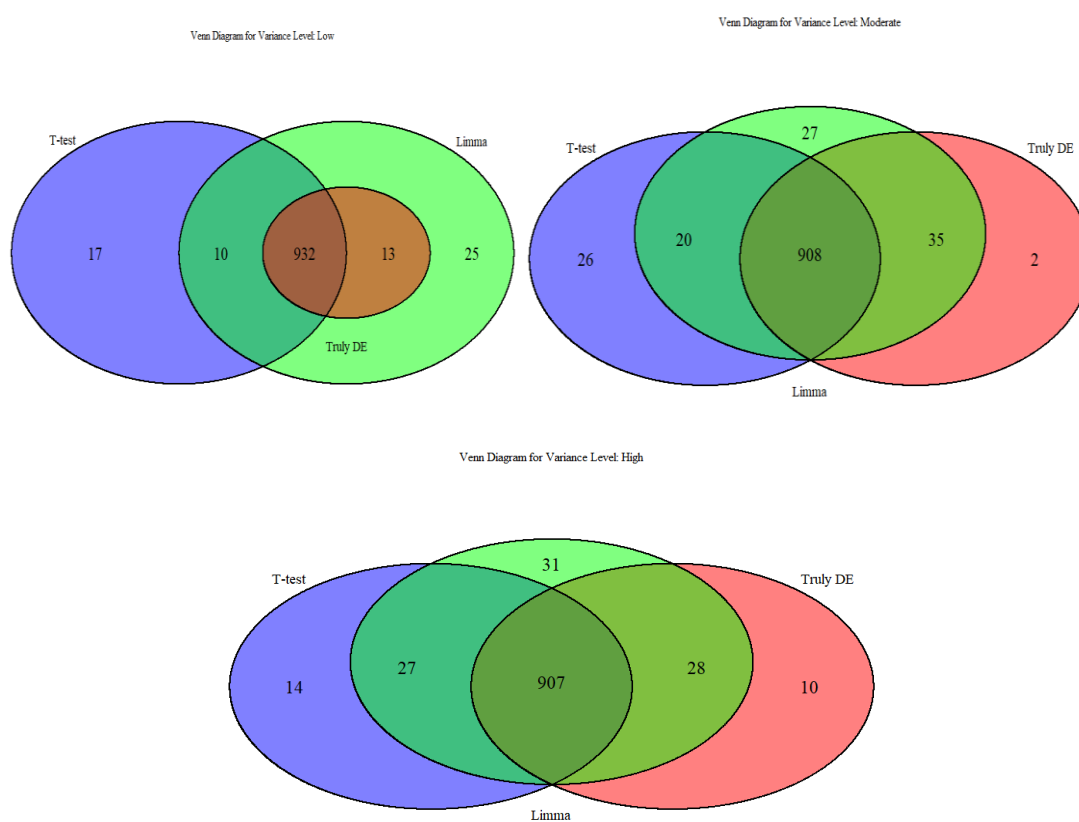


Figure 9 Number of Significant Genes Detected by Limma vs Ttest at different Variance Levels

The metrics for sensitivity, specificity, and *FDR*, as presented in Table 4 and Figure 10, further highlight the performance trends across varying levels of variance.

In terms of sensitivity, limma consistently outperformed the t-test across all conditions. At low variance, limma achieved an impressive sensitivity of 100.0%, compared to 98.6% for the t-test. The trend was maintained at moderate variance (99.8% for limma and 96.1% for the t-test) and high variance



(98.9% for limma and 96.0% for the t-test), underscoring limma’s ability to accurately detect true DE genes even under challenging conditions.

Specificity was very high for both methods at all levels of variance and was greater than 99.4% in all cases. The t-test had slightly greater specificity than limma in all conditions, with the largest difference at high variance (99.54% for the t-test and 99.35% for limma). However, it has to be noted that the differences in specificity are small and certainly do not overshadow limma's superior sensitivity.

False discovery rate (*FDR*) control showed marginally better performance by the t-test across all variance levels. For example, at low variance, the t-test achieved an *FDR* of 2.8%, compared to 3.6% for limma. Similar trends were observed at moderate variance (4.8% for the t-test vs. 4.7% for limma) and high variance (4.3% for the t-test vs. 5.8% for limma).

While the t-test demonstrated slightly better specificity and *FDR* control, limma’s superior sensitivity across all variance levels makes it a more robust choice for high-dimensional analyses. This is particularly evident in high-variance scenarios, where limma’s variance moderation plays a crucial role in accurately detecting subtle expression changes, reinforcing its reliability for differential expression studies.

<i>Method</i>	<i>Variance Level</i>	<i>Sensitivity</i>	<i>Specificity</i>	<i>FDR</i>
Limma	Low	1.0000000	0.9961347	0.03571429
T-TEST	Low	0.9862434	0.9970182	0.02815433
Limma	Moderate	0.9978836	0.9948095	0.04747475
T-TEST	Moderate	0.9608466	0.9949199	0.04821803
Limma	High	0.9894180	0.9935947	0.05840886
T-TEST	High	0.9597884	0.9954721	0.04324895

Table 4 Metrics Table a different Variance Levels



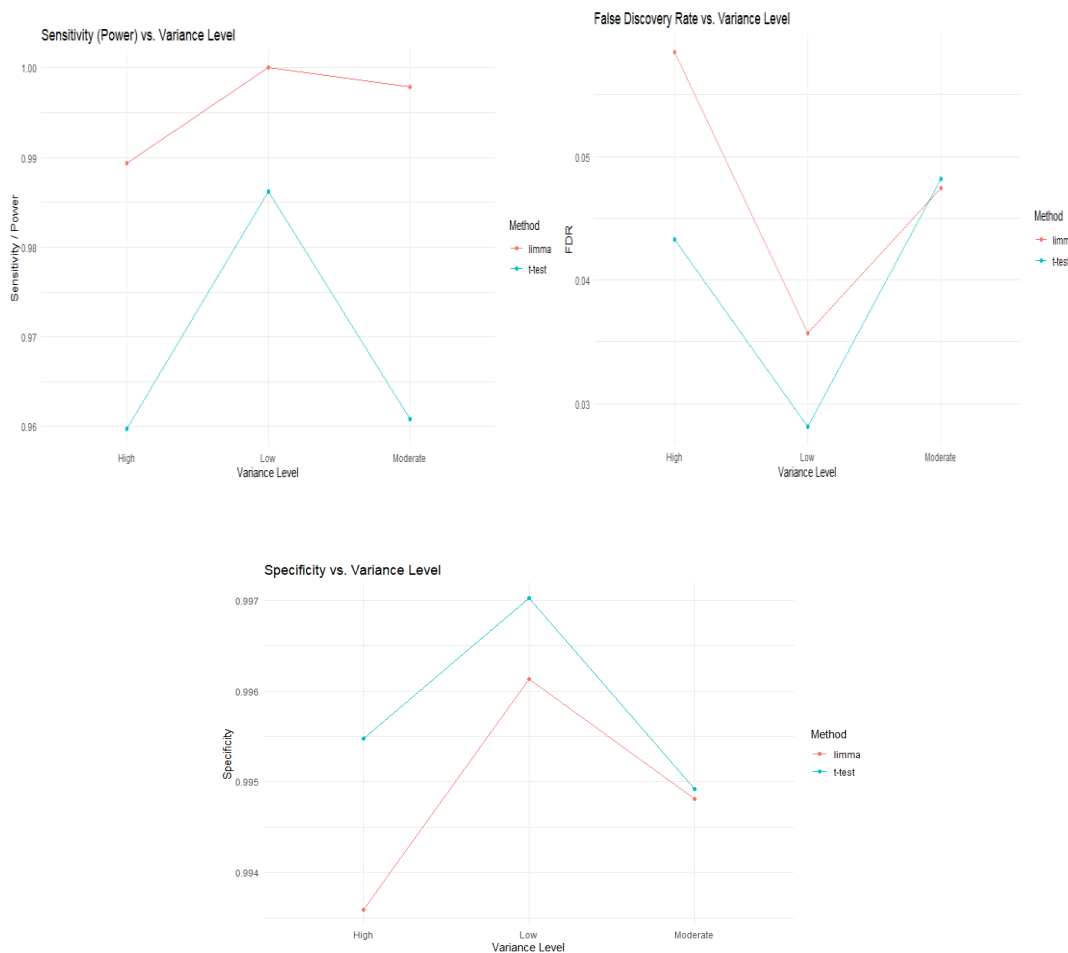
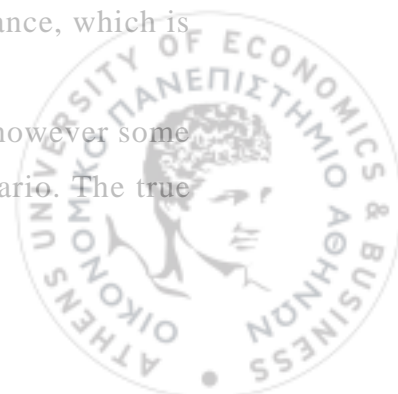


Figure 10 Metrics vs Variance Levels

To better understand the effect of variance on limma's performance, we compared the mean difference vs. standard deviation plots across low, moderate, and high variance levels (Figure 11).

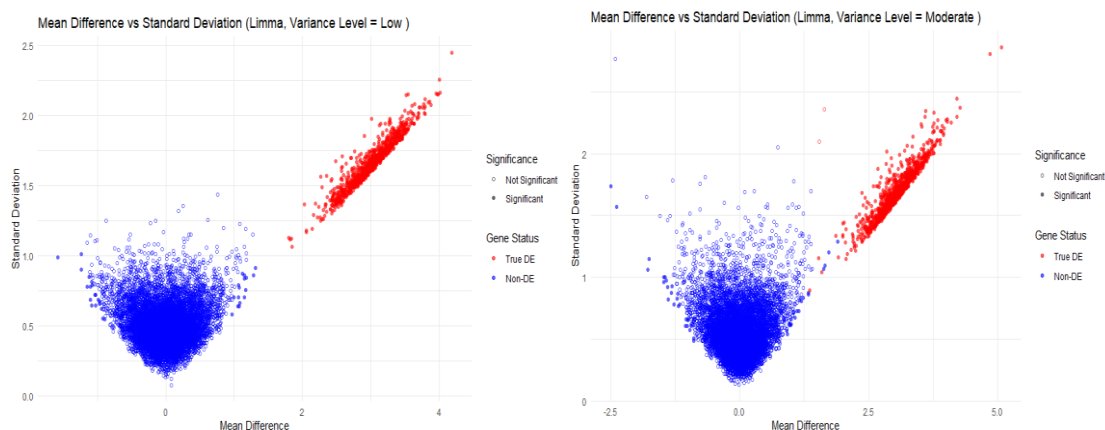
Under low variance conditions, limma performs optimally, since as it can be seen from the plot there is excellent separation between true DE genes and non-DE genes. True DE genes (red points) are tightly concentrated along the mean difference axis, with most of them correctly identified as significant (filled red circles). The standard deviation values for these genes are minimal. The plot also shows that non-DE genes (blue points) are tightly distributed around a mean difference of zero with very few false positives (filled blue circles). This shows that limma has high sensitivity and specificity at low variance, which is particularly useful in scenarios with minimal noise.

At moderate variance, the performance of limma is satisfactory, however some differences are observed in comparison to the low variance scenario. The true



DE genes are still separated from the Non-DE genes, as seen by their spread along the mean difference axis, but there is a slight increase in the spread along the standard deviation axis. Most of the DE genes are still identified as significant (filled red circles), but a small fraction of genes near the significance threshold (genes with lower mean differences) are not identified at all, which indicates a slight decrease in sensitivity. The non-DE genes remain tightly clustered around zero mean difference with minimal false positives (filled blue circles). This demonstrates that limma still performs robustly under moderate variability, retaining its ability to distinguish DE from non-DE genes with the false discovery rate kept at minimal levels.

In high-variance scenarios, the challenges of increased variability become more apparent. True DE genes exhibit greater dispersion along the standard deviation axis, and the proportion of significant DE genes (filled red circles) decreases, especially for genes with lower mean differences. However, DE genes with larger mean differences remain consistently identified as significant. Non-DE genes also display greater spread along both the mean difference and standard deviation axes, leading to a modest increase in false positives (filled blue circles). Despite this, limma maintains reasonable control over false discoveries, highlighting its robustness even under challenging conditions.



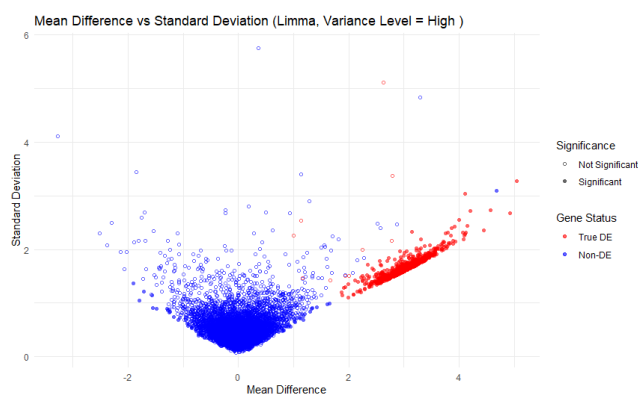


Figure 11 Mean versus variance plots for different Variance Levels for Limma

### *IMPACT OF SAMPLE SIZES ON DIFFERENTIAL EXPRESSION ANALYSIS*

In this analysis we assessed the performance of limma and the t-test in detecting differentially expressed (DE) genes across increasing sample sizes of 5, 10, and 20, with results summarized and visualized using Venn diagrams (Figure 12). At a sample size of 5, limma detected 971 DE genes out of the 945 truly DE genes, outperforming the t-test, which identified 894 DE genes. Both methods missed some truly DE genes, but limma identified 31 unique DE genes, compared to 11 uniquely detected by the t-test.

At a sample size of 10, both methods showed significant improvements in detection performance. Limma identified 998 DE genes, with only 1 truly DE gene missed, while the t-test detected 991 DE genes, missing 2 truly DE genes. The overlap of detected genes increased substantially, with 943 DE genes identified by both methods, demonstrating greater agreement as sample size grew.

For a sample size of 20, the performance of both methods converged. Limma detected 1,005 DE genes, while the t-test identified 1,003. The overlap between methods was very high, with 998 DE genes detected by both approaches, and 53 missed truly DE genes.



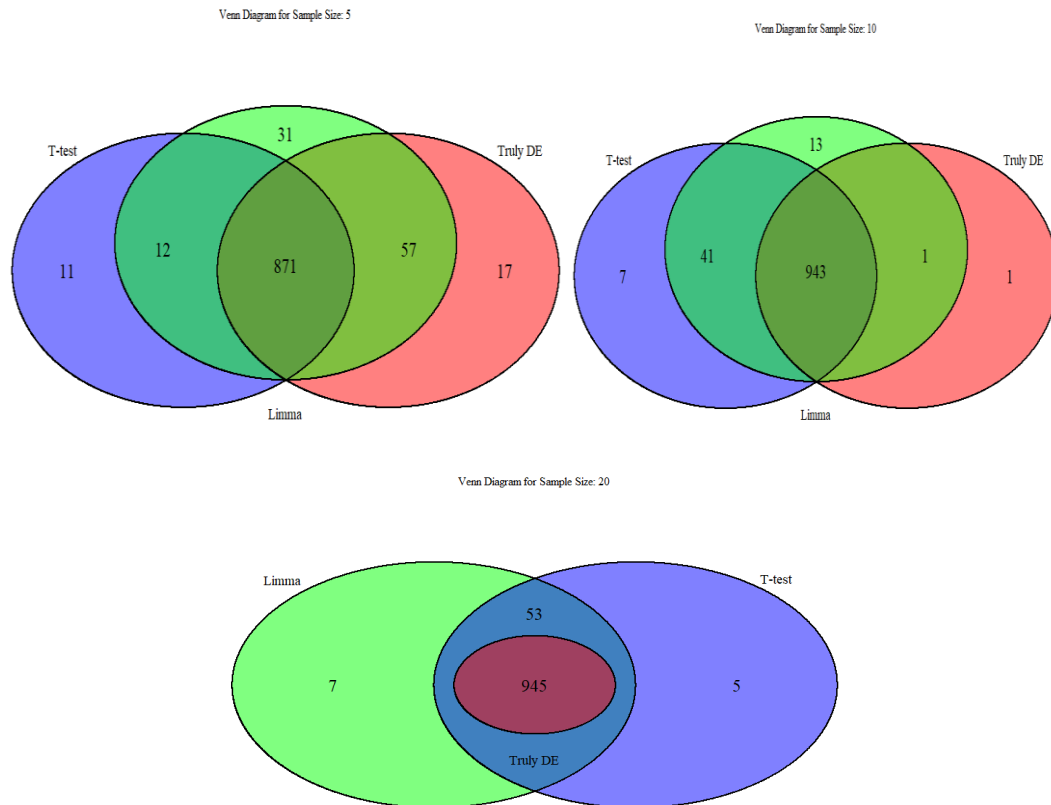


Figure 12 Number of Significant Genes Detected by Limma vs Ttest at different Sample Sizes

The metrics for sensitivity, specificity, and *FDR*, as presented in Table 5 and visualized in Figure 13, reveal the impact of sample size (5, 10, and 20) on the performance of limma and the t-test for detecting differentially expressed (DE) genes.

Sensitivity analysis reveals that limma consistently outperformed the t-test at smaller sample sizes, achieving 98.2% sensitivity at a sample size of 5 compared to the t-test’s 92.2%. As the sample size increased to 10 and 20, the sensitivity of both methods converged, with both approaches detecting nearly 100% of true DE genes, highlighting the minimal differences in performance as statistical power improves with larger samples.

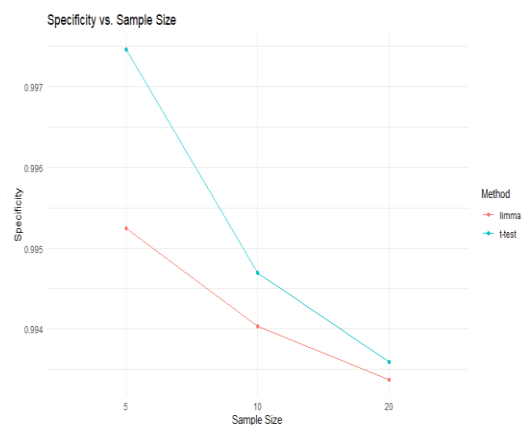
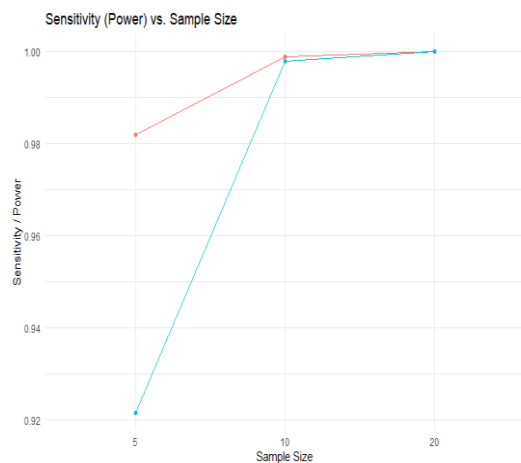
Specificity remained high (>99.3%) across all sample sizes for both methods, with the t-test maintaining slightly higher specificity at smaller sample sizes. For example, at a sample size of 5, the t-test achieved 99.7% specificity, while limma exhibited 99.5% specificity. However, as the sample size increased, the difference in specificity became negligible.



In terms of *FDR*, the t-test consistently demonstrated better control compared to limma across all sample sizes. At a sample size of 5, the t-test achieved an *FDR* of 2.6%, while limma's *FDR* was higher at 4.4%. However, as sample sizes increased, the *FDR* for both methods decreased and converged, with very small differences at a sample size of 20, where the *FDR* for limma was 5.97%, and the t-test achieved a slightly lower value of 5.78%. These results emphasize the advantages of increasing sample size, as both methods achieve nearly identical performance metrics under larger sample sizes. Limma however, exhibits stronger sensitivity at smaller sample sizes, making it more suitable for studies with limited replicates.

<i>Method</i>	<i>Sample Size</i>	<i>Sensitivity</i>	<i>Specificity</i>	<i>FDR</i>
limma	5	0.9820106	0.9952512	0.04428424
T-TEST	5	0.9216931	0.9974600	0.02572707
limma	10	0.9989418	0.9940364	0.05410822
T-TEST	10	0.9978836	0.9946991	0.04843592
limma	20	1.0000000	0.9933738	0.05970149
T-TEST	20	1.0000000	0.9935947	0.05782652

Table 5 Metrics Table a different Sample sizes



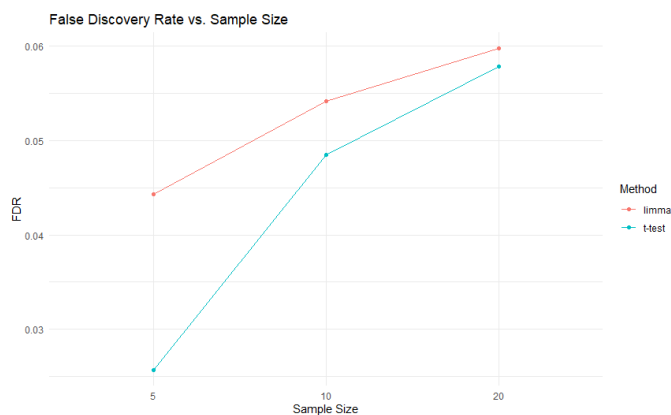


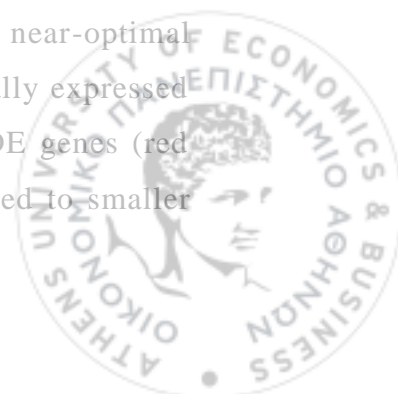
Figure 13 Metrics vs Sample Sizes

The following plots display the mean difference vs standard deviation for limma at varying sample sizes (Figure 14).

For small sample sizes ( $n = 5$  per group), Limma performs well, successfully detecting the majority of truly differentially expressed (DE) genes. This is evident in the plot, where most true DE genes (red points) are classified as significant (filled red circles), aligning with the high sensitivity observed in the results. However, the spread along the standard deviation axis remains wide, reflecting the increased variability in variance estimates due to the small number of observations per group. While some non-DE genes (blue points) are misclassified as significant (filled blue circles), indicating the presence of false positives, Limma maintains overall strong detection power.

At a sample size of 10 per group, Limma demonstrates near-perfect sensitivity, successfully detecting almost all truly differentially expressed (DE) genes. This is reflected in the plot, where all red points (true DE genes) are filled, indicating they have been classified as significant. Compared to  $n = 5$ , sensitivity has improved, reducing the number of undetected DE genes. However, specificity has slightly decreased, meaning a small increase in false positives. The distribution of DE genes remains tightly concentrated along the mean difference axis, while non-DE genes (blue points) continue clustering around zero mean difference.

For larger sample sizes ( $n = 20$  per group), Limma achieves near-optimal performance, successfully identifying almost all truly differentially expressed (DE) genes. This is evident in the plot, where nearly all true DE genes (red points) are classified as significant (filled red circles). Compared to smaller



sample sizes, the spread along the standard deviation axis is more compact, indicating improved variance estimation and reduced noise. Non-DE genes (blue points) remain tightly clustered near zero mean difference, with very few false positives (filled blue circles).



Figure 14 Mean versus variance plots for different Sample Sizes for Limma

### *IMPACT OF ALPHA LEVELS ON DIFFERENTIAL EXPRESSION ANALYSIS*

The comparison of the impact of various alpha levels (0.01, 0.05, and 0.1) on the detection of differentially expressed (DE) genes by limma and t-test revealed the following (Figure 15):

At the most conservative alpha level of 0.01, both methods performed similarly, with limma detecting 949 DE genes and the t-test detecting 946, with great overlap since 941 genes were detected by both methods. The number of uniquely detected genes was minimal, indicating strong agreement between the two approaches under stringent conditions.

As the alpha level increased to 0.05, the number of detected DE genes rose for both methods. limma identified 980 DE genes, while the t-test detected 975,



with an overlap of 943 genes. This demonstrates the improved detection power as the threshold for significance becomes less restrictive, while maintaining strong agreement between the two methods.

At the most lenient alpha level of 0.1, the detection of DE genes further increased, with limma identifying 1,049 and the t-test detecting 1,036. The overlap between methods remained high, with 944 genes identified as DE by both approaches. However, the more lenient threshold also resulted in an increase in uniquely detected genes by each method, reflecting their slight variations in performance and how alpha levels contribute to the sensitivity and specificity trade-off in detecting DE.

These results highlight the importance of selecting an optimal alpha level according to the study-specific goals and the tolerance of false discoveries in microarray studies.

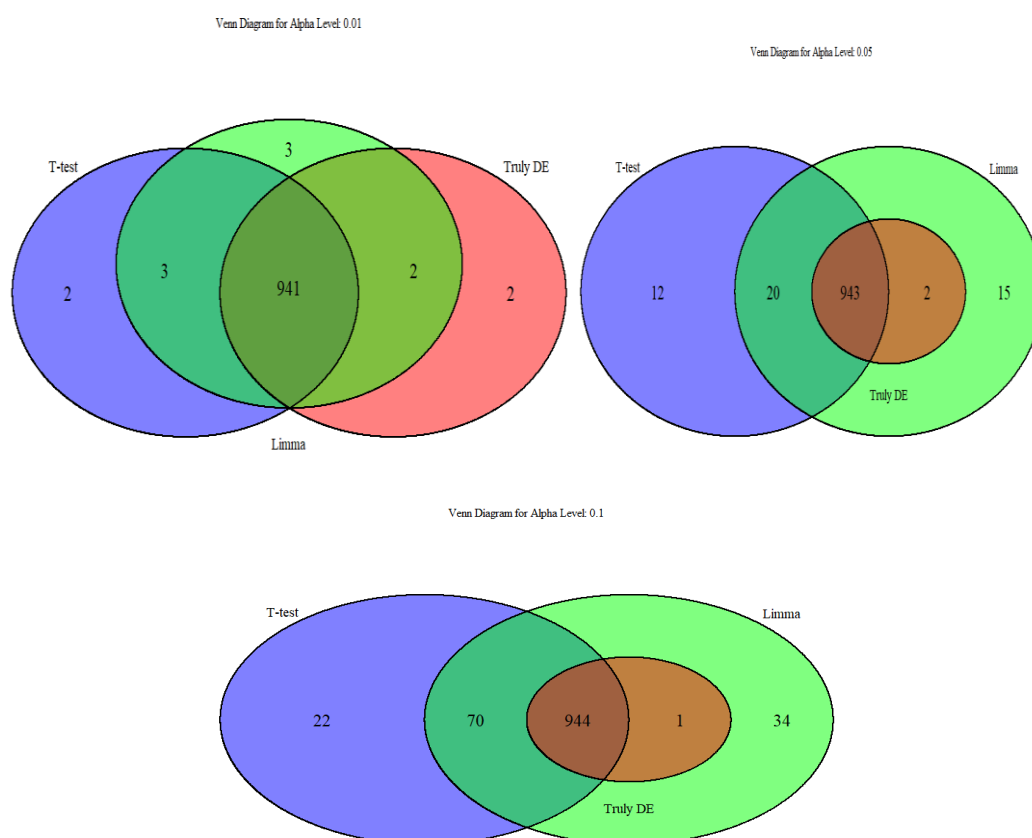


Figure 15 Limma Number of Significant Genes Detected by Limma vs Ttest at different  $\alpha$  levels



The metrics for sensitivity, specificity, and false discovery rate (*FDR*) across varying alpha levels (0.01, 0.05, and 0.10) are summarized in Table 6 and Figure 16.

Sensitivity results highlighted limma’s superior performance at all alpha levels. At the strictest alpha level of 0.01, limma achieved a sensitivity of 99.8% compared to 99.6% for the t-test. As the alpha threshold relaxed to 0.10, limma achieved perfect sensitivity (100%), while the t-test reached 99.9%.

Both methods had very high specificity at all alpha, though a gradual decline was observed as alpha increased, as might be anticipated from the increase in type I errors. At an alpha level of 0.01, specificity was 99.93% for limma and 99.94% for the t-test. However, at an alpha level of 0.10, specificity dropped to 98.85% for limma and 98.98% for the t-test, demonstrating the loss of sensitivity at the expense of specificity as the criterion for alpha was relaxed.

*FDR* analysis revealed that both methods exhibited increasing *FDR* values as alpha levels increased. At the strict alpha level of 0.01, limma had an *FDR* of 0.63%, slightly higher than the t-test’s 0.53%. By alpha 0.10, *FDR* rose to 9.91% for limma and 8.88% for the t-test. Despite the t-test consistently showing marginally lower *FDR* values, the differences were minimal. These results underscore the importance of balancing sensitivity, specificity, and *FDR* in selecting an appropriate alpha threshold for microarray data analysis, with limma generally demonstrating better performance in detecting true DE genes.

<i>Method</i>	<i>Alpha Level</i>	<i>Sensitivity</i>	<i>Specificity</i>	<i>FDR</i>
limma	0.01	0.9978836	0.9993374	0.006322445
T-TEST	0.01	0.9957672	0.9994478	0.005285412
limma	0.05	1.0000000	0.9961347	0.035714286
T-TEST	0.05	0.9978836	0.9964660	0.032820513
limma	0.1	1.0000000	0.9885146	0.099142040
T-TEST	0.1	0.9989418	0.9898399	0.088803089

Table 6 Metrics Table a different  $\alpha$  levels



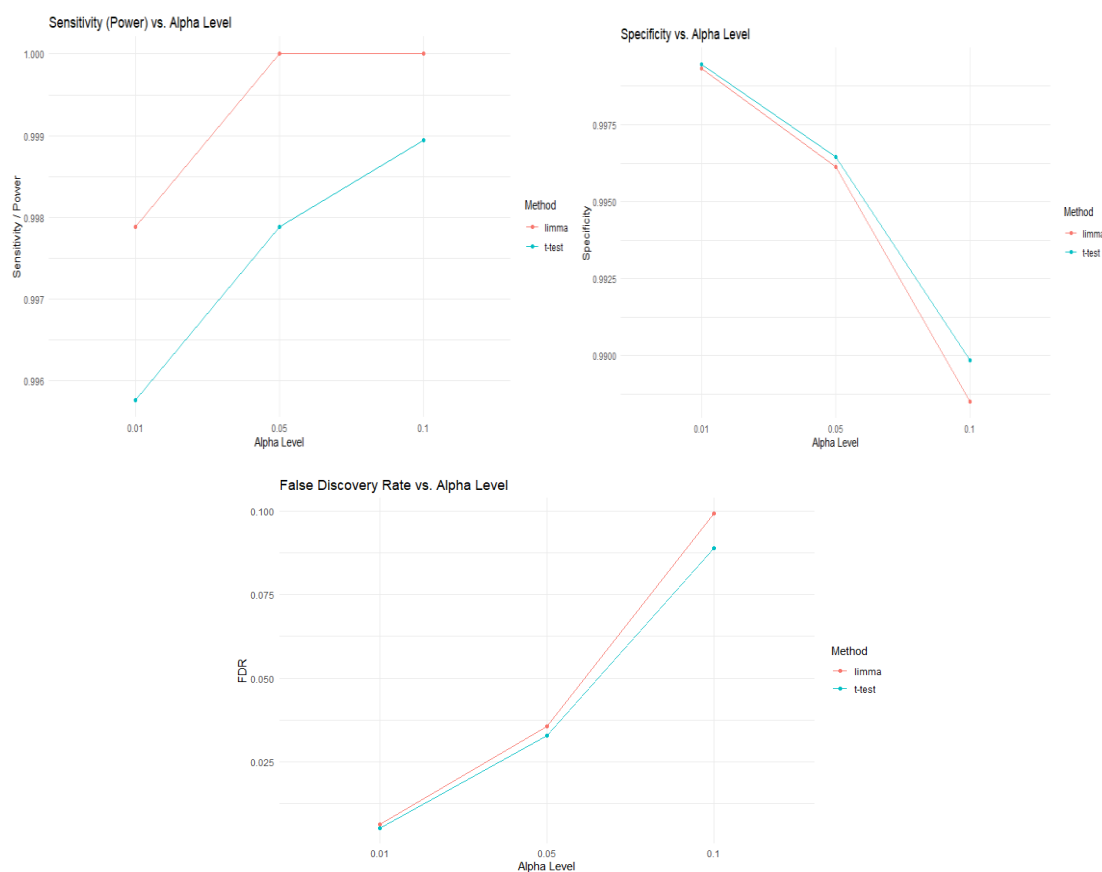
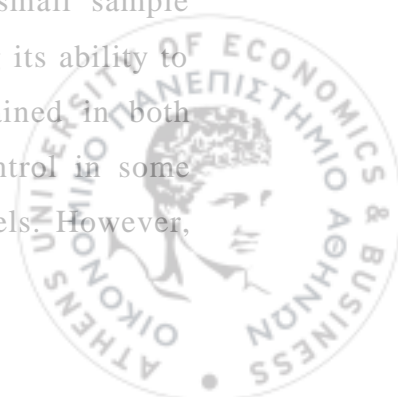


Figure 16 Metrics vs  $\alpha$  levels

The mean difference vs. standard deviation plots for varying alpha levels (0.01, 0.05, and 0.1) are not strictly necessary, as they mostly illustrate the expected increase in the number of significant genes as the alpha threshold becomes less stringent. While the plots show more filled circles (significant genes) at higher alpha levels, the overall distribution of points and clustering of DE and non-DE genes remain unchanged.

## Conclusion

Across all comparisons, including varying effect sizes, variance levels, alpha thresholds, and sample sizes, limma consistently outperformed the t-test in detecting differentially expressed (DE) genes under a wide range of conditions. Limma demonstrated superior sensitivity, particularly under small sample sizes, low effect sizes, and varying levels of variance, reflecting its ability to reliably identify true DE genes. High specificity was maintained in both methods overall, with t-test providing slightly better *FDR* control in some scenarios, especially at larger sample sizes or higher alpha levels. However,



limma's robust empirical Bayes framework provided improved stability and power in challenging scenarios, for instance, small sample sizes or low alpha thresholds for which the t-test often underperformed. These results highlight the flexibility and strength of limma as the primary method for differential expression analysis, particularly in complicated experimental setups.

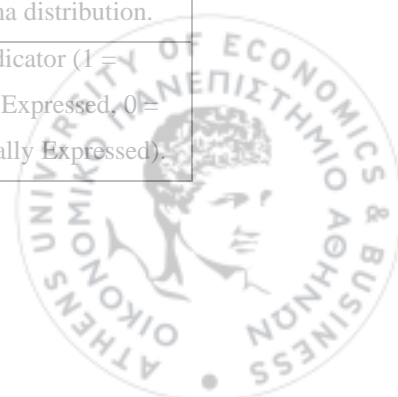
### 2.5.2 Part 2: Three Group Comparison (Multiple Contrasts)

This part of the analysis focuses on differential expression analysis across three experimental groups: A control group and two treatment groups, Treatment A and Treatment B. The aim was to determine how two statistical methods, limma and the ANOVA F-test, perform in identifying the true DE genes in a multi-group experimental design. This assessment focused on global testing across groups, in terms of sensitivity and specificity as well as false discovery rates under simulated microarray experimental conditions.

The simulation framework designed for this analysis comprised a dataset of 10,000 simulated genes, with 1,000 (10%) of them identified as the true DE. Gene-specific variances were drawn from an inverse gamma distribution with a shape parameter of 2 and a scale parameter of 3, representing high variability among genes often observed in biological datasets. Expression levels for the control group were generated with a mean of 0, while the means for DE genes were shifted by 1 in Treatment A and 1.4 in Treatment B to simulate realistic effect sizes. Small sample sizes of four samples per group were used, resulting in 12 samples in total, to test the robustness of these methods under challenging conditions.

A brief description of the data's structure can be seen below (Table 7).

<i>Column Name</i>	<i>Description</i>
Gene_ID	Unique identifier for each gene (e.g., Gene_1, Gene_2, ..., Gene_n).
Variance	Gene-specific variance, drawn from a Gamma distribution.
Is_DE	Binary indicator (1 = Differentially Expressed, 0 = Not Differentially Expressed).

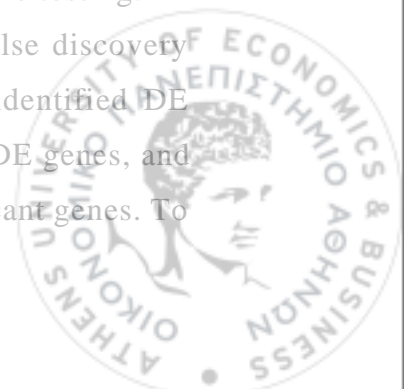


Control_1	Simulated expression value for Control Group, Sample 1.
Control_2	Simulated expression value for Control Group, Sample 2.
Control_3	Simulated expression value for Control Group, Sample 3.
Control_4	Simulated expression value for Control Group, Sample 4.
TreatmentA_1	Simulated expression value for Treatment Group A, Sample 1.
TreatmentA_2	Simulated expression value for Treatment Group A, Sample 2.
TreatmentA_3	Simulated expression value for Treatment Group A, Sample 3.
TreatmentA_4	Simulated expression value for Treatment Group A, Sample 4.
TreatmentB_1	Simulated expression value for Treatment Group B, Sample 1.
TreatmentB_2	Simulated expression value for Treatment Group B, Sample 2.
TreatmentB_3	Simulated expression value for Treatment Group B, Sample 3.
TreatmentB_4	Simulated expression value for Treatment Group B, Sample 4.

Table 7 Data Structure for three group comparison

The hypothesis testing framework was built around the null hypothesis that no differences exist in mean gene expression across the three groups, against the alternative hypothesis that at least one group differs significantly. Limma, employing an empirical Bayes approach, performed a global F-test leveraging moderated statistics to enhance performance with small sample sizes. The classical ANOVA F-test, which does not incorporate such variance moderation, was used as a comparative method. For both approaches, p-values were adjusted using the false discovery rate (*FDR*) method to control for multiple testing.

Performance metrics included sensitivity, specificity, and the false discovery rate (*FDR*). Sensitivity quantified the proportion of correctly identified DE genes, specificity measured the ability to correctly classify non-DE genes, and *FDR* estimated the proportion of false positives among all significant genes. To



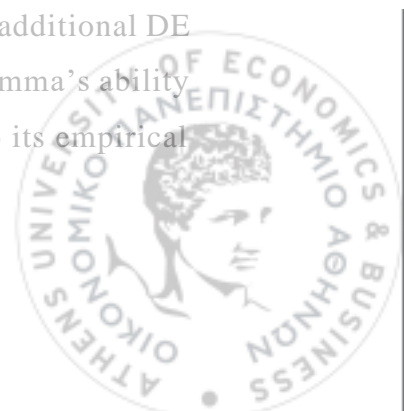
validate and visualize the results, Venn diagrams were used to highlight the overlap between genes identified as significant by limma and the ANOVA F-test, alongside the true DE gene set. Additionally, ROC curve visualizations and AUC calculations were presented to assess the discriminative power of each method in distinguishing DE from non-DE genes.

## Results

The global test for three groups captures the differences in performance between the limma and ANOVA F-test methods in identifying truly differentially expressed (DE) genes. 945 out of 10,000 simulated genes are designated as truly DE (Table 8).

The limma method demonstrated strong performance (Tables 8, 9), detecting 794 genes as significant, with 787 of these overlapping with the truly DE genes. This corresponds to a sensitivity of 83.3%, highlighting limma's ability to correctly identify the majority of DE genes. Its specificity of 99.92% reflects effective control over false positives relative to true negatives, while its *FDR* of 0.8% indicates a small proportion of false positives among its significant findings. In contrast, the ANOVA F-test detected 168 genes as significant, of which only 167 overlapped with the truly DE genes. This resulted in a much lower sensitivity of 17.7%, suggesting that the ANOVA F-test missed a significant number of truly DE genes. However, it achieved a slightly higher specificity of 99.98% compared to limma, demonstrating strong control over false positives. The ANOVA F-test's *FDR* was lower than that of limma, at 0.6%, indicating a slightly lower proportion of false positives among its detected genes.

The Venn diagram (Figure 17) further illustrates these findings. The intersection of all three sets—True\_DE, limma, and ANOVA F-test—consists of 167 genes. These represent the genes consistently detected by both methods that are also truly DE, highlighting areas of agreement between the methods. However, limma demonstrated a clear advantage, identifying 620 additional DE genes that were missed by the ANOVA F-test. This underscores limma's ability to detect a broader range of DE genes, which can be attributed to its empirical



Bayes approach that moderates variance estimates and increases statistical power in small-sample and high-variance scenarios.

Interestingly, 158 truly DE genes were not detected by either method. This indicates potential limitations in sensitivity, which could be due to the high gene variance or the small sample sizes used in this analysis. Both methods also detected non-DE genes, contributing to their respective false discovery rates (*FDR*). Limma identified 6 non-DE genes, while the ANOVA F-test detected 1 non-DE gene. The relatively low false-positive detections by both methods demonstrate effective control of type I errors.

<i>Method</i>	<i>Truly DE</i>
TRULY_DE	945
LIMMA DETECTED	794
F-TEST DETECTED	168

Table 8 Summary Table Of DE Gene Counts

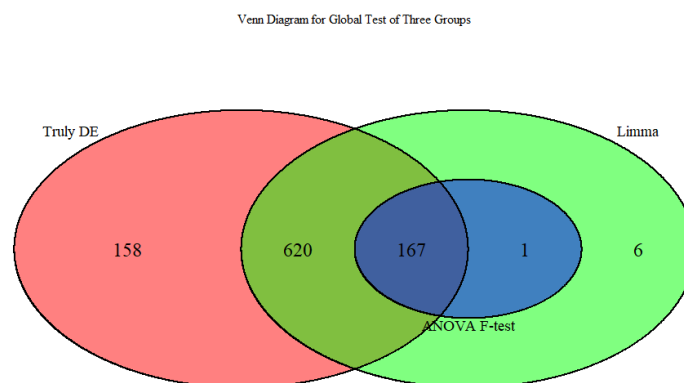
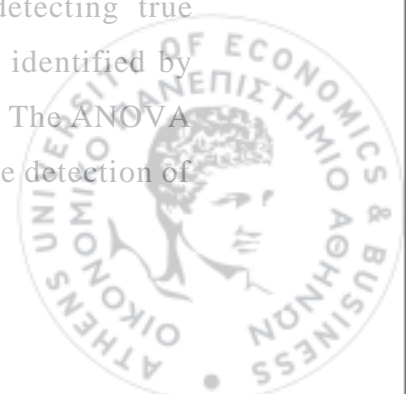


Figure 17 Number of Significant Genes Detected by Limma vs Ftest

<i>Method</i>	<i>Sensitivity</i>	<i>Specificity</i>	<i>FDR</i>
limma	0.8328042	0.9992269	0.008816121
F-TEST	0.1767196	0.9998896	0.005952381

Table 9 Metrics Table

The Venn plot demonstrates that limma is more robust in detecting true differentially expressed (DE) genes, since it captured DE genes identified by both methods, as well as a substantial number of additional genes. The ANOVA F-test, being more conservative, contributes only marginally to the detection of



truly DE genes, as indicated by its small intersection of just one gene. The results also show that both methods fail to identify some truly DE genes, highlighting the effect of high variance and small sample sizes on detection performance.

The ROC curve (Figure 18) compares the performance of limma and the ANOVA F-test in detecting DE genes by plotting sensitivity against specificity across different thresholds. limma achieves an AUC of 0.993, indicating near-perfect discriminative power, while the ANOVA F-test reaches an AUC of 0.957, showing slightly lower accuracy in identifying true positives.

limma's curve, shown in blue, consistently stays closer to the top-left corner of the graph compared to the F-test, indicating higher sensitivity and specificity. This is due to limma's empirical Bayes method, which stabilizes variance estimates and enhances statistical power, particularly in cases with small sample sizes or high variance. By reducing the effects of noise, this approach improves the detection of differentially expressed genes.

On the other hand, the ANOVA F-test, shown in red, performs worse than limma because it relies on individual variance estimates without shrinkage. This limitation increases the likelihood of false positives and reduces sensitivity in difficult experimental conditions.

Therefore, these results support the recommendation of limma for microarray data analysis, where precise identification of DE genes is paramount, particularly in complex or resource-limited experimental setups.

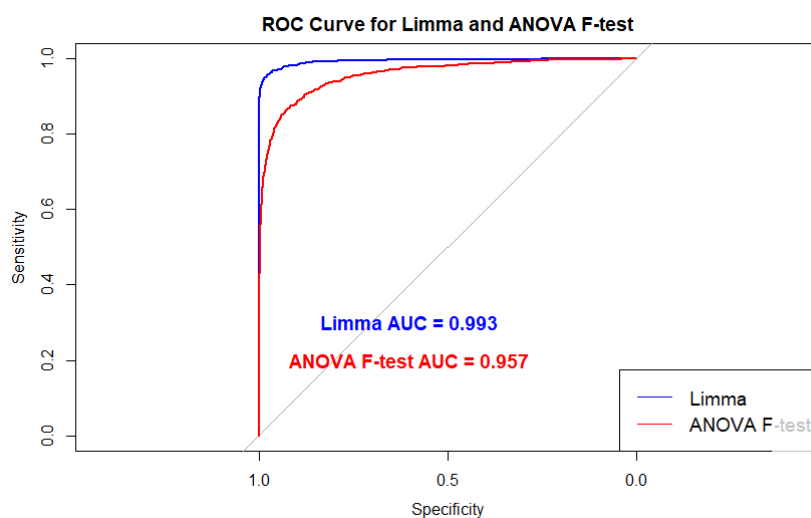


Figure 18 Roc Curve



The mean difference vs. standard deviation plots for the global comparison using limma provide a detailed visual representation of the separation between groups in differential expression analysis. These plots show pairwise comparisons, including Control vs. Treatment A, Control vs. Treatment B, and Treatment A vs. Treatment B, and explore the consistency and performance of each method in detecting significant genes.

The mean difference vs. standard deviation plot (Figure 19) for the Control vs. Treatment A comparison illustrates limma's performance under challenging conditions of small sample size and high variability. True DE genes (red points) are largely distributed along the positive mean difference axis, with many showing small to moderate shifts in expression. Limma successfully identifies a significant proportion of these DE genes as significant (filled red circles), particularly those with larger mean differences and lower standard deviations, demonstrating its sensitivity in detecting biologically meaningful changes. However, a notable fraction of true DE genes with smaller mean differences or higher variability remains undetected (open red circles).

Non-DE genes (blue points) are tightly clustered near zero mean difference, with most classified as not significant (open blue circles), reflecting excellent control over false positives. Only a small subset of non-DE genes is incorrectly identified as significant (filled blue circles), demonstrating limma's ability to maintain specificity despite the noise introduced by small sample size and high variability. The spread in the standard deviation axis underscores the variability inherent in this scenario, particularly impacting genes with higher variability, which are less likely to be classified as significant.



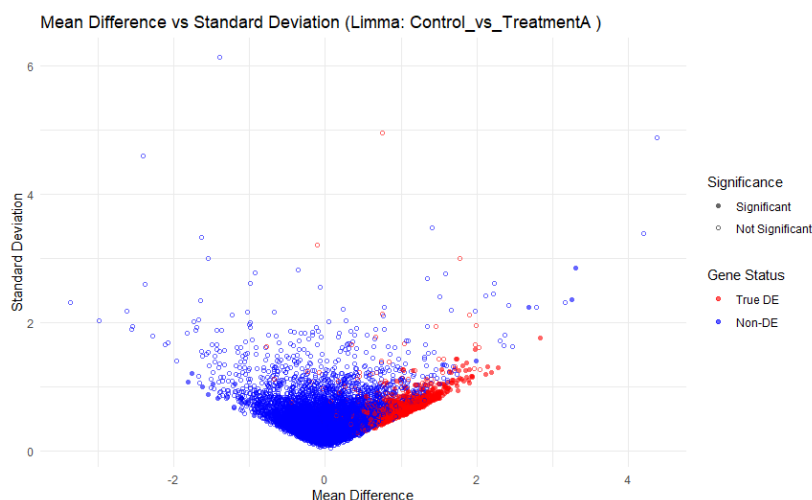


Figure 19 Mean difference vs Variance for Control vs Treatment A

The mean difference vs. standard deviation plot (Figure 20) for the Control vs. Treatment B comparison highlights limma's performance, with an increased mean shift ( $effect\ size = 1.4$ ) for Treatment B compared to Treatment A. True DE genes (red points) are more spread out along the positive mean difference axis, reflecting the larger effect size, which improves their separation from non-DE genes. A higher proportion of true DE genes are classified as significant (filled red circles), particularly those with larger mean differences, demonstrating limma's enhanced sensitivity when the effect size increases. However, true DE genes with smaller mean differences or higher standard deviations remain less likely to be detected, demonstrating the continued influence of variability on detection accuracy.

Non-DE genes (blue points) are tightly centered near a mean difference of zero, with most classified as not significant (open blue circles). This indicates excellent specificity and robust control over false positives, even under the constraints of small sample size and high variability. A minimal number of non-DE genes are misclassified as significant (filled blue circles). The variability inherent in the dataset is reflected in the wide spread of standard deviations, with true DE genes showing increased dispersion.



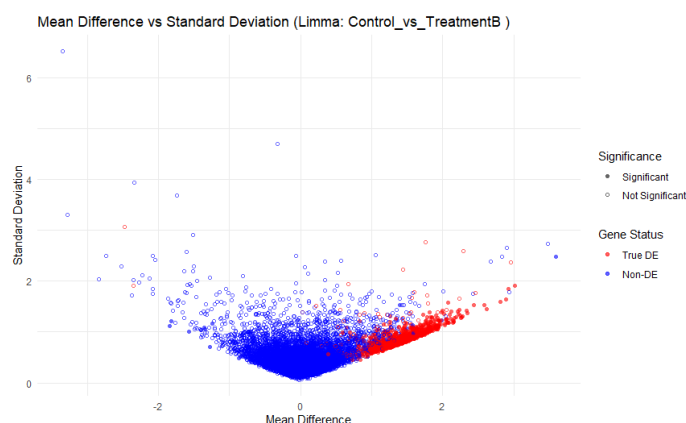


Figure 20 Mean difference vs Variance for Control vs Treatment B

The mean difference vs. standard deviation plot for the comparison between Treatment A and Treatment B illustrates the challenges of differential expression analysis when effect sizes are small, variance is high, and sample sizes are limited (Figure 21). True DE genes show a narrow spread along the mean difference axis because their expression levels change less between Treatment A and Treatment B compared to Control vs. Treatment comparisons. The number of true DE genes identified as statistically significant remains low (filled red circles), as Limma's performance is affected under conditions of small effect sizes and high noise. Many true DE genes, particularly those with small mean differences or high variability, remain undetected.

Non-DE genes (blue points) are positioned near zero on the mean difference axis, reflecting the absence of expression changes between Treatment A and Treatment B. Most non-DE genes are correctly classified as not significant (open blue circles), showing that Limma effectively controls false positives. A small number of non-DE genes are incorrectly classified as significant (filled blue circles). The overlap between true DE and non-DE genes in regions with small mean differences makes classification more difficult, especially for genes with higher variability.

The widespread along the standard deviation axis, which reflects the dataset's variability, poses additional challenges in detecting true DE genes, particularly those with smaller mean differences. This variability reduces the distinction between DE and non-DE genes, making it harder to detect subtle differences in gene expression between the two treatments.



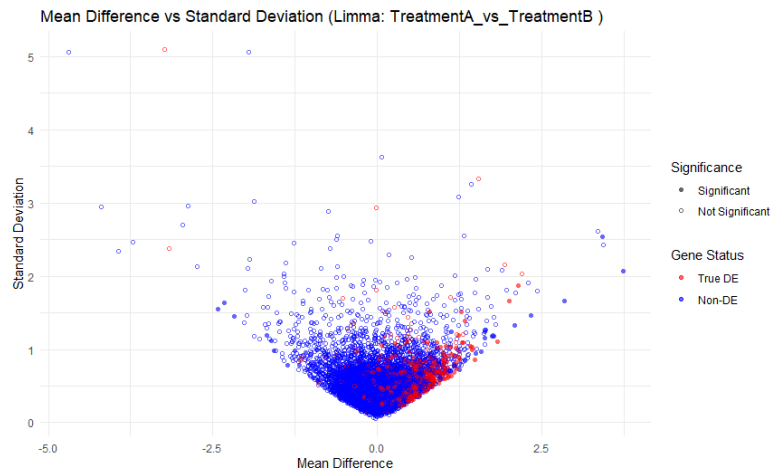


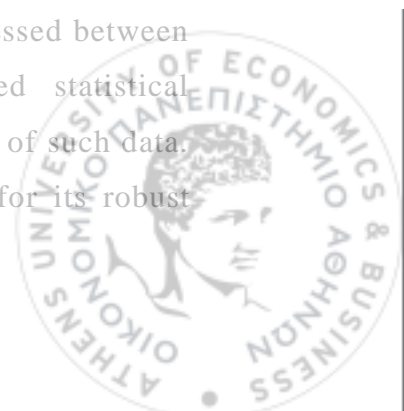
Figure 21 Mean difference vs Variance for Treatment A vs Treatment B

### Conclusion

The comparison of limma and the ANOVA F-test for three-group differential expression analysis demonstrates that limma is consistently superior in identifying truly differentially expressed (DE) genes. Limma detected a substantially higher number of DE genes (794 vs. 168), with significantly greater sensitivity (83.3% vs. 17.7%) while maintaining a low false discovery rate (*FDR*: 0.8% vs. 0.6%) and high specificity (>99%). The overlap between the two methods showed that limma uniquely detected 620 DE genes, emphasizing its ability to capture subtle group-specific effects. The ROC analysis further confirmed limma's robust performance, with a higher AUC compared to the F-test. These findings highlight limma's empirical Bayes framework as a clear advantage for high-dimensional, small-sample datasets, making it the preferred method for three-group comparisons in microarray studies.

### 2.6 Application 2: Differential Expression Analysis on Real Data

This exercise focuses on the application of statistical methods for differential expression analysis to real-world microarray data, specifically the leukemia dataset. The primary aim is to identify genes differentially expressed between leukemia subtypes and normal controls by using advanced statistical frameworks to handle the high dimensionality and heterogeneity of such data. The exercise employs the limma package, widely recognized for its robust



empirical Bayes variance moderation, to perform global and pairwise differential expression analysis.

In the first half of this exercise, a global F-test is conducted to determine genes that exhibit significant expression variability across five groups: Acute Lymphoblastic Leukemia (ALL), Acute Myeloid Leukemia (AML), Chronic Lymphocytic Leukemia (CLL), Chronic Myeloid Leukemia (CML), and healthy controls (No Leukemia, NoL). This analysis highlights genes that are globally significant across all groups. The findings are presented using graphical representations such as volcano plots and MA plots, providing details on the distribution of the significant genes.

The second part proceeds to pairwise comparisons by comparing NoL to each leukemia subtype (e.g., ALL vs. NoL, AML vs. NoL) to identify subtype-specific patterns of gene expression. This allows for better understanding of the molecular differences between leukemia subtypes and controls. The analysis includes the generation of volcano plots for each comparison to visualize the distribution of adjusted p-values and log-fold changes and mean difference vs. standard deviation plots to assess variability and significance.

### 2.6.1 Global Tests to identify significant genes

The global test for identifying significant genes aims to detect genes that are differentially expressed across multiple groups simultaneously, providing us with information on the molecular differences between leukemia subtypes and healthy controls.

#### *DATA PREPARATION AND MODEL FITTING*

The analysis began with data preparation, where the expression matrix was created. Each gene's expression levels across the relevant samples were extracted to focus on the subtypes of interest. The group labels, corresponding to the five groups, were encoded as categorical variables to allow accurate modeling of group-specific effects. This preprocessing step ensured that the subsequent analysis captured meaningful differences in gene expression across the leukemia subtypes and controls.

A design matrix was constructed to model the group-specific effects across all subtypes. Using the limma package, a linear model was fitted to the expression



data for each gene. This modeling step allowed the estimation of group-specific coefficients, facilitating the assessment of expression differences between groups.

#### *GLOBAL F-TEST AND STATISTICAL ADJUSTMENT*

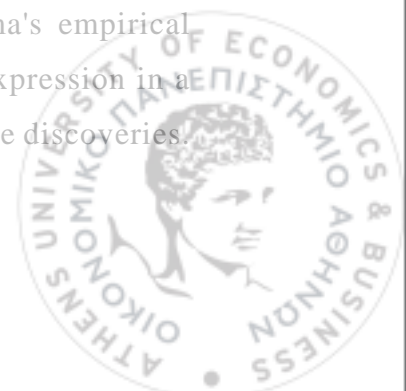
To identify significant genes, a global F-test was performed. This test evaluates the null hypothesis that all group means are equal for each gene, effectively identifying genes with expression patterns that differ significantly across the groups. The false discovery rate (*FDR*) was controlled by the Benjamini-Hochberg procedure, which adjusted p-values to account for multiple testing. This adjustment ensured the validity of the findings by minimizing false positives without compromising the statistical power of identifying significant differences.

#### *VISUALIZATION AND INTERPRETATION*

Visualization played a key role in the interpretation of the results. A volcano plot was generated to depict the relationship between average gene expression and adjusted p-values, with significant genes highlighted. The plot provided an intuitive way of identifying genes with large expression differences between the groups. Additionally, an MA plot was used to illustrate the relationship between mean expression levels and log-fold changes, offering further insights into the distribution and variability of significant genes.

#### *EXPLORING FDR THRESHOLDS*

To provide a broader perspective on gene significance, the proportion of significant genes was assessed across varying *FDR* thresholds, ranging from 0.01 to 0.1. This analysis demonstrated how the number of significant genes changes with stricter or more lenient thresholds, highlighting the robustness of the findings at different levels of statistical stringency. Together, these results emphasized the ability of the global test, combined with limma's empirical Bayes methods, to identify meaningful patterns of differential expression in a high-dimensional dataset while controlling for variability and false discoveries.



## Results

The global F-test identified 12,141 significant genes, indicating widespread differential expression across the leukemia subtypes and the healthy control group. The volcano plot (Figure 22) visualizes the results, displaying the relationship between average gene expression (x-axis) and the significance of expression differences ( $-\log_{10}$  adjusted p-values, y-axis). Significant genes are shown in grey, while non-significant genes are shown in red.

The graph demonstrates that most genes with varying average expression levels exhibit considerable differential expression. The dashed blue horizontal line represents the *FDR* threshold (adjusted *p*-value < 0.05), and genes above it are considered significant. Most significant genes are clustered along the lower expression range, reflecting the high-dimensional nature of the dataset and the sensitivity of the global F-test to detect differences across groups. However, some genes with higher average expression levels are also identified as significant, suggesting robust performance of the test across a wide range of expression intensities.

This high number of significant genes underscores the molecular complexity of leukemia subtypes, with distinct patterns of expression contributing to the differentiation among groups. The use of the limma package's empirical Bayes methods enhances statistical power while controlling the false discovery rate, ensuring the reliability of these findings.

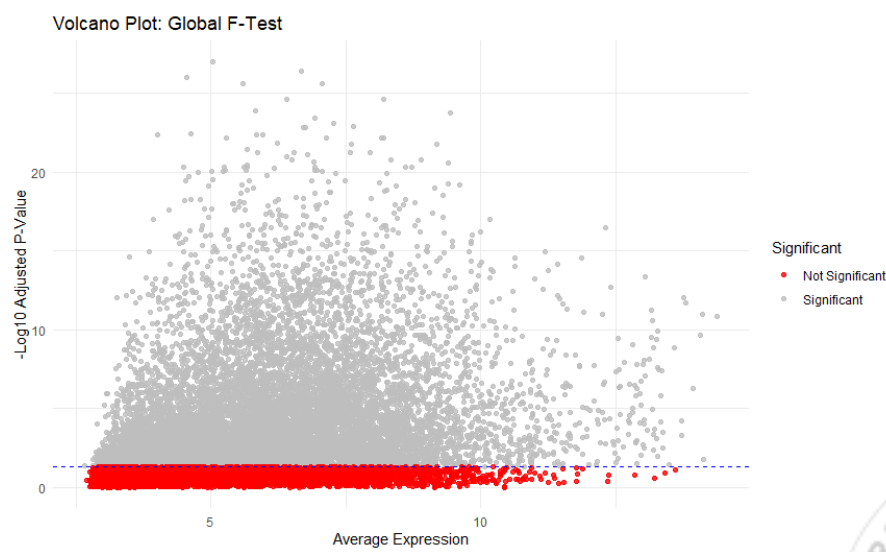


Figure 22 Volcano plot with identified significant genes



The MA plot of the global F-test is a way to visualize the relationship between the average log-expression (x-axis) and log-fold change (y-axis) of genes across the leukemia subtypes and healthy controls. Every point on the plot represents a gene, with its x and y values corresponding to its mean expression and log-fold change relative to the overall mean.

Most genes are located near a  $\log_2$  fold change of zero, as shown in Figure 23, meaning their expression levels do not differ significantly across the groups. This is because not all genes are differentially expressed. However, few genes have log-fold changes with a deviation from zero, indicating that the expression levels for these genes between the groups significantly differ. These genes may be worth exploring in more detail since they may be involved in the difference between leukemia subtypes.

The distribution of points becomes wider at higher average log-expression levels, meaning that log-fold change variability is greater for genes with higher expression levels. This aligns with the expected behavior of microarray data, where technical and biological variation generally increases at higher expression levels. The empirical Bayes analysis in the limma package helps control these variances while maintaining sensitivity in detecting significant genes.

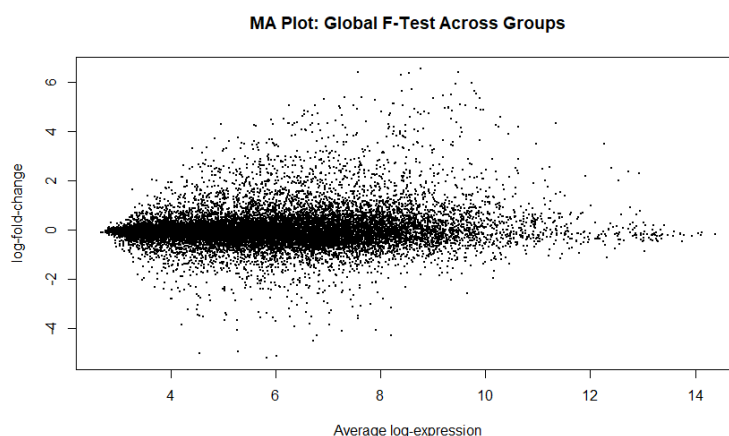
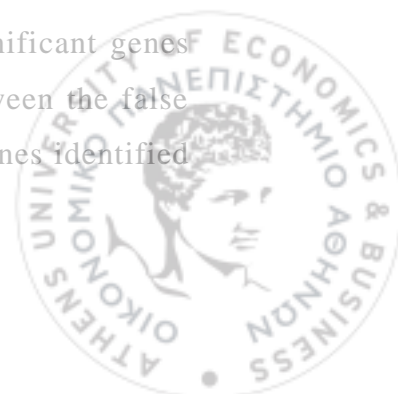


Figure 23 MA plot with average log-expression vs log-fold change

The plot (Table 10, Figure 24) depicting the proportion of significant genes across varying Alpha thresholds illustrates the relationship between the false discovery rate (*FDR*) threshold (x-axis) and the proportion of genes identified



as significant (y-axis). With increasing *FDR* threshold, the proportion of important genes increases linearly.

At the strictest threshold of  $FDR < 0.01$ , approximately 45.2% of genes are classified as significant. This proportion gradually increases, reaching 68.9% at the most lenient threshold of  $FDR < 0.10$ . As concluded, more lenient criteria allow more genes to be identified as significant, however at the cost of potentially inflating the number of false positives.

The smooth and consistent increase in the proportion of significant genes across thresholds demonstrates the robustness of the statistical framework used. The results also highlight the necessity of selecting a suitable *FDR* threshold based on the study aim. For example, a stricter threshold (e.g.,  $FDR < 0.01$ ) may be preferable for exploratory analyses requiring high confidence in the findings, while a more lenient threshold (e.g.,  $FDR < 0.10$ ) may be applied for studies with objectives such as hypothesis generation where inclusiveness is more preferable.

This analysis complements the global F-test results by providing a broader perspective on how the choice of *FDR* threshold impacts the detection of significant genes.

<i>Alpha Levels</i>	<i>Proportion of Significant genes</i>
0.01	0.4524589
0.02	0.5061967
0.03	0.5438231
0.04	0.5740135
0.05	0.6018739
0.06	0.6229427
0.07	0.6416320
0.08	0.6575947
0.09	0.6746480
0.1	0.6894706

Table 10 Proportion of Significant Genes for every Alpha



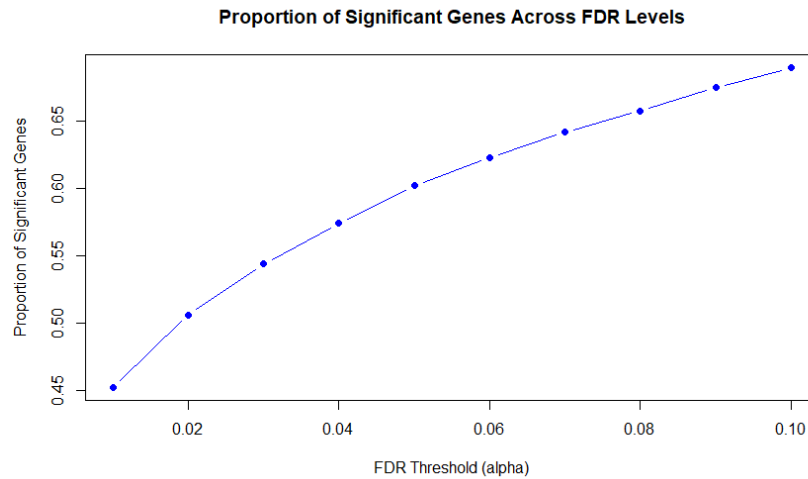


Figure 24 Proportion of Significant Genes per alpha levels

### 2.6.2 Pairwise comparison to identify significant genes

To further investigate the differential expression patterns identified in the global F-test, pairwise comparisons were conducted between each leukemia subtype (ALL, AML, CLL, and CML) and the healthy control group (NoL). The purpose of the comparisons was to identify the genes that were exhibiting significant differential expression specific to each leukemia subtype when compared to the healthy samples, providing a clearer and more accurate representation of subtype-specific patterns of gene expression.

#### *DATA PREPARATION AND ANALYSIS*

The analysis began with the preparation of the expression and phenotype data from the leukemiasEset dataset. The data were filtered to include only the selected groups, with samples categorized into their respective subtypes. Using the limma framework, a design matrix was constructed without intercepts to avoid collinearity issues, enabling robust modeling of the data for pairwise comparisons.

Pairwise contrasts were defined to compare each leukemia subtype against the control group. Linear models were fitted to the data, followed by empirical Bayes moderation to stabilize variance estimates.

#### *VISUALIZATION*



For each pairwise comparison, significant genes were identified based on an *FDR* threshold of 0.05. Volcano plots were generated for each comparison, providing a visualization of the log-fold changes plotted against the adjusted p-values for all genes. Significant genes were highlighted, with thresholds for both significance ( $FDR < 0.05$ ) and fold change ( $|\log FC| > 1$ ) explicitly marked.

In addition to volcano plots, mean difference versus standard deviation plots were generated for each comparison. These plots show the relationship between the variability of gene expression and the magnitude of differential expression.

## Results

The results (Table 12) from the pairwise comparisons between each leukemia subtype and the healthy control group (NoL) revealed a varying number of significant genes across the subtypes. For the comparison of ALL versus NoL, a total of 5,241 significant genes were identified, indicating that a substantial number of genes are differentially expressed in the acute lymphoblastic leukemia subtype compared to the NoL group. The AML versus NoL comparison yielded 3,291 significant genes, reflecting the distinct molecular differences associated with acute myeloid leukemia.

In the case of CLL versus NoL, the analysis identified 6,837 significant genes, the highest among all subtypes, suggesting a broader range of gene expression differences in chronic lymphocytic leukemia compared to the healthy control group. Lastly, the comparison of CML versus NoL resulted in 3,578 significant genes, demonstrating the distinct gene expression profile of chronic myeloid leukemia.

These findings highlight the heterogeneity in gene expression patterns across the leukemia subtypes, with each subtype exhibiting a unique set of differentially expressed genes when compared with the healthy control group. This variation underscores the molecular complexity of leukemia and the importance of subtype-specific analyses in understanding its underlying biology.

<i>Comparison</i>	<i>Number of Significant genes</i>
	5241



ALL_vs_NoL	
AML_vs_NoL	3291
CLL_vs_NoL	6837
CML_vs_NoL	3578

Table 11 Number of significant genes identified vs NoL comparison

The volcano plot (Figure 25) for the comparison between acute lymphoblastic leukemia (ALL) and the healthy control group (NoL) reveals 5,241 significant genes that meet the criteria of an adjusted  $p$ -value  $< 0.05$  and an absolute  $\log$ -fold change  $> 1$ . These significant genes are symmetrically distributed, with upregulated genes (positive log-fold change) on the right and downregulated genes (negative log-fold change) on the left. There is a tight cluster of significant genes near the log-fold change boundary, which corresponds to moderate but biologically significant variation in expression levels. There is also a cluster of genes with extreme log-fold changes (beyond  $\pm 2.5$ ), which corresponds to highly differentially expressed genes. Non-significant genes are primarily concentrated near the origin, indicating minimal fold changes and higher  $p$ -values.

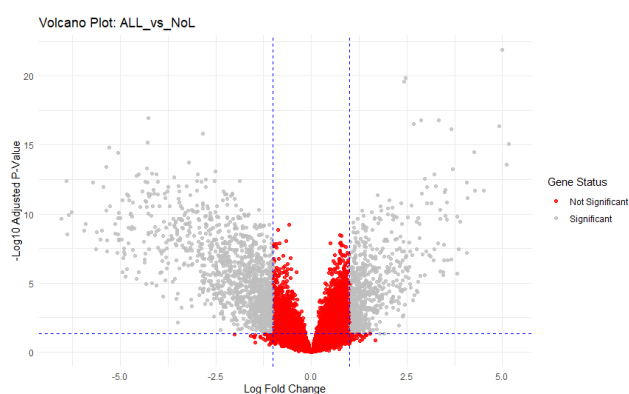


Figure 25 Volcano plot ALL vs NoL

The volcano plot (Figure 26) for AML vs. NoL reveals 3,291 significant genes (gray points), identified by an adjusted  $p$ -value  $< 0.05$  and an absolute  $\log$ -fold change  $> 1$ . A noticeable trend in this plot is the slightly higher



concentration of significant genes on the left side, corresponding to downregulated genes (negative log-fold changes) in AML compared to NoL. This slight asymmetry suggests that a larger proportion of differentially expressed genes in AML exhibit reduced expression levels relative to the control group. Additionally, the genes are also more spread out vertically towards the left, indicating greater variability in adjusted p-values among downregulated genes. Compared to the ALL vs. NoL volcano plot, where significant genes were more symmetrically distributed, this plot suggests distinct transcriptional alterations in AML. Non-significant genes (red points) are primarily clustered near the origin, reflecting minimal fold changes and higher p-values.

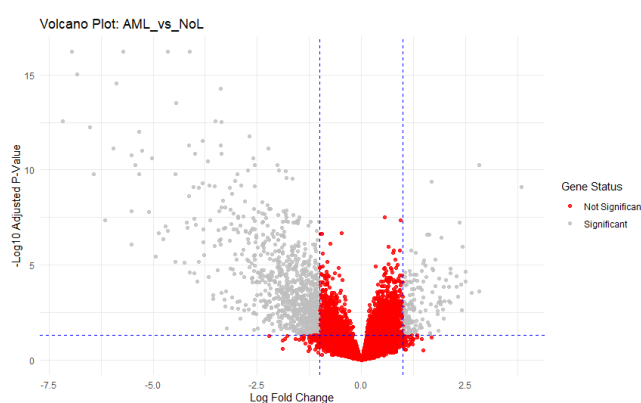


Figure 26 Volcano plot AML vs NoL

The volcano plot (Figure 27) for the comparison between CLL and NoL identifies 6,837 significant genes (gray points), highlighting extensive molecular differences between the two groups (adjusted  $p$ -value  $< 0.05$  and absolute  $\log$ -fold change  $> 1$ ). Unlike the AML vs. NoL comparison, the distribution of significant genes in this plot is more symmetrical around the y-axis, indicating a relatively balanced number of upregulated and downregulated genes. However, a greater vertical spread among upregulated genes suggests that more of these genes have particularly strong statistical significance (lower adjusted p-values). Compared to previous comparisons, this distribution suggests that CLL is associated with widespread transcriptional alterations, affecting a large number of genes across a broad range of expression levels. Non-significant genes (red points) remain clustered near the origin, indicating



minimal fold changes and higher p-values, consistent with the patterns observed in other leukemia subtypes.

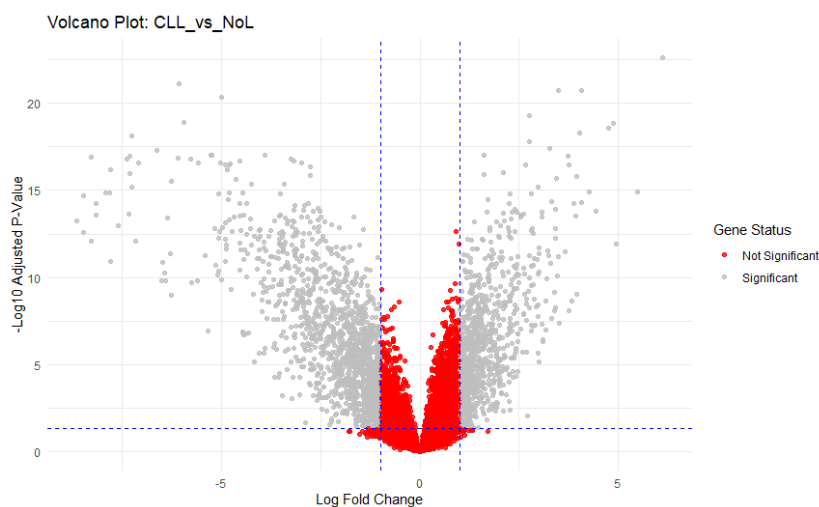


Figure 27 Volcano plot CLL vs NoL

The volcano plot (Figure 28) for the CML vs. NoL comparison presents a symmetrical distribution of 3,578 significant genes, represented by gray points. Downregulated genes (negative log-fold change) show a slightly higher concentration, but since upregulated and downregulated genes are evenly distributed, molecular changes appear relatively balanced in CML compared to NoL.

The most significant genes are positioned near the log-fold change threshold, while the majority of non-significant genes (red points) cluster near the origin, reflecting minimal fold changes and higher p-values. The statistical significance of gene expression changes in CML is evident, yet these changes remain moderate compared to the more extreme variations seen in other comparisons.

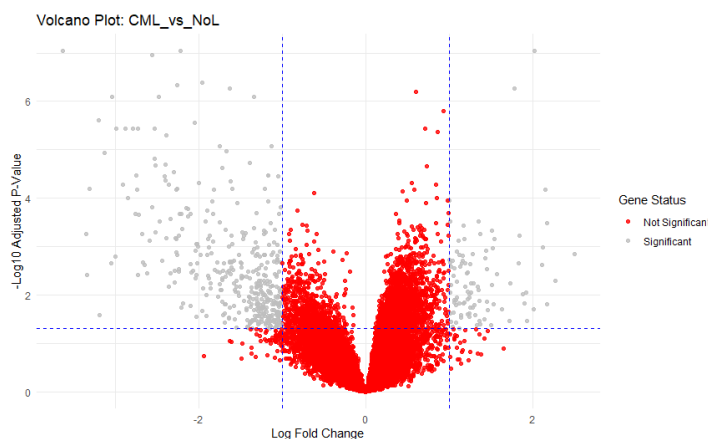


Figure 28 Volcano plot CML vs NoL

The plot for ALL vs. NoL (Figure 29) reveals a clear pattern of differential gene expression, with red bubbles representing DE genes and blue bubbles representing non-DE genes. A high number of DE genes (red points) are identified across both upregulated and downregulated directions, suggesting that ALL exhibits widespread transcriptional changes compared to NoL. The distribution of DE genes appears symmetric, indicating a relatively balanced presence of both upregulated and downregulated genes in ALL. Larger bubbles, corresponding to genes with lower p-values (higher statistical significance), are observed predominantly at the extreme ends of the mean difference axis. Non-DE genes (blue points) cluster around the mean difference of zero, showing minimal expression changes between the two groups. The spread of standard deviations is moderate, with most genes maintaining relatively low variability, although some DE genes exhibit higher standard deviations, reflecting increased expression variability in ALL.

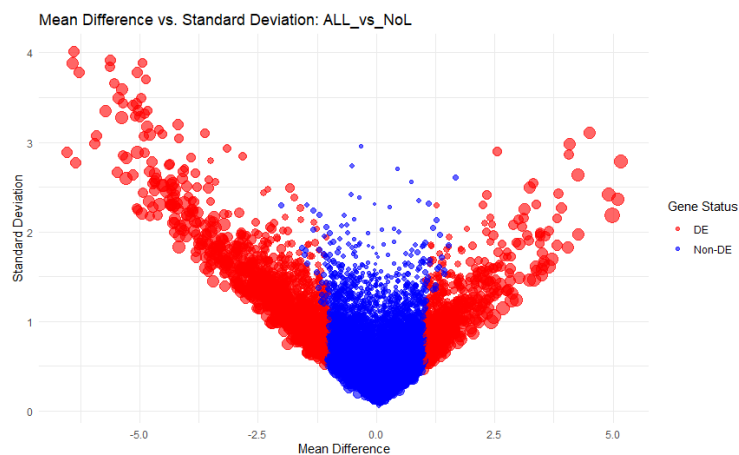
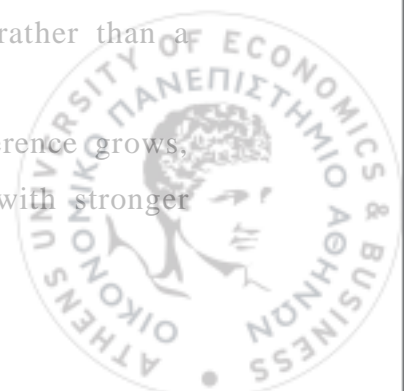


Figure 29 Mean difference vs Standard Deviation for ALL vs NoL

A similar pattern appears in the AML vs. NoL comparison (Figure 30), where red bubbles indicate DE genes and blue bubbles signify non-DE genes. Most DE genes (red points) are positioned on the left side of the plot, suggesting a strong trend of gene downregulation in AML compared to NoL. This asymmetry implies that AML is characterized more by gene repression rather than a balanced upregulation and downregulation of expression.

Standard deviation values increase as the absolute mean difference grows, forming a funnel-like shape, which is expected since genes with stronger



differential expression tend to exhibit greater variability. The larger bubble sizes correspond to genes with higher statistical significance, meaning that genes with lower p-values tend to show larger mean differences. The blue non-DE genes cluster near the center of the plot, reflecting small mean differences and low variability, consistent with background noise in gene expression data.

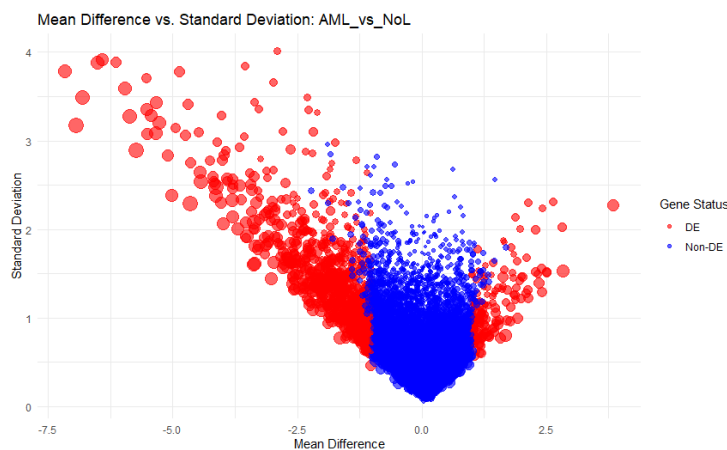


Figure 30 Mean difference vs Standard Deviation for AML vs NoL

The mean difference vs. standard deviation plot for CLL vs. NoL (Figure 31) reveals a clear separation between differentially expressed (DE) genes (red points) and non-DE genes (blue points). The funnel-shaped distribution indicates that genes with larger absolute mean differences also tend to have higher variability. A substantial number of DE genes are present on both the upregulated and downregulated sides. The non-DE genes are tightly clustered near the zero-mean difference, reflecting minor expression differences between the two groups. The distribution of DE genes appears to be somewhat asymmetrical, with a higher density of downregulated genes. Larger red bubbles at the extremes indicate genes with the highest differential expression and highest statistical significance.



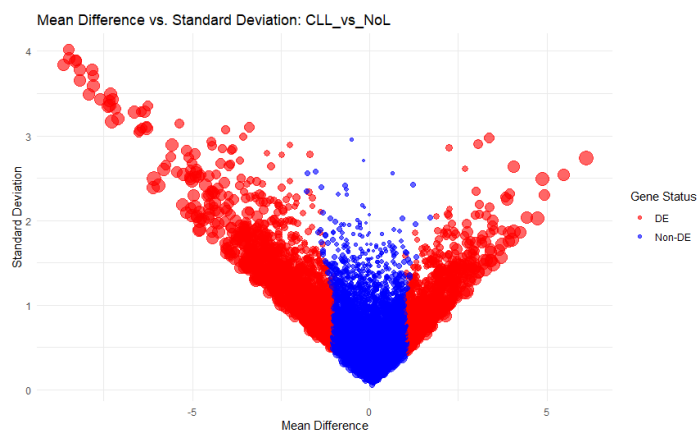


Figure 31 Mean difference vs Standard Deviation for CLL vs NoL

The mean difference vs. standard deviation plot for CML vs. NoL (Figure 32) shows a distinct pattern compared to previous plots, with a more dispersed distribution of non-DE genes (blue points). Unlike other leukemia subtypes, the non-DE genes exhibit greater variability, forming a wider spread along the standard deviation axis, particularly at higher standard deviations. This suggests that CML samples may have greater heterogeneity in gene expression levels compared to the other leukemia types. The DE genes (red points) are still clearly separated, but there is a noticeable asymmetry, with a higher density of downregulated genes (negative mean difference values) compared to upregulated genes. Additionally, the top section of the plot appears more scattered, indicating that some DE genes exhibit unusually high variability, which may contribute to increased noise in differential expression analysis.

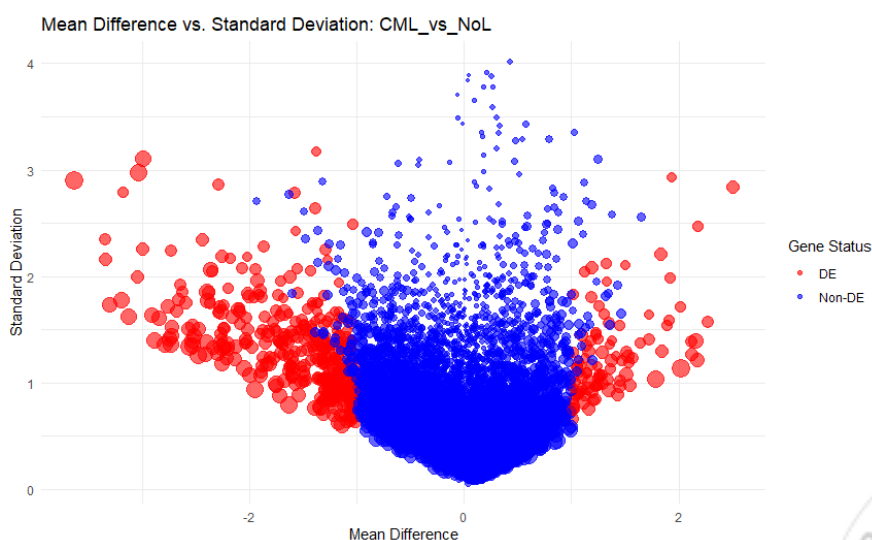


Figure 32 Mean difference vs Standard Deviation for CML vs NoL

The Venn diagram (Figure 33) illustrates the overlap of differentially expressed (DE) genes across four comparisons: ALL vs. NoL, AML vs. NoL, CLL vs. NoL, and CML vs. NoL. Each comparison has a distinct set of unique DE genes (Table 12), such as 1,467 genes exclusive to ALL vs. NoL, 710 to AML vs. NoL, 2,694 to CLL vs. NoL, and 1,911 to CML vs. NoL, highlighting subtype-specific transcriptional changes. Shared intersections reveal key commonalities across comparisons. Notably, 289 DE genes are common across all four comparisons, suggesting these genes may represent a core set of expression changes distinguishing leukemia subtypes from healthy controls. Smaller subsets are shared among three groups, such as the 421 genes common to ALL, and AML, but not CLL and CML, and 102 genes common to AML, CLL, and CML but not ALL. The intersections also include specific pairwise overlaps, like 1,347 DE genes shared between ALL vs. NoL and CLL vs. NoL, signifying similarities between these subtypes. This pattern of overlapping and unique genes provides insights into both shared and subtype-specific molecular alterations.

<i>Comparison</i>	<i>Number of Unique genes</i>
ALL vs NoL	5241
AML vs NoL	3291
CLL vs NoL	6837
CML vs NoL	3578
Shared Across All	289

Table 12 Number of unique genes per comparison



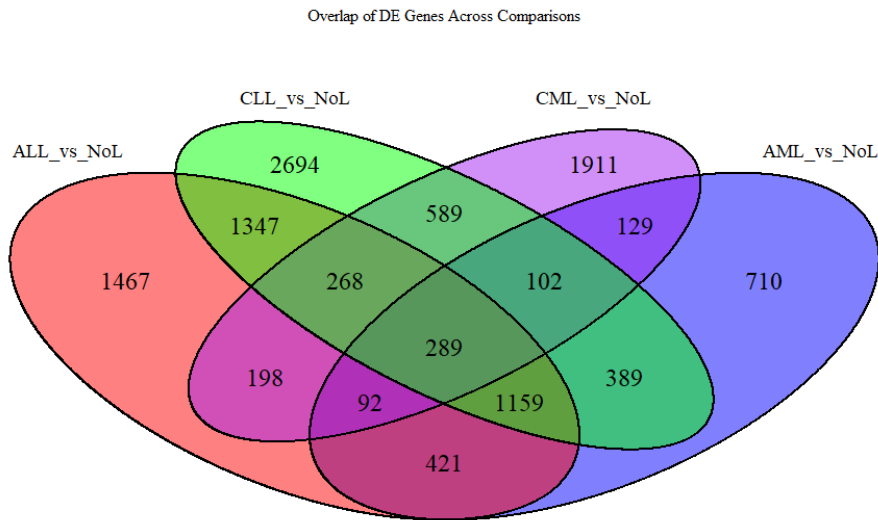
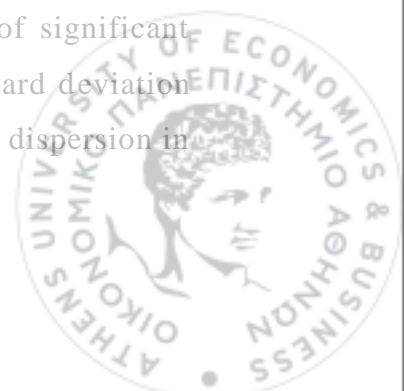


Figure 33 Venn Diagram for unique and common genes identified

### Conclusion

The gene expression profile comparison between leukemia subtypes (ALL, AML, CLL, and CML) and the normal control group (NoL) yielded valuable information on the molecular similarities and differences between these groups. The global F-test identified a large number of significant genes (12,141) that exhibited differential expression between all groups. The results highlight the global variation in gene expression between the leukemia subtypes and the control, demonstrating the utility of a global approach to discover broad patterns of differential expression. Visualization using volcano and MA plots reinforced these findings, demonstrating a dense concentration of significant genes near moderate log-fold changes, with a subset exhibiting extreme expression differences.

Pairwise comparisons provided further granularity by identifying DE genes specific to each leukemia subtype relative to the control group. ALL vs. NoL exhibited the largest number of DE genes (5,241), followed by CLL vs. NoL (6,837), AML vs. NoL (3,291), and CML vs. NoL (3,578). Volcano plots highlighted symmetrical distributions of DE genes around the log-fold change axis, with subtype-specific nuances, such as a broader spread of significant genes in AML and CLL comparisons. Mean difference vs. standard deviation plots further revealed subtype-specific patterns of variability and dispersion in gene expression.



Overlap analyses, visualized through Venn diagrams, revealed a shared core of DE genes across comparisons while also emphasizing unique molecular signatures for each subtype. The largest intersection was observed between CLL and ALL, which suggested shared molecular mechanisms, while between AML and ALL the profiles were more divergent.

In summary, the combined global and pairwise analyses highlight both broad and specific patterns of differential expression, providing a detailed insight into the molecular underpinnings of leukemia subtypes. While the global test captured overall differences, pairwise analyses allowed the detection of subtype-specific DE genes and their intersections. These findings provide the basis for further exploration into the biological mechanisms and potential therapeutic targets underlying these molecular profiles.



## CHAPTER 3.

### CLUSTERING

#### 3.1 Introduction to Clustering methods

Clustering is a frequently used technique in microarray data analysis, through which genes are clustered into coherent groups depending on their expression profiles. Clustering is very useful to study co-expression of genes, identify regulatory networks, and find patterns that might be connected to biological processes.

Gene clustering is particularly useful in the examination of high-dimensional data, such as those generated in microarray experiments, in which thousands of genes are measured simultaneously under a range of conditions or samples. In contrast to most other analyses, clustering does not rely on preconceived notions or hypotheses. Instead, it recognizes natural patterns in the data, so it is an exploratory technique adaptable to many applications. Clustering applications might encompass the detection of co-regulated gene sets, discrimination between treatment conditions, and explanation of phenotype-related pathways.

The power of clustering in microarray analysis is that it can transform noisy and complex data into interpretable patterns. Genes in the same cluster are likely to have related biological functions or be part of related pathways. However, the noise, variability, and sparsity of microarray data create challenging problems, and therefore robust and flexible clustering algorithms are required.

This chapter will present the theory of clustering techniques. It addresses traditional and modern techniques, with a focus on those used for microarray datasets. By examining the underlying theory and applications of the methods, this chapter aims to provide a comprehensive overview of how clustering facilitates the interpretation of gene expression data.



## 3.2 Traditional Clustering methods

The field of gene expression data analysis has traditionally relied on clustering methods, including hierarchical clustering and k-means clustering, as its core analytical tools. The method groups genes by their expression levels to organize and visualize large datasets through distance or similarity-based approaches. Traditional clustering methods were preferred in the initial stages of microarray data analysis because they were both simple and computationally efficient, while advanced probabilistic tools remained less common.

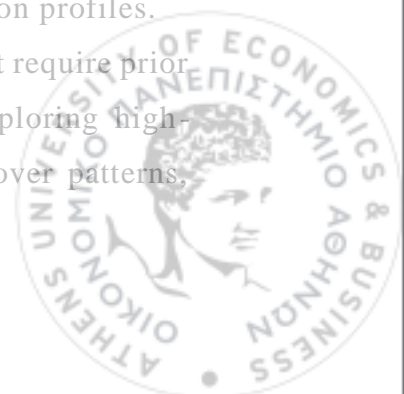
Researchers frequently chose hierarchical clustering to create dendrograms, which enabled both gene relationship visualization and exploratory analyses of co-expression patterns. The iterative data partitioning process of k-means clustering gained widespread acceptance because of its flexible implementation and simple workflow. Through their iterative approaches, traditional clustering methods have significantly contributed to gene clustering research.

Traditional clustering methods show some limitations however when applied to complex, high-dimensional microarray data. Gene expression noise, along with high variability inherent in these data, strongly affects these methods, causing their clustering results to become distorted. The trial-and-error nature of these methods creates challenges because they require manual specification of cluster numbers and parameter settings, which leads to inconsistent results. Hierarchical clustering tends to produce chaining effects during analysis, while k-means clustering faces difficulties from high-dimensional data sparsity as well as non-spherical cluster shapes.

### 3.2.1 Hierarchical Clustering

Hierarchical clustering is a key technique used in gene expression analysis, allowing researchers to group genes with similar expression patterns into clusters. It produces a dendrogram, which is a tree-like diagram that visually shows the relationships between genes or samples. The visualization makes it easier to interpret how genes are grouped based on their expression profiles.

One of the main benefits of hierarchical clustering is that it doesn't require prior knowledge of the number of clusters, making it useful for exploring high-dimensional data, such as microarray datasets. It can help uncover patterns,



identify genes that work together, and find biological pathways or disease mechanisms. For example, hierarchical clustering is often used in cancer research to group genes that have similar patterns of expression across different conditions.

There are two primary methods of hierarchical clustering: agglomerative and divisive. Agglomerative clustering, the more commonly used method, starts by treating each gene as its own cluster and then gradually merges the closest clusters. This bottom-up approach is computationally efficient, especially for small to medium-sized datasets. On the other hand, divisive clustering works in the opposite direction, starting with all genes in one large cluster and splitting them into smaller ones. While divisive clustering can be useful, it is more computationally intensive and less practical for large datasets.

### *INITIALIZATION*

The initialization step of hierarchical clustering sets the foundation for constructing a dendrogram by treating each gene or sample in the dataset as an individual cluster.

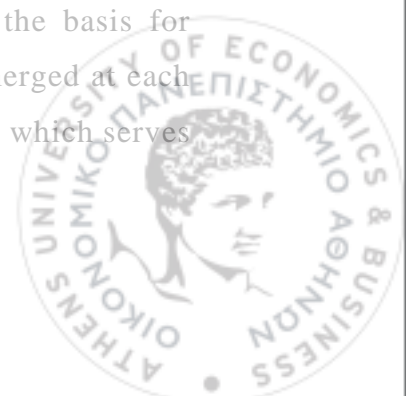
In this stage, the dataset is represented as a collection of points in a high-dimensional space.

Let  $X = \{x_1, x_2, \dots, x_n\}$  denote the dataset, where each  $x_i$  corresponds to a gene or sample described by its expression levels across multiple conditions or experiments. At the start, each  $x_i$  is treated as its own cluster, forming  $n$  individual clusters:  $C = \{C_1, C_2, \dots, C_n\}$ , where  $C_i = \{x_i\}$ .

Here,  $C_i$  represents the  $i$ -th cluster containing only one data point.

### *DISTANCE CALCULATION*

Once the hierarchical clustering algorithm initializes by treating each data point (e.g., each gene) as its own cluster, the next critical step involves calculating the pairwise distances between these data points. This forms the basis for determining which clusters are the most similar and should be merged at each iteration. The computed distances are stored in a distance matrix, which serves as a guide for the clustering process.



### *Distance Metrics*

The choice of distance metric is essential in hierarchical clustering, as it directly affects the structure of the resulting dendrogram and the quality of the clusters. For both hierarchical clustering and k-means clustering, the distance metric determines how the similarity or dissimilarity between data points is measured. The appropriate metric depends on the nature of the data and the specific clustering objectives.

The distance metrics for clustering are presented in [Appendix Tables Section](#). Table 17 includes standard metrics such as Euclidean distance, which captures overall dissimilarity, and more specialized metrics like correlation-based distance, which focuses on expression pattern similarity. For applications where proportional differences or multivariate relationships are significant, metrics such as Canberra distance and Mahalanobis distance are also considered.

### *Constructing the Distance Matrix*

The calculated distances are organized into a distance matrix  $D$ , which is symmetric with diagonal entries of zero. For a dataset with  $n$  data points, the distance matrix  $D$  is of size  $n \times n$ , with entries:

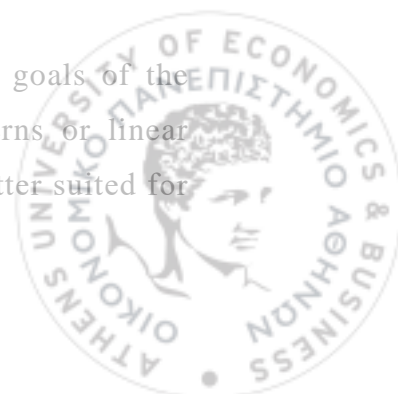
$$D_{ij} = d(x_i, x_j), \text{ for } i, j = 1, \dots, n, \text{ and } i \neq j.$$

The distance matrix provides a comprehensive summary of the pairwise dissimilarities between all data points, serving as the input for the next step: linkage calculations.

### *LINKAGE METHODS*

The choice of linkage method directly impacts the clustering results, influencing the shape of the dendrogram and the interpretation of the clusters. Instead of detailing the formulas for each linkage method here, the specific mathematical definitions and formulas can be found in [Appendix Tables Section](#) (Table 18), which provides a comprehensive overview of linkage methods and their applications.

The choice of linkage method depends on the dataset and the goals of the analysis. Single linkage is ideal for detecting elongated patterns or linear relationships, while complete linkage and Ward's method are better suited for



compact, well-separated clusters. Average linkage offers a balanced approach for exploratory analyses, and correlation-based metrics can enhance the biological relevance of clustering results in gene expression studies.

### *ITERATIVE MERGING*

The iterative merging process is the core of agglomerative hierarchical clustering. After the initialization step, where each data point is treated as its own cluster, and the distance matrix is calculated, the algorithm proceeds to iteratively merge the two most similar clusters based on the chosen distance metric and linkage method. This process continues until all data points are merged into a single cluster, resulting in the hierarchical structure represented as a dendrogram. In each iteration, the algorithm identifies the pair of clusters A and B with the smallest distance  $d(A,B)$ , as determined by the selected linkage method. Once the pair of closest clusters is identified, they are merged into a new cluster C, and the distance matrix is updated to reflect the newly formed cluster. The distances between the new cluster C and the remaining clusters are recalculated using the same linkage method. This process of finding the closest clusters, merging them, and updating the distance matrix is repeated iteratively. The hierarchical relationships among clusters are captured in a dendrogram, where the height of each node represents the distance at which the corresponding clusters were merged.

### *STOPPING CRITERION*

The stopping criterion in hierarchical clustering determines when the iterative merging process should terminate. In agglomerative hierarchical clustering, the process continues until all data points are merged into a single cluster. However, this complete hierarchical structure is not always the most useful for practical applications, and additional criteria are often applied to extract meaningful clusters from the dendrogram.

#### *Full Hierarchical Clustering*

The standard stopping criterion is to stop merging once a single cluster containing all data points is formed. This complete hierarchy is useful for visualizing the relationships between clusters in a dendrogram. Each branch



point (or node) in the dendrogram corresponds to a merge, and the height of the branch reflects the distance between the merged clusters. This approach provides a comprehensive view of the data structure, but further analysis is required to decide where to "cut" the dendrogram to obtain meaningful clusters.

#### *Cutting the Dendrogram*

To identify a specific number of clusters or optimize clustering based on some criteria, the dendrogram is "cut" at a chosen height. The height represents the dissimilarity threshold above which clusters are considered distinct. The optimal height can be determined using one of the following methods:

- **Predefined Number of Clusters:** The user specifies the desired number of clusters, and the dendrogram is cut to achieve this number. This approach is straightforward but may not align with the natural structure of the data.
- **Dynamic Tree Cutting:** Algorithms such as dynamic tree cutting identify clusters based on the shape of the dendrogram, automatically detecting meaningful subdivisions without requiring a predefined number of clusters.
- **Distance Threshold:** The user sets a maximum allowable distance for clusters, and any merges above this threshold are excluded, yielding a more interpretable set of clusters.

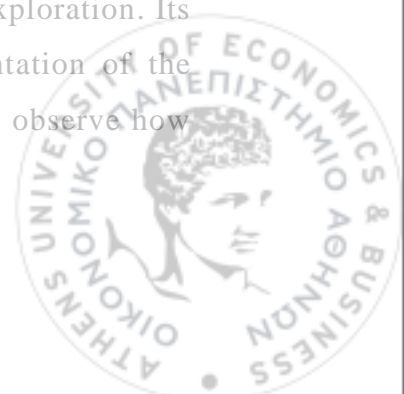
#### *Practical Stopping Criteria for Gene Expression Data*

In gene expression analysis, the stopping criterion often depends on biological considerations. For example:

- If the goal is to identify co-expressed gene groups, the dendrogram might be cut at a height where clusters correspond to biologically meaningful groups of genes.
- For clustering samples, the cut height might be chosen to identify subtypes of diseases or experimental conditions.

### **ADVANTAGES OF HIERARCHICAL CLUSTERING**

Hierarchical clustering is a reliable and widely used method in the analysis of gene expression with several advantages for unsupervised data exploration. Its largest asset is the dendrogram, a tree-formed visual representation of the hierarchical relationships between data points, making it easy to observe how



the data clusters. It facilitates the observation of fine-scale patterns and as well as broader trends within the data.

One of the benefits of hierarchical clustering is that it is not rigid, and it does not require the pre-specification of the number of clusters. This is particularly useful in exploratory analyses where the researcher may not know the number of clusters. By adjusting the height at which the dendrogram is cut, researchers can ascertain how many clusters to produce based on their own research needs. The algorithm supports numerous distance measurements and linkage methods and can be optimized for specific types of data. Correlation-based distances have been used in the analysis of gene expression to find co-regulated genes, while other distance metrics such as Manhattan or Euclidean distances are generally more universal. Another advantage of hierarchical clustering is that it is deterministic, i.e., it produces the same results given a specific dataset and a specific set of parameters, compared to algorithms such as k-means, whose outputs can differ depending on the initial conditions.

Moreover, hierarchical clustering is also effective for small datasets, making it appropriate in research with reduced samples or genes. Hierarchical clustering is also applicable for various types of data, including numerical, categorical, or even mixed data types. Its versatility has made hierarchical clustering a standard technique in gene expression analysis, where data information can be of various forms.

#### *LIMITATIONS OF HIERARCHICAL CLUSTERING*

Hierarchical clustering, while useful, has several limitations, especially for large data or noisy data. One major drawback is its computational inefficiency, since it requires the calculation of a distance matrix at each iteration resulting in increased time and space complexity. The method is also sensitive to noise and outliers, with techniques such as single linkage and complete linkage being prone to issues like "chaining effects" and distortion by extreme values. Hence careful data preprocessing becomes a necessity.

Another limitation is the lack of flexibility in readjusting cluster membership after a merge, leading to irreversible clustering mistakes, particularly when clusters are poorly separated. Moreover, hierarchical clustering interpretability



can be subjective, since the decision on the "correct" cut height for defining clusters is typically arbitrary and might fail to represent the underlying data structure. Finally, hierarchical clustering assumes a tree-like structure of clusters, which may not always hold for actual data, especially if clusters overlap or have more complex relations, requiring more advanced methods like model-based or graph-based clustering.

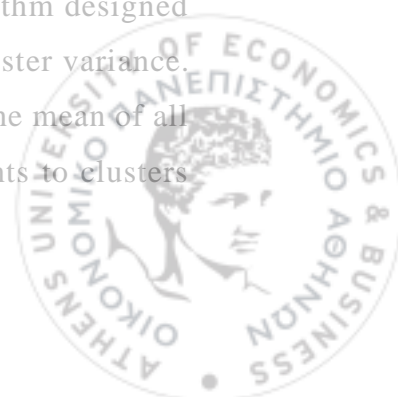
### *EXTENSIONS AND IMPROVEMENTS*

Hierarchical clustering has also introduced a number of extensions to address its limitations in handling noisy and high-dimensional data. Hybrid approaches, which combine hierarchical clustering with model-based approaches like Gaussian Mixture Models (GMMs), have been a commonly used enhancement. Hybrid approaches add statistical rigor through the application of the Expectation-Maximization (EM) algorithm in GMMs, thus improving cluster quality through the application of the EM algorithm. This hybrid approach has been used successfully in gene expression analysis, offering better robustness against noise.

Stability analysis is another enhancement, which measures the stability of clustering solutions using techniques like bootstrap resampling. By inspecting cluster consistency over multiple runs, researchers can validate robust clusters. Dynamic tree cutting can also be used for cluster selection based on stability statistics. Lastly, improved distance measures such as Mahalanobis distance and correlation-based measures provide more accurate clustering results since they consider the covariance structure, and they focus on expression patterns and not their magnitudes. These, as discussed in McLachlan et al., enhance hierarchical clustering's applicability and performance for gene expression analysis.

#### 3.2.2 K-means Clustering

k-Means clustering is a widely used unsupervised learning algorithm designed to partition a dataset into  $k$  clusters by minimizing the within-cluster variance. Each cluster is represented by a centroid, which is calculated as the mean of all points assigned to it. The algorithm iteratively assigns data points to clusters



and updates centroids until convergence is achieved. Its primary goal is to ensure that points within the same cluster are as similar as possible, while points in different clusters are as dissimilar as possible.

In the context of gene expression analysis, k-means is particularly effective for identifying co-regulated genes or clustering samples into biologically meaningful subtypes, such as disease versus healthy states. Its simplicity and computational efficiency make it a popular choice for large datasets, including microarray studies. However, the method assumes that clusters are spherical and of similar size, which may not always align with the complex structures found in gene expression data. Despite these limitations, k-means remains a foundational method in clustering, serving as a benchmark for more advanced techniques.

By leveraging its iterative optimization approach, k-means provides an intuitive framework for exploring high-dimensional datasets. Its popularity stems from its balance of simplicity and practical utility, making it a starting point for many clustering analyses in biological research.

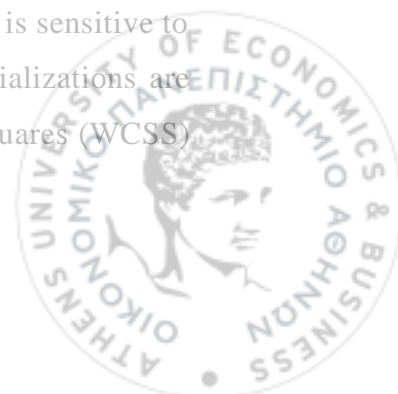
### *THE K-MEANS ALGORITHM*

The k-means algorithm is a centroid-based clustering method that partitions a dataset into k distinct clusters by minimizing the within-cluster variance. The algorithm operates through a series of iterative steps, involving initialization, assignment, and updating of centroids, and continues until a convergence criterion is met.

#### *Initialization*

The algorithm begins by selecting k initial centroids, which serve as representatives for the clusters. These centroids can be chosen randomly or using optimized methods such as k-means++, which improves clustering stability by spreading the initial centroids more strategically. Mathematically, the initial set of centroids is denoted as:  $C = \{c_1, c_2, \dots, c_k\}$ , where  $c_i$  represents the centroid of the  $i$ -th cluster.

Poor initialization can lead to suboptimal solutions, as the algorithm is sensitive to the starting positions of the centroids. To avoid this, multiple initializations are often run, and the solution with the lowest within-cluster sum of squares (WCSS) is selected.



### Assignment Step

In the assignment step, each data point is assigned to the cluster with the closest centroid, based on a chosen distance metric. The most common metric is the squared Euclidean distance, defined as:

$$d(x_i, x_j) = \|x_i - x_j\|^2 = \sum_{p=1}^m (x_{ip} - c_{jp})^2$$

where  $x_i$  is the data point,  $c_j$  is the centroid of the  $j$ -th cluster, and  $m$  is the number of dimensions. A data point  $x_i$  is assigned to cluster  $j$  if:

$$x_i \in C_j \Leftrightarrow d(x_i, c_j) \leq d(x_i, c_l) \quad \forall l = 1, \dots, k$$

For alternative distance metrics, such as Manhattan or correlation-based distances, see Table 17 in [Appendix Tables Section](#).

### Update Step

Once the data points are assigned to clusters, the centroids are recalculated to reflect the updated memberships. The new centroid  $c_j$  for cluster  $j$  is computed as the mean of all points assigned to that cluster:

$$c_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

where  $|C_j|$  is the number of points in cluster  $j$ . The recalculated centroids serve as the new representatives for the clusters in the next iteration.

### Iteration and Convergence

The assignment and update steps are repeated iteratively until the algorithm converges. Convergence is typically defined by one of the following criteria:

1. The centroids stabilize and do not change significantly between iterations.
2. A predefined maximum number of iterations is reached.
3. The decrease in *WCSS* between iterations falls below a set threshold.

The objective function minimized by k-means is the within-cluster sum of squares (*WCSS*), given by:

$$WCSS = \sum_{j=1}^k \sum_{x_i \in C_j} \|x_i - c_j\|^2$$

Minimizing *WCSS* ensures that clusters are compact and points within the same cluster are similar.



### *ADVANTAGES OF K-MEANS CLUSTERING*

k-Means clustering has been found to be an extremely easy, efficient, and effective method for clustering, particularly for the analysis of gene expression. Among the benefits is its computational simplicity with linear time complexity ( $O(n \cdot k \cdot t)$ ), which makes it appropriate for large data sets like microarray gene expression profiles.

The second advantage is that it is simple to interpret and comprehend. By reducing within-cluster variation, k-means produces easily interpretable clusters represented by centroids, which are calculated as the mean of points in a given cluster. Through this straightforward technique, any biologically meaningful groups such as co-regulated genes or disease subtypes, can be identified even by researchers with minimal computational experience.

A further strength of k-means' flexibility is the universality across applications, varying from gene expression data analysis to disease classification. The capability of the algorithm to handle high-dimensional data via distance measurements like Euclidean distance makes it adaptable to microarray data. Preprocessing such as dimensionality reduction may also enhance its performance.

Finally, k-means is flexible regarding the usage of distance measures, since apart from the Euclidean distance it can also accommodate alternatives such as Manhattan or correlation-based distances, which are specifically relevant for gene expression data analysis. The algorithm is also often used as a baseline method, a benchmark or initialization step in more advanced clustering techniques.

### *LIMITATIONS OF K-MEANS CLUSTERING*

Although k-means clustering has its strengths, it also has a number of weaknesses, especially in gene expression data analysis. A significant weakness is the assumption of equal-sized spherical clusters. More specifically, k-means algorithm functions by minimizing the within-cluster variance with the assumption that clusters are compact and evenly distributed. Gene expression data, however, frequently feature irregularly shaped clusters, varying densities, or overlapping regions, and such settings may yield poor clustering outcomes.

Another issue is that it is sensitive to initial centroid selection. Poor initialization will cause the algorithm to converge to a local minimum, which will results in non-



optimal clusters. While procedures like k-means++ address the issue to some degree, multiple runs are typically needed for stable solutions.

Noise and outliers are also a problem for k-means, as they can impact the centroid calculations, resulting in distorted clusters. Preprocessing, e.g., removal of outliers, is generally necessary to counter this effect.

Additionally, k-means requires the number of clusters ( $k$ ) to be specified beforehand. In exploratory gene expression analysis, the true number of clusters is typically unknown, and estimating  $k$  can be computationally intensive, requiring methods like the Elbow Method or silhouette analysis.

The algorithm's use of Euclidean distance can also be a problem, as it will not always represent the biological relationships in gene expression data. Although other measures, like Manhattan distance or correlation-based measures, can be used, these do not address the algorithm's limitation to handle complex clusters.

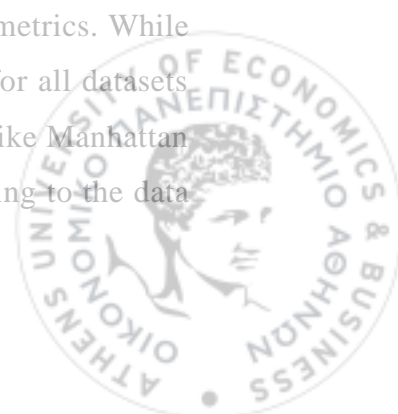
Finally, k-means performs poorly on high-dimensional data without preprocessing. While it can scale to such data, performance degrades when irrelevant or noisy features dominate. Techniques like principal component analysis (PCA) are often used to improve performance, but they can reduce interpretability.

### *EXTENSIONS OF K-MEANS CLUSTERING*

To address the limitations of k-means clustering, several extensions and modifications have been developed, making the algorithm more robust, flexible, and applicable to complex datasets, e.g., gene expression analysis.

One of the key improvements is the k-means++ initialization method, which refines the initial centroid selection. Unlike random initialization, k-means++ places the initial centroids in a dispersed fashion within the data space by probabilistically selecting points that are further apart from one another. This reduces the possibility of poor initialization and helps the algorithm convergence to better solutions. It improves stability, particularly in gene expression analysis, where high dimensionality enhances the impact of initialization on clustering outcomes.

A second important improvement is the use of alternative distance metrics. While squared Euclidean distance is the default, it is not always suitable for all datasets with non-linear relationships or features of varying scales. Metrics like Manhattan distance or Mahalanobis distance provide more flexibility by adapting to the data



structure. These changes allow k-means to better represent gene expression data relationships, where the biological patterns may not align with Euclidean geometry. However, caution is necessary because different measures affect computational efficiency and interpretability.

To overcome the sensitivity of the algorithm to outliers and noise, robust k-means variants have been proposed. These variants modify the objective function to reduce the weight of the outliers' effect so that they do not have a disproportionate impact on centroid calculations. Pre-processing techniques like removal of outliers or data normalization can also make clustering more robust. This is particularly useful in gene expression analysis, where noisy or outlying data points skew the results.

The issue of determining the optimal number of clusters ( $k$ ) has been mitigated with techniques like the Elbow Method and silhouette score analysis. The Elbow Method determines the optimal  $k$  by plotting the within-cluster sum of squares (WCSS) against different values of  $k$  and selecting the point where the rate of WCSS reduction decreases significantly. Silhouette analysis evaluates the quality of clusters by comparing the similarity of a data point to points within its own cluster and to other clusters. These methods offer practical guidance for choosing  $k$ , especially in exploratory gene expression analysis, where the true number of clusters is typically undefined.

### 3.2.3 K-Medoids Clustering

k-Medoids is a partition-based clustering algorithm closely related to k-means but designed to be more robust against noise and outliers. Unlike k-means, which updates cluster centroids by averaging all points within a cluster, k-medoids selects actual data points, known as medoids, to represent each cluster. By using representative points instead of arithmetic means, k-medoids mitigates the impact of extreme values, making it particularly useful for datasets with non-spherical distributions or outliers.

#### *ALGORITHM DESCRIPTION*

The k-medoids algorithm follows a similar iterative process as k-means but differs in how cluster representatives are chosen and updated. The algorithm begins by randomly selecting  $k$  medoids from the dataset. Each data point is then assigned to



the nearest medoid based on a chosen distance metric, typically Manhattan or Euclidean distance. The algorithm then evaluates potential swaps between medoids and other data points within the same cluster to minimize the overall sum of distances. This process repeats until the medoids stabilize, meaning that no further swaps reduce the total clustering cost. By using actual data points as cluster centers rather than computed centroids, k-medoids ensures that the cluster representatives remain meaningful in the context of gene expression data.

- Initialization: Select k medoids (representative data points) randomly from the dataset.
- Assignment Step: Assign each data point to the cluster of the nearest medoid, based on a distance metric (typically Manhattan or Euclidean distance).
- Update Step: For each cluster, evaluate whether swapping the medoid with another data point in the cluster reduces the total sum of distances.
- Iteration: Repeat the assignment and update steps until medoids stabilize (i.e., no further swaps reduce the total cost function).

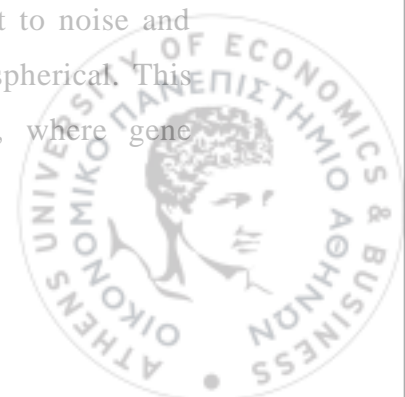
The objective function minimized in k-medoids is:

$$J = \sum_{j=1}^k \sum_{x \in C_j} d(x, m_j)$$

where:  $C_j$  represents the set of points in cluster  $j$ ,  $m_j$  is the medoid of cluster  $j$ , and  $d(x, m_j)$  is the distance function between a data point  $x$  and the medoid  $m_j$ , commonly using Manhattan distance instead of squared Euclidean distance to improve robustness.

#### *ADVANTAGES OF K-MEDOIDS CLUSTERING*

A key distinction between k-medoids and k-means lies in their sensitivity to outliers and cluster representation. Since k-means relies on averaging points to determine cluster centroids, it is highly sensitive to extreme values, which can disproportionately shift the cluster center. In contrast, k-medoids selects representative data points from the dataset, making it more robust to noise and better suited for data distributions that are not well-separated or spherical. This robustness is particularly advantageous in microarray analysis, where gene



expression levels can be highly variable due to technical artifacts or biological heterogeneity.

When applied to microarray gene expression data, k-medoids provides several advantages over k-means. First, it improves cluster stability by ensuring that cluster representatives are actual gene expression profiles rather than abstract computed centroids. This makes it easier to interpret which genes or samples serve as cluster centers. Second, it is less susceptible to the influence of highly variable genes, reducing the risk of misclassification due to extreme expression values. Lastly, k-medoids can be used in conjunction with alternative distance measures, such as Manhattan distance, which may better capture gene expression similarities compared to Euclidean distance.

#### *LIMITATIONS OF K-MEDOIDS CLUSTERING*

Despite its advantages, k-medoids is computationally more expensive than k-means due to the need for pairwise distance computations and iterative medoid swapping. While this makes it less scalable for very large datasets, efficient implementations such as Partitioning Around Medoids (PAM) and its large-scale adaptation, CLARA (Clustering Large Applications), allow k-medoids to be applied to high-dimensional genomic data. Additionally, like k-means, k-medoids requires pre-specification of the number of clusters, which can be determined using methods such as the gap statistic, silhouette score, or Bayesian Information Criterion (BIC).

### 3.3 Model-Based Clustering

Model-based clustering assumes that the data is sampled from a mixture of probability distributions, where each cluster belongs to a distinct underlying distribution. This probabilistic approach allows for more interpretable and flexible clustering, in which data points are assigned to clusters based on posterior probabilities. Unlike the traditional approaches, model-based clustering also provides explicit statistical models for cluster structure, thus it can handle variability and noise in the data.

One of the greatest advantages of model-based clustering is that it can estimate the number of clusters in a data-adaptive manner. By using model selection criteria, e.g., Bayesian Information Criterion (BIC) or Akaike Information Criterion (AIC),



the optimal number of clusters can be identified without any arbitrary pre-specifications. Model-based clustering also has the ability to handle clusters of varying shapes, sizes, and densities using flexible covariance structures in statistical models.

Microarray gene expression data with high dimensionality and high noise level pose serious challenges to conventional clustering methods. Model-based clustering is especially well-suited to overcoming these challenges due to its power and versatility. By modeling the process of data generation, it can separate true biological variation from experimental noise and hence produce more accurate and meaningful clustering results. Additionally, the probabilistic nature of model-based clustering facilitates the identification of patterns that tend to be dominated by noise in traditional approaches.

### 3.3.1 Gaussian Mixture Models (GMMs)

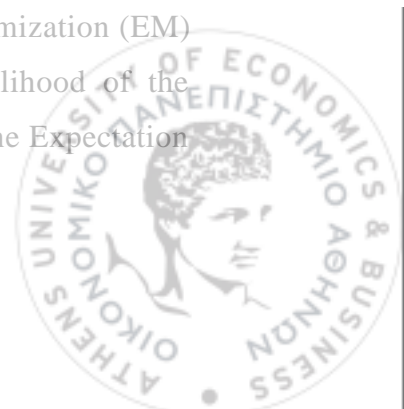
Gaussian Mixture Models (GMMs) are a foundational approach in model-based clustering, offering a probabilistic framework to model complex data structures. GMMs assume that the observed data is generated from a mixture of multivariate Gaussian distributions, where each Gaussian component corresponds to a cluster. This formulation allows for the modeling of data with diverse cluster shapes, sizes, and densities, making GMMs particularly suitable for high-dimensional datasets such as microarray gene expression data.

The mathematical foundation of GMMs lies in representing the probability density of the data as a weighted sum of Gaussian components. Formally, the probability density function of a GMM is given by:

$$P(x_i) = \sum_{k=1}^K \pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k),$$

where  $\pi_k$  is the mixing proportion,  $\mathcal{N}(x_i | \mu_k, \Sigma_k)$  is the multivariate Gaussian density with mean  $\mu_k$  and covariance  $\Sigma_k$ . The parameters  $\mu_k$ ,  $\Sigma_k$ ,  $\pi_k$  and together define the location, shape, and size of each cluster.

To estimate these parameters, GMMs rely on the Expectation-Maximization (EM) algorithm, an iterative procedure designed to maximize the likelihood of the observed data. The EM algorithm alternates between two steps. In the Expectation



step (E-step), the posterior probabilities of cluster membership, known as responsibilities, are calculated for each data point:

$$\tau_{ik} = \frac{\pi_k N(x_i | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j N(x_i | \mu_j, \Sigma_j)}$$

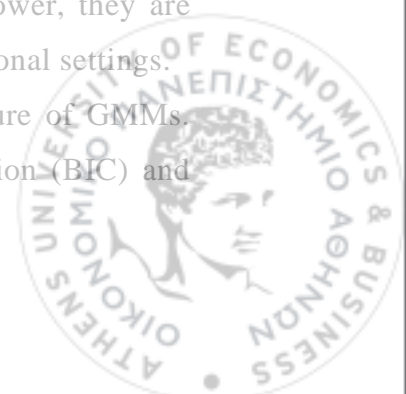
These responsibilities represent the probability that a data point belongs to a cluster. In the Maximization step (M-step), the parameters are updated using the responsibilities to maximize the expected log-likelihood. The updates for the parameters are as follows:

$$\begin{aligned} \text{Mean: } \mu_k &= \frac{\sum_{i=1}^N \tau_{ik} x_i}{\sum_{i=1}^N \tau_{ik}} \\ \text{Covariance: } \Sigma_k &= \frac{\sum_{i=1}^N \tau_{ik} (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^N \tau_{ik}} \\ \text{Mixing proportion: } \pi_k &= \frac{\sum_{i=1}^N \tau_{ik}}{N} \end{aligned}$$

The EM algorithm iteratively refines these parameters until convergence, typically when the change in log-likelihood between iterations falls below a predefined threshold. However, the EM algorithm is sensitive to the initialization of parameters, which can lead to convergence to local optima. Techniques such as hierarchical clustering or k-means are often employed to initialize the parameters and improve the robustness of the algorithm.

An essential aspect of GMMs is the flexibility provided by different covariance structures, which influence the shape and orientation of the clusters. Spherical covariance assumes equal variance in all directions, resulting in clusters with spherical shapes. Diagonal covariance allows variances to differ along each dimension, enabling the modeling of axis-aligned ellipsoidal clusters. Full covariance matrices, on the other hand, provide the most flexibility by capturing correlations between dimensions, allowing for arbitrarily oriented ellipsoidal clusters. While full covariance matrices offer greater modeling power, they are computationally expensive and prone to overfitting in high-dimensional settings.

The selection of the optimal number of clusters is a critical feature of GMMs. Model selection criteria, such as the Bayesian Information Criterion (BIC) and



Akaike Information Criterion (AIC), are widely used to evaluate models with varying numbers of components. The *BIC* is defined as:  $BIC = \ln(N)p - 2\ln(\hat{L})$  and the *AIC* is given by:  $AIC = 2p - 2\ln(\hat{L})$  where  $N$  is the number of data points,  $p$  is the number of parameters, and  $\hat{L}$  is the maximized likelihood.

Despite their strengths, GMMs face challenges in high-dimensional datasets. The EM algorithm's reliance on initialization can result in suboptimal solutions, and computational complexity increases with the number of dimensions and clusters. To address these issues, constrained covariance structures, dimensionality reduction techniques, or robust initialization methods are often applied.

### 3.4 Advanced Model-Based Clustering Techniques

Gaussian Mixture Models (GMMs) use powerful probabilistic models for clustering but applying them to high-dimensional and complex data, e.g., microarray gene expression data, can reveal their shortcomings, such as overfitting, computational complexity, and sensitivity to noise. To counter these limitations, several extensions to GMMs have been introduced, with the objective of enhancing their scalability, robustness, and flexibility. These extensions introduce innovations such as constrained covariance structures, latent variable models, and embedded dimensionality reduction techniques. These collectively enhance the utility of GMMs, making them suitable for more complex and higher-dimensional clustering tasks.

#### 3.4.1 Parsimonious Gaussian Mixture Models (PGMMS)

Parsimonious Gaussian Mixture Models (PGMMs) are a modification of Gaussian Mixture Models (GMM) designed to improve efficiency by placing constraints on covariance structures of the Gaussian components. By reducing the dimension of parameters to be estimated, the PGMM models become more practical for clustering applications, specifically in the scenario of high-dimensional data like microarray gene expression data. In these types of datasets, the number of genes examined is typically far larger than the number of available samples, and therefore parameter reduction becomes a crucial step for enhancing computational feasibility and interpretability.



In GMMs, each cluster is modeled as a multivariate Gaussian distribution defined by a mean vector  $\mu_k$ , a covariance matrix  $\Sigma_k$ , and a mixing proportion  $\pi_k$ . The covariance matrix  $\Sigma_k$  governs the size, shape, and orientation of each cluster. While full covariance matrices provide flexibility, they introduce computational challenges and risk overfitting in high-dimensional datasets.

PGMMs address these challenges by parameterizing the covariance matrices using an eigenvalue decomposition:

$$\Sigma_k = \lambda_k D_k A_k D_k^T$$

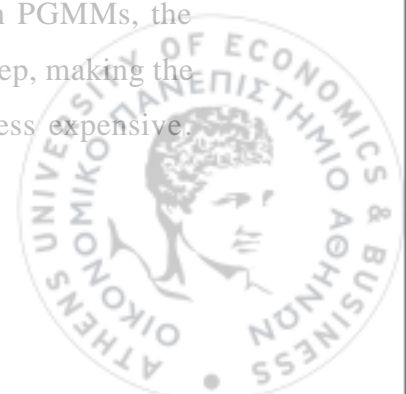
where:  $\lambda_k$  is a scalar controlling the overall volume of the cluster,  $D_k$  is an orthogonal matrix specifying the orientation of the cluster,  $A_k$  is a diagonal matrix determining the shape of the cluster.

By imposing constraints on these components, PGMMs allow for a range of covariance structures:

- Isotropic covariance ( $\Sigma_k = \lambda_k I$ ): Clusters are spherical and have equal variance in all directions.
- Diagonal covariance ( $\Sigma_k = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_p)$ ): Variances are independent along each axis.
- Shared covariance ( $\Sigma_k = \Sigma$ ): All clusters share the same covariance structure.
- General covariance ( $\Sigma_k$  unrestricted): Provides maximum flexibility but may lead to overfitting.

This parsimonious approach reduces the parameter space, such that clustering of high-dimensional data is achievable without sacrificing interpretability.

Parameter estimation in PGMMs is accomplished by using the Expectation-Maximization (EM) algorithm, as with standard GMMs. The algorithm alternates between two steps: the E-step, which calculates the posterior probabilities of cluster membership for each data point, and the M-step, which updates the model parameters to maximize the expected log-likelihood of the data. In PGMMs, the constraints on the covariance matrices are incorporated into the M-step, making the updating process of the parameters simpler and computationally less expensive.



This iterative process continues until convergence, typically defined as an insignificant change in the log-likelihood between iterations.

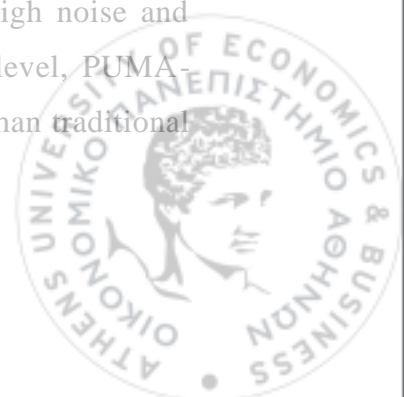
One of the most important aspects of PGMMs is model selection, that is, the determination of the optimal number of clusters and the appropriate covariance structure. Model selection criteria such as the Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC) are widely used for this purpose. These criteria balance model fit, as measured by the likelihood, with model complexity, as indicated by the number of parameters. By penalizing over-parameterized models, BIC and AIC help ensure that the selected model generalizes well to new data.

PGMMs have been successfully applied to microarray gene expression data, where their parsimonious covariance structures address the challenges of high dimensionality and noise. In such datasets, PGMMs perform well at clustering genes with similar patterns of expression or classifying tissue samples based on gene expression profiles. Their ability to handle complex clustering tasks while maintaining computational efficiency has made PGMMs a valuable tool in biological data analysis.

Although PGMMs are powerful, they do have limitations. Selecting the optimal covariance structure and number of clusters can be computationally intensive, especially for large datasets. Furthermore, although the constraints on covariance matrices reduce overfitting, overly restrictive parameterizations may fail to capture complex cluster shapes. Nevertheless, the flexibility and efficiency of PGMMs make them a powerful generalization of GMMs, with applications to high-dimensional and complex data sets such as those in microarray experiments.

### 3.4.2 Puma-Clust

PUMA-CLUST generalizes the Gaussian Mixture Model (GMM) framework by incorporating probe-level measurement error as an integral component of the clustering algorithm. This addition is particularly valuable in the context of microarray gene expression data, which is frequently subject to high noise and variability. Through the introduction of uncertainty at the probe level, PUMA-CLUST generates more biologically meaningful clustering results than traditional GMM-based approaches.



At the core of PUMA-CLUST is the recognition that observed gene expression values ( $y_i$ ) are influenced by both the true expression levels ( $x_i$ ) and measurement noise ( $\varepsilon_i$ ). This relationship is modeled as:

$$y_i = x_i + \varepsilon_i,$$

where  $\varepsilon_i \sim N(0, \text{diag}(v_i))$  and  $v_i$  represents the probe-level measurement variance. The variance  $v_i$  is estimated using probabilistic models like multi-mgMOS, which derive probe-level uncertainties from raw microarray intensity data. These uncertainties allow PUMA-CLUST to explicitly account for the variability inherent in the measurement process, which is typically ignored by traditional clustering algorithms.

Each cluster  $k$  in PUMA-CLUST is parameterized as a multivariate Gaussian distribution:

$$p(x_i | k; \theta_k) = N(x_i | \mu_k, \Sigma_k),$$

where  $\mu_k$  is the mean vector for cluster  $k$ ,  $\Sigma_k$  is the covariance matrix for cluster  $k$ ,  $\theta_k = \{\mu_k, \Sigma_k, \pi_k\}$  are the parameters of the model. By incorporating the probe-level variance, the observed data likelihood becomes:

$$p(x_i | k; \theta_k) = N(x_i | \mu_k, \Sigma_k + \text{diag}(v_i)).$$

This formulation augments the covariance matrix of each cluster with the measurement variance,  $\text{diag}(v_i)$ , which adjusts the model to account for gene-specific noise. This is especially important for genes with low expression levels, where noise tends to dominate the signal.

The parameters of the PUMA-CLUST model are estimated using the Expectation-Maximization (EM) algorithm. The EM algorithm iteratively maximizes the likelihood of the observed data by alternating between two steps:

E-step: Compute the posterior probabilities ( $\tau_{ik}$ ) of cluster memberships for each data point  $x_i$ :

$$\tau_{ik} = \frac{\pi_k N(x_i | \mu_k, \Sigma_k + \text{diag}(v_i))}{\sum_{j=1}^K \pi_j N(x_i | \mu_j, \Sigma_j + \text{diag}(v_i))}.$$

These posterior probabilities represent the likelihood that the data point  $x_i$  belongs to cluster  $k$ , adjusted for the probe-level noise.

M-step: Update the model parameters to maximize the expected log-likelihood:



The mean vector for cluster  $k$  is updated as:

$$\mu_k = \frac{\sum_{i=1}^N \tau_{ik} x_i}{\sum_{i=1}^N \tau_{ik}}$$

The covariance matrix for cluster  $k$  is updated to reflect both the variability within the cluster and the noise measurement:

$$\Sigma_k = \frac{\sum_{i=1}^N \tau_{ik} (x_i - \mu_k)(x_i - \mu_k)^T - \sum_{i=1}^N \text{diag}(v_i)}{\sum_{i=1}^N \tau_{ik}}$$

The mixing proportions are updated as:

$$\pi_k = \frac{\sum_{i=1}^N \tau_{ik}}{N}$$

The algorithm iterates between these two steps until convergence, typically determined by the reduction in log-likelihood between iterations, more specifically, being less than some user-specified threshold. The incorporation of  $\text{diag}(v_i)$  in the covariance matrix update ensures that the model adapts to the noise characteristics of the data.

One of the strengths of PUMA-CLUST is its explicit modelling of measurement error, which improves clustering robustness in noisy data. By incorporating probe-level variance, the model reduces the risk of spurious clusters caused by noise, especially in low-expression genes. The model's ability to model uncertainty also leads to biologically more meaningful clusters. PUMA-CLUST also outperforms traditional GMM-based methods in terms of cluster stability and biological interpretability, particularly in datasets with high variability.

Despite its strengths, PUMA-CLUST has some limitations. The inclusion of probe-level variance increases the computational complexity of the EM algorithm; thus, the model is more computationally intensive for large datasets. Additionally, the accuracy of the probe-level variance estimates ( $v_i$ ) has a great impact on the performance of the model. Variance estimates that are inaccurate or inconsistent across genes can affect the quality of the clustering results.



### 3.4.3 EMMIX-Gene

EMMIX-GENE builds on Gaussian Mixture Models (GMMs) to better cluster gene expression data from microarrays. It tackles issues like high dimensionality, noise, and variability by using gene filtering, dimension reduction, and clustering to improve results.

Microarray datasets involve thousands of genes but only a few samples, making analysis challenging due to their complexity and noise. Traditional clustering methods often struggle to handle this level of complexity effectively. EMMIX-GENE alleviates these issues by filtering irrelevant or noisy genes to reduce dimensionality, using GMMs with constrained covariance structures to improve computational efficiency and incorporating statistical tests and likelihood-based criteria to ensure the relevance of selected genes.

EMMIX-GENE operates in two key stages:

**Gene Filtering and Dimensionality Reduction:** The goal of the first stage is to identify genes with meaningful expression patterns. This is achieved by applying a likelihood ratio test (LRT) to determine the optimal number of mixture components for each gene.

Formally, for a gene  $g$ , the null hypothesis  $H_0: G = 1$  is tested against  $H_1: G = 2$  using:  $-2\log\lambda = 2[\log L(H_1) - \log L(H_0)]$ , where  $\lambda$  is the likelihood ratio statistic, and  $L(H_0)$  and  $L(H_1)$  are the likelihoods under the respective hypotheses. Genes that satisfy a prespecified threshold for  $-2\log\lambda - 2$  are retained. If  $H_0$  is not rejected, a secondary test comparing  $H_0: G = 2$  to  $H_1: G = 3$  is conducted. A gene is retained if at least two of the three clusters contain a minimum number of observations.

**Clustering Using GMMs:** The second stage applies GMMs to the filtered dataset. EMMIX-GENE uses a constrained covariance structure for efficiency, typically assuming isotropic or diagonal covariance matrices. The multivariate Gaussian density for a cluster  $k$  is:

$$p(x_i | \theta_k) = \frac{1}{(2\pi)^{p/2} |\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x_i - \mu_k)^T \Sigma_k^{-1} (x_i - \mu_k)\right)$$

where:  $x_i$  is the gene expression vector for sample  $i$ ,  $\mu_k$  is the mean vector for cluster  $k$ ,  $\Sigma_k$  is the covariance matrix for cluster  $k$ .



Dimensionality reduction techniques, such as Principal Component Analysis (PCA), may be employed to manage the high dimensionality further. By retaining only the top principal components, EMMIX-GENE simplifies the clustering process while preserving most of the variance.

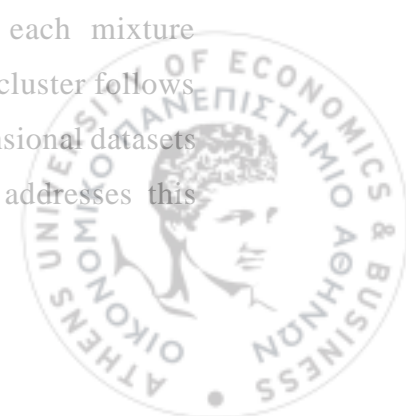
EMMIX-GENE estimates parameters using the Expectation-Maximization (EM) algorithm. The E-step generates posterior probabilities that show how likely each gene is to belong to a particular cluster. The M-step calculates model parameters such as cluster means, covariance matrices and mixing proportions to maximize the expected log-likelihood. Determining the optimal number of clusters together with the appropriate covariance structure forms the essential process of model selection in EMMIX-GENE. This is achieved using the Bayesian Information Criterion (BIC), which balances model fit and complexity.

EMMIX-GENE has demonstrated significant utility in microarray data analysis. By filtering out irrelevant genes and focusing on those with distinct patterns, EMMIX-GENE improves interpretability and the precision of the produced clustering results. Moreover, it can include dimension reduction techniques in the clustering process, that may contribute to the effective management of the high-dimensional data.

Despite its strengths, EMMIX-GENE has limitations. Filtering genes from large datasets requires significant computational resources especially when dealing with thousands of genetic sequences. The gene filtering employed in EMMIX-GENE clustering method relies on predetermined thresholds alongside restricted covariance models, which can reduce its capacity to detect intricate gene-gene interactions. Nevertheless, the ability to rank biologically important genes makes EMMIX-GENE a valuable tool for microarray data analysis.

#### 3.4.4 Factor Analytic Bayesian Mixture Models (FABMix)

FABMix (Factor Analytic Bayesian Mixture Models) is an extension of Gaussian mixture models (GMMs) whose objective is to improve clustering for high-dimensional data by incorporating latent factor models inside each mixture component. Traditional Gaussian mixture models assume that each cluster follows a multivariate normal distribution, but they struggle with high-dimensional datasets where variables are highly correlated with each other. FABMix addresses this



limitation by adding a factor analytic form in each Gaussian component to facilitate dimension reduction and improved covariance structure estimation. This makes FABMix particularly useful for microarray gene expression data, where genes often exhibit complex dependencies and high dimensionality.

The central idea behind FABMix is to model the covariance structure of each cluster using a lower-dimensional factor model rather than assuming a full covariance matrix. When the number of dimensions is large, estimating  $\Sigma_k$  directly becomes computationally expensive and prone to overfitting. FABMix overcomes this by decomposing the covariance matrix into a low-rank factor model, reducing the number of free parameters:

$$\Sigma_k = \Lambda_k \Lambda_k^T + \Psi_k$$

where:

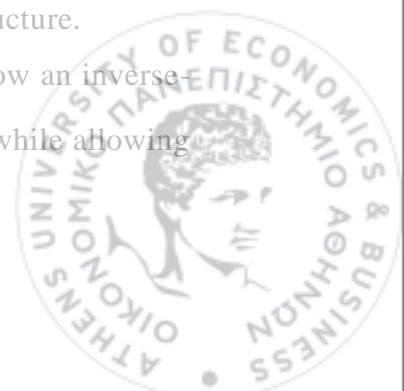
- $\Lambda_k$  is a factor loading matrix of dimension  $p \times qp$  (with  $q \ll pq$ ), capturing shared variation among the variables,
- $\Psi_k$  is a diagonal matrix representing the residual variances of each variable.

Such decomposition allows FABMix to model complex covariance structures that have fewer parameters, avoiding the overfitting issue present in high-dimensional clustering.

#### *BAYESIAN FRAMEWORK AND PRIOR DISTRIBUTIONS*

FABMix follows a Bayesian approach in model parameter estimation, including prior distributions to improve parameter estimation and prevent overfitting. The Bayesian approach introduces prior distributions for the cluster means, factor loadings, and residual variances, enabling regularization and more stable parameter estimation in high-dimensional data. The priors typically used in FABMix are:

1. Cluster Means: A normal prior is assigned to each cluster mean:  $\mu_k \sim \mathcal{N}(\mu_0, \Sigma_0)$  where  $\mu_0$  and  $\Sigma_0$  represent hyperparameters controlling the prior distribution.
2. Factor Loadings: The elements of  $\Lambda_k$  follow a sparsity-inducing prior, encouraging a parsimonious representation of the covariance structure.
3. Residual Variances: The diagonal elements of  $\Psi_k$  typically follow an inverse-gamma distribution:  $\psi_{kj} \sim IG(\alpha, \beta)$ , ensuring positive variances while allowing flexibility in the model.



By integrating these hierarchical priors, FABMix improves cluster estimation and prevents overfitting in high-dimensional microarray data.

#### *INFERENCE VIA GIBBS SAMPLING AND REVERSIBLE JUMP MCMC*

Inference in FABMix is accomplished via using Gibbs sampling and Reversible Jump Markov Chain Monte Carlo (RJ-MCMC). Gibbs sampling allows for iterative updates of parameters by sampling from their conditional distributions, making it feasible to estimate high-dimensional models efficiently. RJ-MCMC is used to determine the number of clusters dynamically by adding or removing mixture components during sampling. This adaptive clustering mechanism is particularly advantageous for microarray data, where the number of true biological clusters is typically unknown.

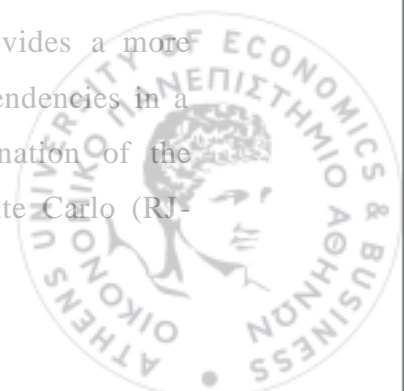
The Gibbs sampling algorithm iteratively updates:

1. Cluster assignments  $z_i$  using posterior probabilities.
2. Cluster parameters  $(\mu_k, \Lambda_k, \Psi_k)$  by drawing from their respective posterior distributions.
3. Mixing proportions  $\pi_k$  using a Dirichlet prior.

Reversible Jump MCMC further enables model selection by exploring different values of  $K$ , allowing the data to inform the number of clusters rather than requiring it to be specified in advance.

#### *ADVANTAGES OF FABMIX IN MICROARRAY DATA ANALYSIS*

FABMix is particularly suited for clustering gene expression data since it has the ability to model dependencies in high-dimensional data, avoid overfitting, and automatically determine the number of clusters. Unlike traditional Gaussian mixture models that estimate full covariance matrices, FABMix employs a factor analytic structure, which significantly reduces the number of parameters that are utilized in representing the covariance relations and hence is more computationally efficient for high-dimensional microarray data. By decomposing the covariance matrix into residual variances and factor loadings, FABMix provides a more flexible representation of gene expression patterns, modeling dependencies in a non-overparameterized way. It also includes automatic determination of the number of clusters through Reversible Jump Markov Chain Monte Carlo (RJ-



MCMC), eliminating the need to pre-specify  $K$  and allowing for more flexible exploratory data analysis. The Bayesian framework further enhances the method by including prior distributions that regularize parameter estimation, reduce the risk of overfitting, and lead to more stable and interpretable clustering results. These advantages make FABMix a powerful tool for uncovering meaningful gene expression clusters in microarray studies while addressing common issues associated with high-dimensional genomic data.

#### *LIMITATIONS AND COMPUTATIONAL CONSIDERATIONS*

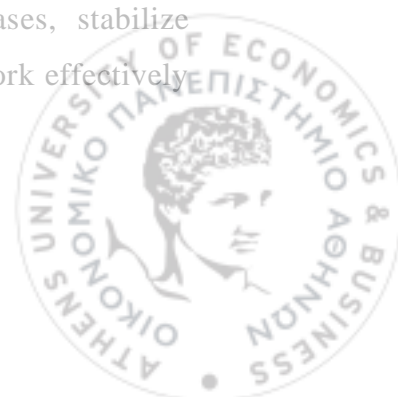
Despite its advantages, FABMix has some computational challenges. The use of Gibbs sampling and RJ-MCMC increases computational complexity compared to traditional model-based clustering methods. The iterative nature of MCMC needs careful tuning of the convergence parameters and sampling from posterior distributions of high-dimension can be computationally intensive. Advances in parallel computing and variational Bayesian approximations have, nevertheless, made it possible to render Bayesian mixture models more scalable with large genomic datasets.

Another challenge is the interpretability of factor loadings. While FABMix provides a lower-dimensional representation of the covariance structure, the biological interpretation of the resulting latent factors can be complex. In microarray studies, factor loadings can sometimes correspond to biological pathways or regulatory modules, but additional functional enrichment analyses may be required to link clusters to biological processes.

### 3.5 Preprocessing and Dimensionality Reduction

#### 3.5.1 Preprocessing

Microarray gene expression data analysis begins with preprocessing, a step that is required to prepare the data for the upcoming clustering. The raw microarray data is noisy and affected by various technical artifacts that can mask interesting biological patterns. Preprocessing is designed to normalize biases, stabilize variance, and filter the data so that the clustering algorithms can work effectively and yield biologically meaningful insights.

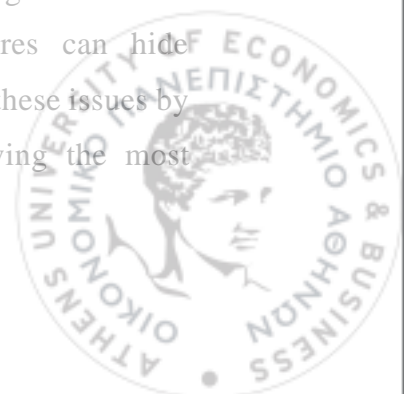


Normalization procedures form the foundation of preprocessing because they eliminate systematic technical biases and allow observed gene expression differences to reflect real biological variation. Quantile normalization is one such common procedure that adjusts the distribution of expression values for all samples, given that the majority of genes are not differentially expressed. It effectively removes hybridization and dye biases, particularly in large-scale data. For two-color microarrays, Loess normalization adjusts for intensity-dependent biases, removing variation caused by unequal dye incorporation and hybridization efficiencies. Robust Multi-array Average (RMA) is a very popular approach, specifically designed for Affymetrix microarrays. RMA combines background correction, quantile normalization, and probe-level summarization to produce robust gene-level expression values with reduced variability. All these normalization techniques form the basis, so that the clustering techniques can operate effectively with biologically meaningful data.

Variance stabilization is yet another crucial preprocessing task, which addresses the heteroscedasticity of microarray data. Variance stabilization ensures that the variance of the gene expression values remain independent of their magnitude, allowing for unbiased statistical testing. A very common method to achieve this is log-transformation. By transforming multiplicative noise into additive noise, log-transformation reduces the effect of extreme values and facilitates statistical modeling. In addition, variance-stabilizing transformations, such as those used in RMA, further transform the data, making it even more suitable for high-dimensional statistical analysis and clustering. Preprocessing ensures that the data is not only biologically relevant but also conforms to the demands of advanced clustering algorithms.

### 3.5.2 Dimensionality Reduction

Dimensionality reduction is an essential step in managing the high dimensionality of microarray data, which traditionally contains thousands of genes but relatively few samples. High dimensionality creates limitations for clustering algorithms both computational and statistical, since noise and irrelevant features can hide interesting patterns. Dimensionality reduction techniques overcome these issues by mapping the data into a lower-dimensional space while preserving the most informative features.



One of the most well-known dimensionality reduction methods is Principal Component Analysis (PCA). PCA identifies orthogonal components of maximum variance in the data, which allows it to project the data into a lower-dimensional space and eliminate noise. PCA simplifies complex high-dimensional data by retaining only those components that account for most of the variance, thereby simplifying the clustering process. PCA does assume global linearity, however, which can hinder its ability to capture complex relationships in the data. Despite this limitation, PCA remains a standard preprocessing technique in workflows such as EMMIX-GENE, where it reduces dimensionality before the fitting of Gaussian Mixture Models.

t-distributed Stochastic Neighbor Embedding (t-SNE) is a nonlinear dimensionality reduction technique that builds on PCA by preserving local structures of the data. Unlike PCA, which favors variance preservation, t-SNE favors preservation of proximity between data points in clusters. t-SNE is therefore particularly well-suited for low-dimensional representation of high-dimensional gene expression data to two or three dimensions for visualization and can detect subgroups or clusters not evident otherwise. Even though t-SNE is primarily used for visualization, the ability that it has in uncovering local structures makes clustering outcomes more interpretable.

### 3.6 Application 1 Gene Clustering

#### *DATASET DESCRIPTION*

The dataset used in this analysis is a mouse time-course microarray dataset designed for investigating gene expression dynamics during the hair growth cycle. The dataset includes 25 Affymetrix MG-U74Av2 microarray chips, capturing gene expression at eight representative time points covering both synchronous and asynchronous phases of the cycle. The first five time points (days 1, 6, 14, 17, and 23) correspond to a synchronized growth phase, while the subsequent phases (week 9, month 5, and year 1) correspond to progressively asynchronous cycles.

#### *PREPROCESSING AND NORMALIZATION*

To ensure consistency in the analysis and minimize systematic biases, a sequence of preprocessing methods was applied before clustering. These methods aimed to



refine the dataset, maintaining biologically significant patterns, and preparing the data for model-based clustering.

The dataset was first retrieved from the Gene Expression Omnibus (GEO) under the accession GDS912 and converted to an ExpressionSet format. Gene expression values were extracted into a structured matrix, which acted as the starting point for subsequent analysis. Because the dataset spans to multiple time points during the hair growth cycle, only samples that corresponded to synchronized growth phases were retained. Specifically, 1-day, 6-day, 14-day, 17-day, and 23-day samples were selected since these time-points represent a synchronized and controlled biologically cycle. This preprocessing step resulted in a cleaned dataset consisting of 15 microarray chips.

For scale-dependent effects removal and for maximizing clustering precision, z-score normalization across genes was employed. This transformation normalized expression values ensuring that the clustering algorithms focus on relative differences rather than absolute expression values. Additionally, since microarray data tends to be high-dimensional in nature, feature selection was also performed in order to retain only the most predictive genes. Differential expression among the synchronized time points was conducted with an F-test run by limma, selecting genes that have considerable variance over the time-course. The top 1,100 genes with the most significant F-statistics were selected for clustering, ensuring that the analysis focuses on genes with biologically significant expression patterns.

By carefully filtering, applying normalization, and gene selection, the preprocessing steps facilitated stable clustering analyses, to ensure that subsequent results reflect biologically meaningful patterns of gene expression dynamics.

## *CLUSTERING METHODS RESULTS*

### *PUMA*

To cluster the selected gene expression data, PUMA-CLUST was applied, a model-based clustering approach that integrates probe-level measurement uncertainty into Gaussian mixture models (GMMs). This method has been shown to improve clustering robustness for microarray datasets by explicitly accounting for technical noise and variability in gene expression measurements, as described by Liu et al. (2007).



The clustering was performed on the 1,100 most variable genes, using PUMA-CLUST with the number of clusters ranging from 2 to 35. The BIC values were computed for each clustering solution, and the optimal number of clusters was selected as the one corresponding to the minimum BIC value. The results indicated that the optimal number of clusters was 31, result that is consistent with findings from Liu et al. (2007), where similar clustering solutions were observed in their study. Their analysis demonstrated that PUMA-CLUST outperforms traditional Gaussian mixture models (e.g., MCLUST) when handling microarray data with inherent technical noise, reinforcing the effectiveness of our approach.

The BIC plot (Figure 34) shows a rapid decline in BIC values as the number of clusters increases, indicating a substantial improvement in model fit with additional clusters. However, after 31 clusters, the BIC values stabilize, suggesting that adding more clusters does not significantly enhance the model's fit. This trend reflects the elbow criterion, where the optimal number of clusters corresponds to the point at which further division of clusters yields diminishing returns in terms of statistical improvement. The red dashed line at  $K = 31$  in the plot highlights this optimal point.

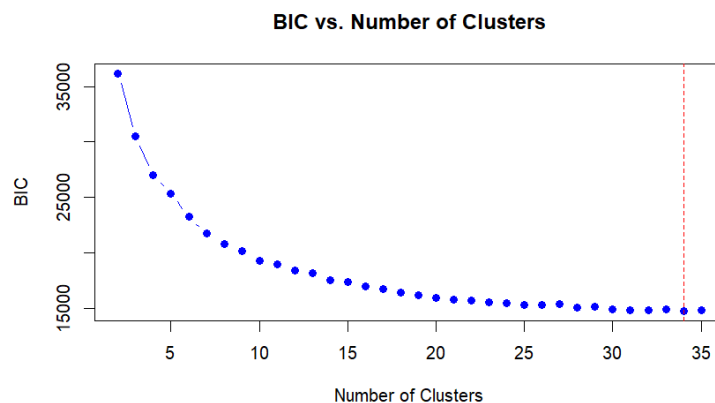
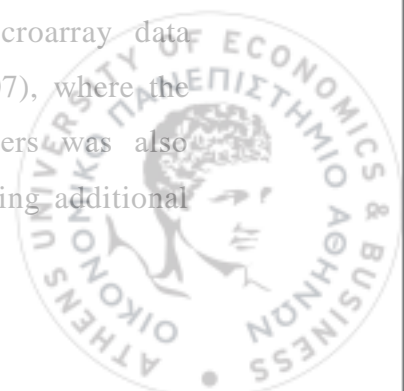


Figure 34 BIC vs. Number of Clusters plot for Puma

The final cluster assignments provided a biologically meaningful partitioning of the genes, which can be further explored through functional analysis. The results confirm the validity and reliability of PUMA-CLUST for microarray data clustering, aligning with the approach outlined in Liu et al. (2007), where the model's capability to identify biologically relevant gene clusters was also demonstrated. The next step involves evaluating these clusters using additional



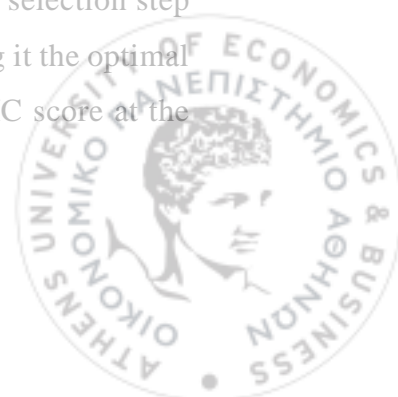
validation metrics and comparing them with other clustering methods to assess clustering stability and consistency across different approaches.

### *MCLUST*

MCLUST was applied to further investigate the clustering structure of the most significantly differentially expressed genes. This method, based on Gaussian mixture models (GMMs), employs the Expectation-Maximization (EM) algorithm to estimate cluster parameters while determining the optimal number of clusters and covariance structures through Bayesian Information Criterion (BIC) minimization. The clustering analysis was performed by testing models with 2 to 35 clusters, allowing the algorithm to explore different covariance structures and select the best-fitting model. Through iterative likelihood optimization, MCLUST assigned genes to clusters while optimizing their probability of belonging to a given group.

The optimal model, selected based on BIC, identified 32 clusters, with the VII covariance structure (spherical, varying volume) emerging as the best fit. This model assumes that clusters are spherically distributed, meaning that genes within a cluster exhibit similar variance patterns, but each cluster can have a different overall variance. This assumption restricts clusters to a fixed shape, preventing elongated or elliptical structures, in contrast to models that allow for cluster-specific shape flexibility. The final clustering resulted in well-distributed cluster sizes, ranging from 4 to 28 genes per cluster, with a log-likelihood of -722.412 and a final BIC score of -4819.356. The BIC plot (Figure 35) demonstrates a steady increase in BIC values as the number of clusters grows, eventually plateauing around 30 clusters, confirming that 32 clusters provide the best trade-off between model complexity and fit.

The BIC plot (Figure 35) shows the BIC scores across different numbers of clusters and covariance models. The BIC values decrease sharply up to approximately 10–12 clusters, followed by a more gradual decline. A noticeable stabilization occurs at around 26–30 clusters, where the BIC values start to level off. While the EVI model achieved lower BIC values at some cluster numbers, the final selection step in MCLUST determined that VII had the lowest BIC at K=32, making it the optimal model. The VII covariance model consistently achieved the best BIC score at the final selection stage, reinforcing its suitability for this dataset.



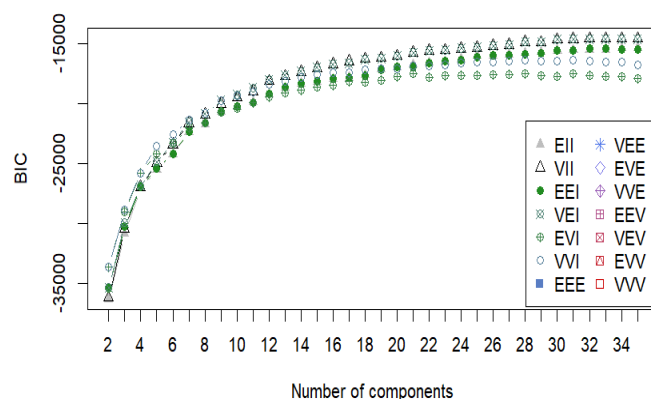
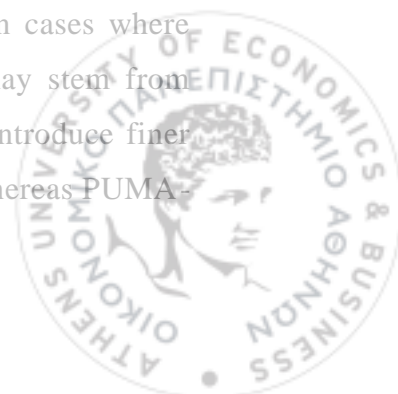


Figure 35 BIC vs. Number of Clusters plot for MClust

In the study by Liu et al. (2007), MCLUST selected the "EEE" model, which assumes a full-rank covariance matrix for each cluster, indicating that gene clusters may have different shapes, orientations, and volumes. In contrast, our analysis selected the VII model, which enforces spherical cluster shapes while allowing varying volumes. This deviation in model selection suggests that our dataset exhibits a lower complexity in covariance structure than observed in Liu et al. (2007), likely due to differences in preprocessing steps or the subset of genes analyzed. The use of z-score normalization and filtering to 1,100 genes may have reduced covariance heterogeneity, making spherical clusters more appropriate for capturing gene expression variations.

When compared to PUMA-CLUST, which identified 31 clusters, MCLUST produced a highly similar solution, reinforcing the hypothesis that the dataset contains numerous distinct expression patterns. However, key methodological differences explain subtle variations between the two clustering outcomes. PUMA-CLUST incorporates measurement error correction, which allows it to refine expression profiles and potentially split clusters more finely based on noise characteristics. In contrast, MCLUST relies purely on likelihood-based model selection using BIC, which does not explicitly account for measurement uncertainty, potentially leading to over-segmentation of clusters in cases where gene expression variability is high. The two-cluster difference may stem from MCLUST's reliance on variance-based partitioning, which could introduce finer distinctions between clusters with different expression variances, whereas PUMA-



CLUST groups genes that share underlying regulatory influences, even if their noise characteristics differ slightly.

### *PGMM*

PGMM clustering method assumes that gene expression variability can be explained by a combination of a few latent factors rather than modeling full covariance structures explicitly, which is computationally advantageous for high-dimensional datasets. We selected 1,100 of the most differentially expressed genes, allowing for a sufficient number of features to estimate both cluster structures and latent factors.

Applying PGMM across 2 to 35 clusters and 1 to 6 latent factors, we identified the optimal number of clusters as 6, with 5 latent factors, using the Bayesian Information Criterion (BIC) for model selection. The optimal model was identified as UUC, which assumes unconstrained volume and shape but constrained orientation across clusters. This model allows each cluster to have a unique covariance structure while enforcing a shared correlation structure among all clusters. The UUC model is defined as follows:

- U (Unconstrained volume): Each cluster has a different overall variance structure.
- U (Unconstrained shape): Covariance matrices can differ in terms of eigenvalues, capturing unique variance patterns.
- C (Constrained orientation): The principal axes of variance remain consistent across clusters, meaning that while clusters may vary in size and shape, their correlation structures follow the same directional patterns.

The selection of UUC suggests that clusters exhibit heterogeneous variance and shape patterns, but their correlation structures are aligned across clusters. The constraint on orientation in UUC implies that, while gene expression variability within each cluster can differ significantly, there is a consistent underlying correlation structure governing gene relationship across clusters. This assumption introduces stability to the model while still capturing meaningful biological heterogeneity.

When comparing PGMM to PUMA-CLUST and MCLUST, we observed notable differences in the number of clusters identified. PUMA-CLUST identified 31 clusters, utilizing an augmented Gaussian mixture model with measurement error correction, while MCLUST identified 32 clusters, selecting the VII model



(spherical, varying volume). The discrepancy in the number of clusters across methods can be attributed to fundamental differences in their modeling assumptions. PUMA-CLUST applies an empirical Bayes approach, explicitly accounting for measurement error, which allows for finer distinctions in gene expression patterns, often leading to a greater number of clusters as genes with similar expression variance but slightly different noise characteristics are separated. MCLUST, using the VII model, applies a diagonal covariance structure, assuming that gene expression variances are independent within each cluster. While this assumption adds flexibility, it can also over-segment the data, resulting in a higher number of clusters than necessary. In contrast, PGMM (UUC model) incorporates latent factors, effectively reducing the dimensionality of the data. By capturing only the dominant variance patterns, PGMM identified fewer clusters than PUMA-CLUST and MCLUST. The introduction of the C (constrained orientation) assumption led to a more stable clustering solution, ensuring that genes were grouped based on shared variance and shape properties while maintaining consistency in correlation structures.

Therefore, the reduced number of clusters in PGMM compared to PUMA-CLUST and MCLUST can be explained by several factors. First, dimensionality reduction plays a crucial role, as PGMM estimates cluster membership based on latent factors, which filter out noise-driven variations, leading to broader, more general gene clusters. Second, unlike the diagonal models used in MCLUST, PGMM's UUC model allows for more complex covariance structures, grouping genes that exhibit similar variance and shape patterns while maintaining a common correlation framework. Finally, there are modeling trade-offs—PUMA-CLUST explicitly corrects for noise, leading to a more fine-grained clustering solution, while MCLUST, relying purely on BIC for model selection, may over-segment clusters. PGMM balances these approaches by integrating factor analysis, ensuring that clusters remain biologically meaningful while avoiding unnecessary fragmentation.

#### *FABMIX*

To further investigate the clustering structure of the selected gene expression data, we applied fabMix, a factor-analytic mixture model specifically designed for high-dimensional data such as microarray gene expression profiles. The fabMix model integrates Gaussian Mixture Models (GMMs) with latent factor analysis, enabling



the model to capture covariance dependencies between genes while simultaneously reducing dimensionality.

The clustering was performed on the 1,100 most variable genes, using a range of 2 to 35 clusters, with latent factors ranging from 1 to 2 ( $qRange = 1:2$ ). The Markov Chain Monte Carlo (MCMC) procedure is a sampling-based approach used to estimate model parameters by iteratively drawing samples from the posterior distribution. In this analysis, we ran 500 total cycles ( $mCycles = 500$ ) to explore the parameter space, with the first 50 cycles ( $burn-in = 50$ ) discarded to remove the influence of initial conditions. Additionally, two independent chains ( $nChains = 2$ ) were used to verify that the algorithm consistently converged to the same clustering solution, reducing the risk of being trapped in local optima.

The model selection was based on the Bayesian Information Criterion (BIC), which penalizes overfitting by balancing model complexity with likelihood maximization. The optimal model was identified based on the lowest BIC value, ensuring a trade-off between model flexibility and generalizability.

After model fitting, the optimal number of clusters was determined to be 33, with the selected model type being "CCU", meaning:

- C (Common cluster covariance structure): All clusters share a common covariance structure.
- C (Common factor covariance structure): The factor covariance matrix is shared across clusters.
- U (Unconstrained error covariance): The residual error variances are freely estimated.

This model configuration suggests that gene clusters share a common underlying covariance pattern, while allowing individual genes within a cluster to exhibit variable expression noise. Additionally, the model selected two latent factors ( $q = 2$ ), meaning that the high-dimensional gene expression profiles could be effectively summarized by two latent factors.

Compared to PUMA-CLUST, which identified 31 clusters, FABMIX selected a slightly higher number of clusters (33). This increase is likely due to FABMIX's ability to model covariance structures while still incorporating latent factors, capturing finer distinctions between gene expression patterns. PUMA-CLUST, on the other hand, applies an empirical Bayes approach that explicitly accounts for measurement error, leading to finer distinctions in gene expression profiles.



Similarly, MCLUST identified 32 clusters, selecting the VII model (spherical clusters with varying volumes). The difference in the number of clusters between MCLUST and FABMIX can be attributed to their assumptions about gene covariance: MCLUST (VII model) assumes independent variance within each cluster, leading to a slightly lower cluster count, FABMIX (CCU model) allows clusters to share covariance patterns, but maintains individual noise structures, leading to more clusters overall. In contrast, PGMM identified only 6 clusters, reflecting its strong dimensionality reduction properties.

FABMIX provides a balance between fine-grained clustering (PUMA-CLUST, MCLUST) and high-level regulatory grouping (PGMM). The selection of 33 clusters with two latent factors suggests that gene expression is best explained by a moderate number of biologically relevant groups, rather than an overly large number of distinct clusters (MCLUST, PUMA-CLUST) or an overly simplified structure (PGMM). The use of MCMC inference in FABMIX enhances the robustness of the clustering solution, ensuring that the posterior distribution of clusters reflects the most stable grouping of genes.

The clustering results from MCLUST, PUMA-CLUST, and FABMIX are highly consistent, with all three methods identifying a similar number of clusters—31, 32, and 33, respectively. This agreement suggests that the underlying gene expression structure is well-defined, and despite their different modeling assumptions, the methods converge on a shared biological signal. PUMA-CLUST, by incorporating measurement error correction, distinguishes subtle variations in gene expression, leading to fine-grained clusters. MCLUST, relying on likelihood-based model selection, captures a comparable structure without explicitly modeling measurement uncertainty. FABMIX, which integrates factor analysis and covariance modeling, was expected to yield broader clusters, but instead, it produced a solution nearly identical to the others. The similarity across methods reinforces the stability of the identified clusters, indicating that the results are not artifacts of a specific approach but rather a reflection of real gene expression patterns.

#### *CLUSTERING METHODS' METRICS AND VISUALIZATION RESULTS*



Evaluating clustering performance in microarray data analysis is essential for determining the quality and stability of different clustering approaches. Since clustering is an unsupervised learning task, where the true labels are not known, a combination of internal and external validation metrics is used to assess how well-defined and biologically meaningful the clusters are. In this study, we computed several clustering evaluation metrics, including the Jaccard Similarity Index, Silhouette Score, Dunn Index, and Davies-Bouldin Index, to compare the results obtained from PUMA-CLUST, MCLUST, PGMM, and FABMIX.

The Jaccard Similarity Index measures the overlap between clustering assignments from two different methods. It is calculated as the ratio of the number of common cluster assignments (intersection) to the total number of unique assignments (union)

$$J(C_1, C_2) = \frac{A}{A + B + C},$$

where given two clustering solutions,  $C_1$  and  $C_2$ , let:  $A$  be the number of pairs of genes that are in the same cluster in both  $C_1$  and  $C_2$ ,  $B$  be the number of pairs that are in the same cluster in  $C_1$  but not in  $C_2$  and  $C$  be the number of pairs that are in the same cluster in  $C_2$  but not in  $C_1$ . Higher values indicate stronger agreement between two clustering solutions, suggesting consistency in the way genes are grouped.

The Silhouette Score is an internal clustering metric that evaluates the compactness and separation of clusters. It measures how similar each gene is to its assigned cluster compared to other clusters.

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))},$$

where for each data point  $i$ :  $a(i)$  is the average distance between  $i$  and all other points in the same cluster and  $b(i)$  is the average distance between  $i$  and all points in the nearest neighboring cluster. The score ranges from -1 to 1, with higher values indicating well-separated and dense clusters. A negative or low score suggests overlapping or poorly defined clusters.

The Dunn Index is another internal validation metric that assesses cluster compactness and separation. It is computed as the ratio of the minimum inter-cluster distance to the maximum intra-cluster distance:



$$D = \frac{\min_{1 \leq i < j \leq k} D(C_i, C_j)}{\max_{1 \leq l \leq k} D(C_l)},$$

where  $d(C_i, C_j)$  is the Euclidean distance between two clusters  $C_i$  and  $C_j$  and  $d(C_l)$  is the maximum intra-cluster distance for cluster  $C_l$ . Higher values indicate better clustering performance, as it suggests greater separation between clusters and lower within-cluster variance.

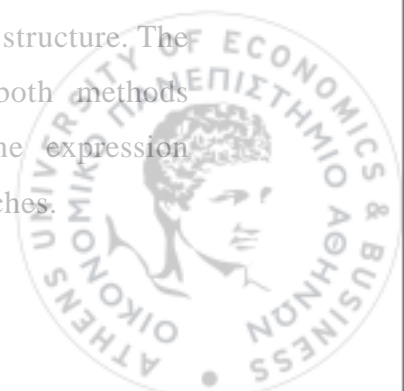
The Davies-Bouldin Index evaluates the clustering structure by measuring the average similarity between each cluster and the most similar other cluster:

$$DBI = \frac{1}{k} \sum_{i=1}^k \max_{j \neq i} \left( \frac{\sigma_i + \sigma_j}{D(C_i, C_j)} \right),$$

where:  $\sigma_i$  is the average distance between points in cluster  $C_i$  and their centroid and  $D(C_i, C_j)$  is the distance between centroids of clusters  $C_i$  and  $C_j$ . Lower values are preferred, as they indicate that clusters are well-separated from each other.

These metrics were calculated to systematically compare the clustering results obtained from the different model-based clustering approaches. By analyzing the similarities and differences across methods, we aim to determine which clustering approach provides the most stable and biologically meaningful partitioning of genes in microarray data. The results of these metrics provide insights into whether PUMA-CLUST, MCLUST, PGMM, or FABMIX best captures the underlying gene expression patterns while maintaining biological interpretability and statistical robustness.

The Jaccard similarity scores (Table 13) indicate a strong alignment between PUMA-CLUST and MCLUST (0.754), suggesting that these methods identify highly overlapping gene clusters. MCLUST and FABMIX (0.645), as well as PUMA-CLUST and FABMIX (0.62), also show considerable agreement, reflecting a shared structure in their clustering outcomes while allowing for some method-specific differences. In contrast, PGMM exhibits lower similarity scores across all comparisons, particularly with MCLUST (0.251) and PUMA-CLUST (0.277), indicating that its latent factor modeling leads to a distinct clustering structure. The PGMM vs. FABMIX similarity (0.338) suggests that while both methods incorporate covariance modeling, PGMM captures broader gene expression patterns rather than the finer partitions detected by the other approaches.



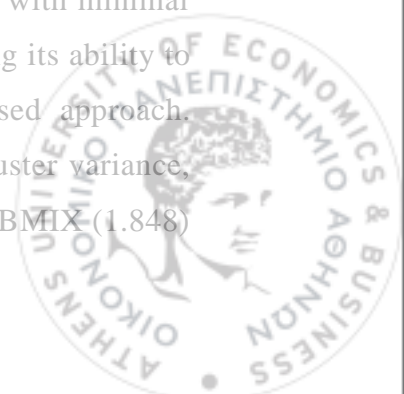
<i>Comparison</i>	<i>JACCARD SIMILARITY SCORE</i>
PUMA-CLUST vs. MCLUST	0.7536364
PUMA-CLUST vs. FABMIX	0.62
MCLUST vs. FABMIX	0.6454545
PGMM vs. PUMA-CLUST	0.2772727
PGMM vs. MCLUST	0.2509091
PGMM vs. FABMIX	0.3381818

Table 13 Jaccard Similarity Scores

The cluster validation metrics provide insights into the overall quality and distinctiveness of the identified clusters across methods. The silhouette score (Table 14), which measures how well-defined clusters are, shows that PGMM achieved the highest value (0.225), indicating relatively compact and well-separated clusters in this model. PUMA-CLUST (0.180) and MCLUST (0.132) followed, suggesting that their clusters maintain moderate separation but exhibit some level of overlap. In contrast, FABMIX obtained the lowest silhouette score (0.085), implying that its clusters may be less well-defined, potentially due to increased heterogeneity in the data partitions.

The Dunn Index (Table 14), which evaluates the ratio of the smallest inter-cluster distance to the largest intra-cluster distance, further reflects cluster distinctiveness. PUMA-CLUST (0.071) and PGMM (0.068) produced the highest values among the methods, suggesting that they maintained a reasonable degree of separation between clusters. FABMIX (0.058) and MCLUST (0.050) exhibited lower scores, implying that these methods generated clusters that were either more compact or had overlapping regions, reducing the overall inter-cluster separation.

For the Davies-Bouldin Index (DBI) (Table 14), a lower value indicates better-defined clusters with minimal intra-cluster variance. PUMA-CLUST (1.368) had the lowest DBI, suggesting that it formed the most distinct clusters with minimal dispersion within them. PGMM (1.458) followed closely, reinforcing its ability to maintain relatively compact clusters despite its latent factor-based approach. MCLUST (1.549) produced a higher DBI, indicating more intra-cluster variance, which could be attributed to its assumption of spherical clusters. FABMIX (1.848)



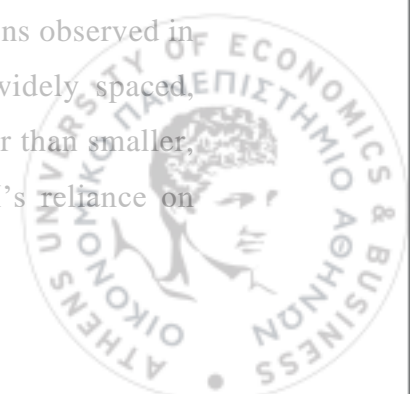
exhibited the highest DBI, suggesting that its clusters are more dispersed and potentially less well-separated than those formed by the other methods.

Taken together, these results suggest that PGMM and PUMA-CLUST performed well in terms of cluster compactness and separation, with PUMA-CLUST exhibiting the lowest intra-cluster variance. MCLUST, while still maintaining reasonable cluster quality, showed some overlap between clusters, and FABMIX appeared to form less distinct partitions, as indicated by its lower silhouette score and higher DBI. These findings highlight how different modeling approaches influence clustering performance, with PGMM leveraging latent factors for compact clusters, PUMA-CLUST optimizing for well-separated gene groups, and FABMIX balancing factor modeling with shared covariance structures but at the expense of cluster cohesion.

<i>Method</i>	<i>SILHOUETTE SCORE</i>	<i>DUNN INDEX</i>	<i>DAVIES-BOULDIN INDEX</i>
PUMA-CLUST	0.1797369	0.07117187	1.367559
MCLUST	0.1319184	0.05003558	1.54944
FABMix	0.08504217	0.05750125	1.84817
PGMM	0.2254853	0.06774834	1.45752

Table 14 Internal Metrics per Clustering Method

The PCA plots (Figure 36) give a comparative view of the clustering patterns found by PUMA-CLUST, MCLUST, FABMix, and PGMM. PUMA-CLUST and MCLUST have similar patterns of clustering, with clusters constituting well-defined groups along a curved path in the PCA space. There is some overlap, but the general cluster pattern is well defined. FABMix follows a comparable distribution, though the arrangement and number of clusters differ, reflecting its factor-analytic approach, which incorporates covariance modeling into the clustering process. In contrast, PGMM produces a markedly different clustering pattern, with only six broader clusters instead of the finer subdivisions observed in the other methods. The PGMM clusters are more distinct and widely spaced, indicating that this method captures broader expression trends rather than smaller, localized variations. This reduced granularity aligns with PGMM's reliance on



latent factor modeling, which seeks to explain variability through a limited number of underlying factors rather than directly modeling individual expression differences. The similarities among PUMA-CLUST, MCLUST, and FABMix suggest a degree of robustness in their clustering solutions, despite differences in methodological assumptions. The presence of overlapping clusters in MCLUST and FABMix may indicate shared covariance structures that are not explicitly accounted for in PUMA-CLUST. Meanwhile, the clear separation of clusters in PGMM underscores its distinct modeling approach, which prioritizes broader patterns over fine-grained differentiation.

This visual, clearly emphasizes that actual microarray analysis cannot focus on a single method of clustering, but rather, any available approach should be employed in order to identify different gene variability, as seen from the results of PUMA-CLUST, MCLUST, and FABMix which clearly map to highly resolved partitions while PGMM emphasizes major trends in expression, thus it seems particularly useful in giving broad views of regulatory patterns of gene expression.

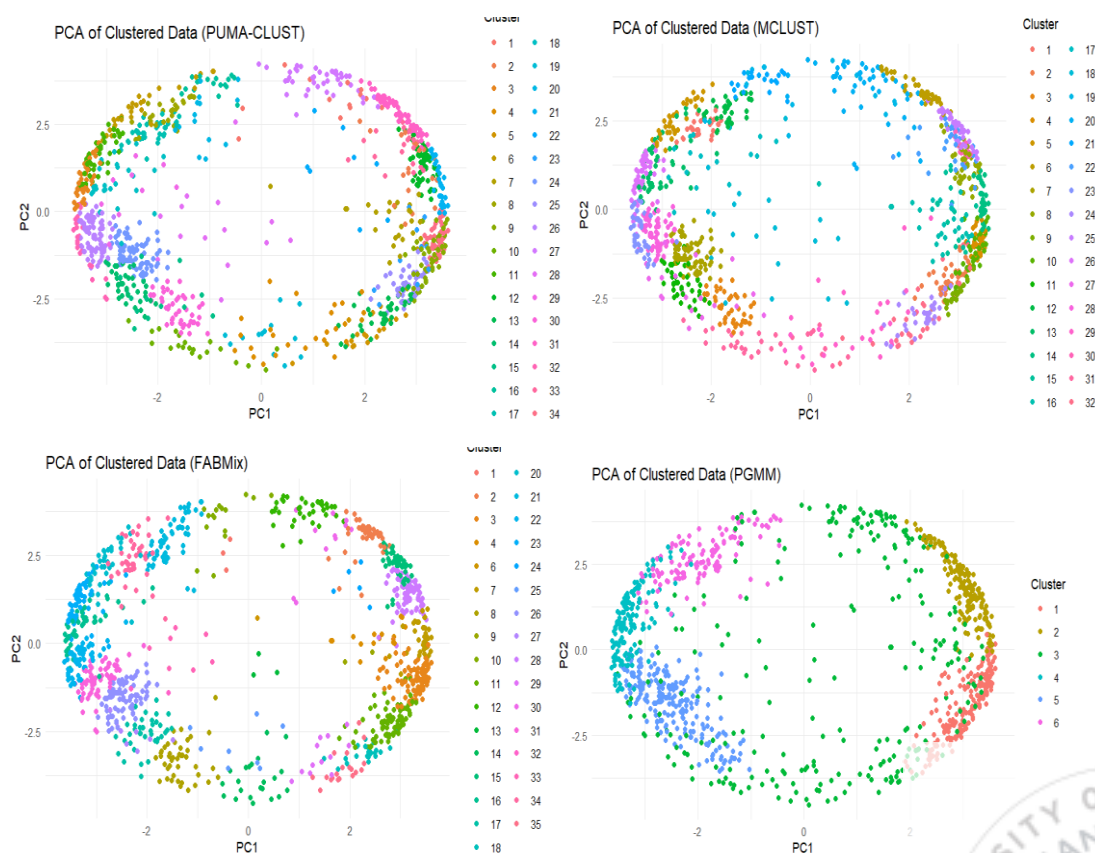


Figure 36 2D PCA of Clustered Data per method



The 3D PCA visualizations (Figure 37) provide a deeper perspective on the clustering structures identified by the different methods, offering additional insights into how well-separated and distinct the clusters are in higher-dimensional space. The distribution, compactness, and separation of clusters reflect the underlying assumptions of each method, and these visualizations are further reinforced by clustering validation metrics.

The PCA plot for PUMA-CLUST displays a globular structure, with the clusters being well-distributed but relatively dispersed and showing smooth transitions among themselves. This seems consistent with the fact that PUMA-CLUST captures small variations in gene expression measurements because it tends to group genes according to common expression profiles but can also allow some flexibility at cluster boundaries. The moderate overlap observed is consistent with its capability to account for measurement error, which refines without strictly enforcing separation between groups. The clustering validation metrics confirm this balance, as PUMA-CLUST maintains a reasonable degree of compactness and separation, ensuring biologically relevant groupings without over-fragmenting the data.

The MCLUST PCA plot displays a similar structure to PUMA-CLUST but with slightly more compact clusters. The clusters appear denser and more tightly packed, suggesting that MCLUST enforces stronger intra-cluster cohesion while maintaining clear partitions. This structure reflects MCLUST's model selection process, which prioritizes spherical clusters with varying volumes, leading to more well-defined but rigid boundaries. The metrics reinforce this observation, indicating that while MCLUST achieves strong clustering cohesion, some degree of cluster overlap remains due to its assumption of spherical distributions, which may not fully capture the complexity of gene expression relationships.

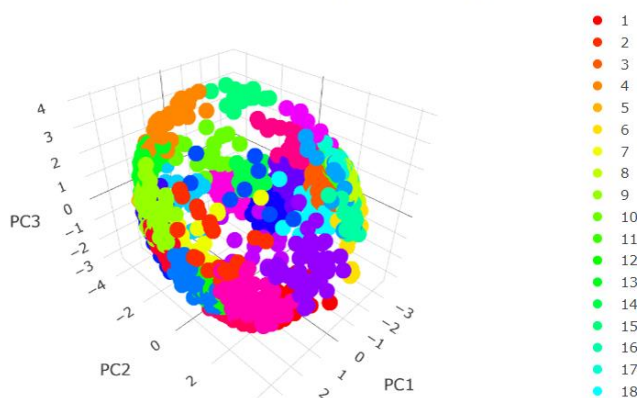
In the FABMix PCA plot, clusters are very dense but located very near to each other, which means there is very little gap between them. This implies that FABMix is more interested in intra-cluster similarity than global separation and is clustering genes based on patterns of covariance. The absence of huge gaps between clusters means that some clusters might be picking up overlapping regulatory processes and hence are densely packed but not clearly separated. This behavior is supported by validation metrics, which show high intra-cluster cohesion but weaker separation.



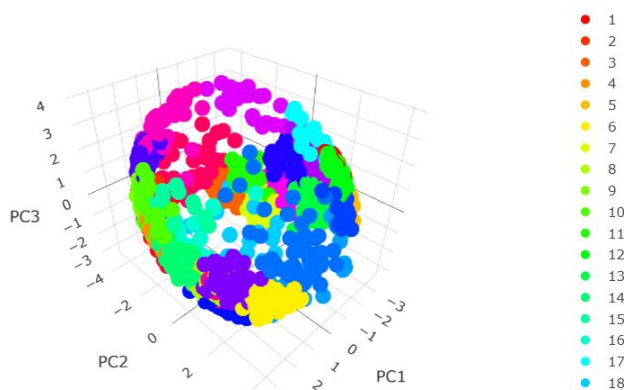
suggesting that while FABMix effectively models covariance relationships, its clusters remain in close proximity, reducing their distinctiveness.

The PGMM PCA plot is quite different from the other approaches, with only six general clusters that are well separated in space. The absence of fine partitions indicates that PGMM clusters genes according to common latent factors instead of expression similarities, leading to more generalized, larger clusters. This is consistent with PGMM's modeling approach, which reduces dimensionality by identifying dominant regulatory patterns instead of minor expression differences. The validation metrics support this trend, with PGMM providing good separation among clusters but potentially at the cost of resolution, clustering genes into larger categories instead of splitting finer subgroups.

3D PCA Plot of Clustered Data (PUMA-CLUST)



3D PCA Plot of Clustered Data (MCLUST)



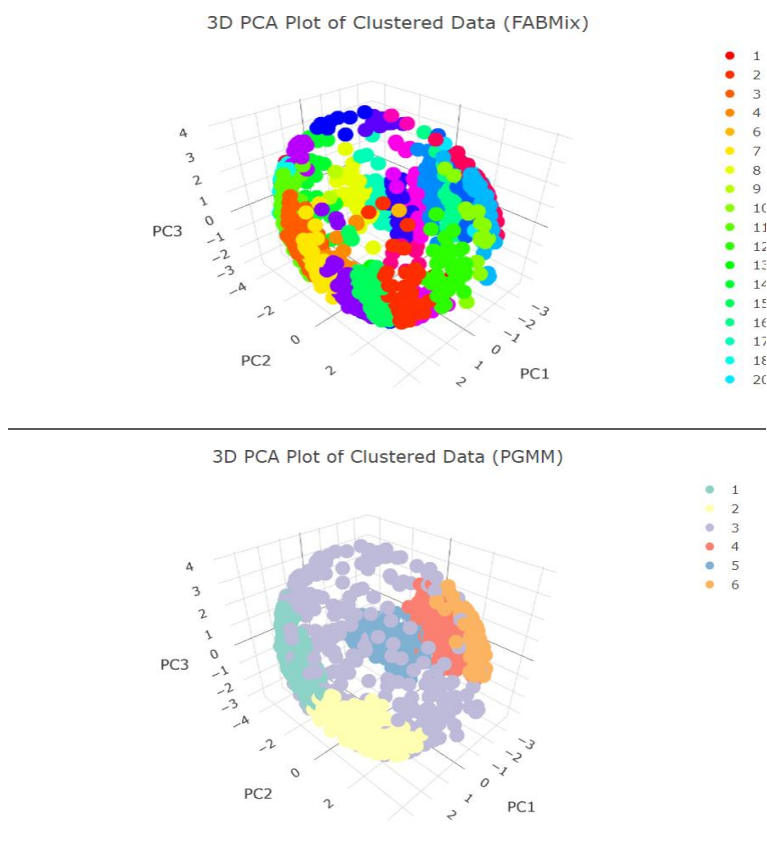


Figure 37 3D PCA of Clustered Data per method

## Conclusion

The comparative analysis of clustering methods in microarray data has revealed key differences in cluster structure, granularity, and stability across model-based approaches on the mouse dataset. PUMA-CLUST, MCLUST, and FABMix effectively captured fine-grained cluster structures, differentiating gene expression patterns while balancing cohesion and separation, though each method modeled covariance differently, leading to variations in cluster stability. PGMM, on the other hand, gave a more concise clustering result with larger clusters that yielded a stable partitioning but may have ignored more detailed biological substructures. The assessment metrics such as Jaccard Similarity, Silhouette Score, Dunn Index, and Davies-Bouldin Index reflected the compromise between compactness and separation, with FABMix having satisfactory intra-cluster cohesion but exhibiting tighter cluster proximity. The PCA visualizations give an additional view on how methods may structure the data in low-dimensional space and thus support the observed differences in clustering behavior. In summary, this analysis emphasizes that the choice of clustering method should be selected based on study objectives,



balancing interpretability, biological relevance, and robustness of the model. By applying various approaches to clustering in parallel, it will increase confidence in the clustering outcomes.

### 3.7 Application 2 Sample/Tissue Clustering

#### *DATASET DESCRIPTION*

The second application in this study focuses on clustering analysis using the leukemia dataset, a widely utilized microarray dataset for gene expression analysis. This dataset consists of gene expression patterns from bone marrow samples of individuals with two well-known leukemia subtypes: Acute Lymphoblastic Leukemia (ALL) and Acute Myeloid Leukemia (AML). The data set includes 72 samples, 47 from ALL patients and 25 from AML patients, and gene expression is measured on 3,731 genes. The primary goal in this analysis is to examine whether unsupervised clustering techniques can effectively distinguish between these two subtypes based solely on gene expression patterns.

#### *PREPROCESSING*

The leukemia dataset used in this analysis originates from Golub et al. (1999) and has been preprocessed already following the approach described by McLachlan et al. (2002). The data contains normalized gene expression values, and no normalization or transformation was necessary before clustering. The expression values are log-transformed, centered, and standardized, and hence differences in scale will not affect the clustering analysis.

While the dataset was pre-normalized, an important preprocessing step was the selection of a subset of the most informative genes to reduce noise and improve clustering performance. To this end, the limma package was utilized for differential expression analysis to identify the genes most differentially expressed between Acute Lymphoblastic Leukemia (ALL) and Acute Myeloid Leukemia (AML). A linear model was fitted to the data with a design matrix coding the two leukemia subtypes using an empirical Bayes approach to compute moderated t-statistics. The genes were ordered by adjusted p-values based on the Benjamini-Hochberg (BH) correction, and the top 1,000 differentially expressed genes were selected for further analysis.



After selecting the most informative genes, the dataset was subsetted to retain just these top 1,000 genes, reducing the feature space of 3,731 genes to 1,000. This dimensionality reduction decreases the effect of noise and redundant features so that clustering algorithms can function better. The final preprocessed dataset was transposed so that the samples were represented by rows and genes by columns. By choosing the most informative genes without compromising the integrity of the original normalization, this preprocessing method enables the clustering methods to operate on a biologically relevant and computationally manageable subset of the data. This approach improves the likelihood of identifying meaningful clusters that reflect the underlying biological differences between leukemia subtypes.

### *CLUSTERING METHODS*

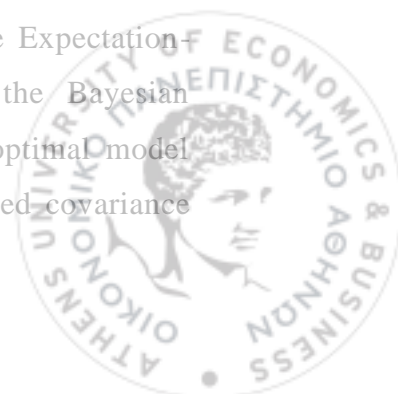
This analysis focuses on tissue (sample) clustering, where the aim is to cluster leukemia samples according to their gene expression profiles and discover whether unsupervised methods can discover the two existing subtypes: Acute Lymphoblastic Leukemia (ALL) and Acute Myeloid Leukemia (AML).

To address this challenge, a range of clustering methods were applied, mostly model-based clustering methods. Among the model-based methods, Parsimonious Gaussian Mixture Models (PGMM), MCLUST, PUMA-CLUST, and EMMIX-GENE were applied to see whether probabilistic modeling can effectively discover distinct leukemia subtypes. These methods employ statistical models and methods such as Gaussian Mixture Models (GMMs) and Expectation-Maximization (EM) algorithms to determine the optimal number of clusters while incorporating model selection criteria such as the Bayesian Information Criterion (BIC).

## **Results**

### *PGMM*

Using Parsimonious Gaussian Mixture Models (PGMM), the leukemia dataset was clustered to identify distinct leukemia subtypes. The pgmmEM algorithm was applied to the preprocessed data, with the number of clusters set to vary between 2 and 4 and the number of latent factors ranging from 1 to 4. The Expectation-Maximization (EM) procedure was initialized randomly, and the Bayesian Information Criterion (BIC) was used for model selection. The optimal model selected was "UCUU", meaning unrestricted means and constrained covariance



structures. The algorithm determined that the best cluster configuration consists of two clusters ( $g = 2$ ), aligning with the known ALL and AML subtypes, while the optimal number of factors was  $q = 1$ , indicating that a single latent factor was sufficient to explain the clustering structure. While a single latent factor might suggest a relatively simple underlying structure in the data, this result is not necessarily problematic. PGMM models assume that high-dimensional data can be represented in a lower-dimensional latent space, and in this case, one factor may be sufficient to capture the primary variance distinguishing leukemia subtypes. However, the final clustering assignments were highly imbalanced, with 69 samples grouped into one cluster and only 3 samples in the other. This strong cluster dominance suggests that the model struggled to separate the leukemia subtypes effectively. Further validation through cluster stability metrics and visualization is necessary to confirm whether the model effectively separates the biological groups.

#### *MCLUST*

The MCLUST algorithm was applied to the leukemia dataset to identify clusters using model-based Gaussian Mixture Modeling (GMM). The Expectation-Maximization (EM) algorithm was used to determine the optimal number of clusters and the best covariance structure based on Bayesian Information Criterion (BIC). The best-selected model was "VII", which assumes spherical clusters with varying volume. Unlike PGMM, which found two clusters, MCLUST determined that the optimal number of clusters was four ( $g = 4$ ), suggesting a more complex underlying structure. The cluster distribution showed 29, 8, 11, and 24 samples assigned to different groups, indicating a significant deviation from the expected two-group classification (ALL vs. AML). While a higher number of clusters may indicate potential substructures within the leukemia subtypes, it could also reflect overfitting due to the high dimensionality of the data. The ICL and BIC values were nearly identical, indicating stable model selection.

#### *PUMA-CLUST*

The PUMA-CLUST algorithm was applied to the leukemia dataset to cluster samples while accounting for probe-level measurement uncertainty, a critical factor in microarray data analysis. The model selection process was guided by Bayesian Information Criterion (BIC), where multiple clustering solutions (ranging from 2 to 5 clusters) were evaluated. The optimal number of clusters was determined to be



three ( $g = 3$ ), indicating partial alignment with known leukemia subtypes but introducing an additional subgroup. The final clustering assignment resulted in 18, 24, and 30 samples per cluster, indicating a non-uniform distribution of samples across clusters. Given that PUMA-CLUST integrates measurement variability into its clustering framework, this result may capture underlying heterogeneity within leukemia subtypes or potential technical artifacts.

### *EMMIX-GENE*

The EMMIX-GENE approach was applied to the leukemia dataset to perform a two-step clustering process, first identifying biologically relevant genes and then clustering tissue samples based on their expression profiles. The method begins by selecting a subset of highly variable genes from the dataset using the `select_genes()` function, which ensures that only the most informative genes contribute to the clustering process. The number of selected genes (578) was determined based on variability filtering, improving the robustness of the clustering step.

After gene selection, the `cluster_genes()` function was used to partition the selected genes into two groups, representing distinct biological expression patterns. These gene clusters were then leveraged for tissue (sample) clustering, where `cluster_tissues()` assigned leukemia samples into two distinct clusters ( $G = 2$ ) using a model-based framework incorporating latent factors ( $q = 2$ ). The final clustering assignments were highly imbalanced, with 13 samples in Cluster 1 and 59 samples in Cluster 2, suggesting that the model primarily grouped most samples together while a smaller subset formed a distinct but unclear cluster. While this result aligns with the known binary classification of ALL and AML, the imbalance in cluster sizes raises the possibility that one subtype exhibits higher heterogeneity or that certain samples may not fit cleanly into the expected classification.

### *CLUSTERING METHODS' METRICS AND VISUALIZATION RESULTS*

To assess the performance of the clustering methods applied to the leukemia dataset, the Adjusted Rand Index (ARI) was computed for each approach:

$$ARI = \frac{RI - E(RI)}{\max(RI) - E(RI)},$$

where  $RI = \frac{a + b}{a + b + c + d}$  is the Rand Index,  $a$  is the number of pairs of samples that are in the same cluster in both C (predicted) and T (true),  $b$  is the number of



pairs of samples that are in different clusters in both C and T,  $c$  is the number of pairs that are in the same cluster in T but in different clusters in C and  $d$  is the number of pairs that are in the same cluster in C but in different clusters in T. ARI quantifies how well the identified clusters align with the true labels of the leukemia subtypes (ALL and AML). A higher ARI indicates stronger agreement with the known classification, while values close to zero suggest that clustering results are largely random. Negative ARI values imply worse-than-random clustering performance.

The model-based clustering methods yielded mixed results (Table 15). PUMA-CLUST and Mclust performed relatively well (ARI = 0.562, 0.491 respectively). The ARI score (0.562) of Puma was the highest among all methods tested, indicating a moderate level of separation between leukemia subtypes. The ARI score of Mclust suggests moderate alignment with the actual leukemia subtypes, but the over-segmentation indicates potential overfitting. However, PGMM (ARI = 0.077) and tissue clustering using EMMIX-GENE (ARI = 0.069) failed to generate meaningful clusters, with results close to random assignment. The poor performance of PGMM, despite its model flexibility, may be due to the high dimensionality of the data and the assumption that a single latent factor was sufficient to capture the structure of the leukemia subtypes. Similarly, the EMMIX-GENE approach, while effective for gene clustering, may not have successfully distinguished between tissue samples in this dataset.

<i>Method</i>	<i>ARI Index</i>
PGMM	0.077
Mclust Clustering	0.491
PUMA Clustering	0.562
Emmix Clustering	0.069

Table 15 Adjusted Rand Index (ARI) per method

Overall, some model-based approaches struggled to separate the leukemia subtypes effectively. However, further investigation with the assistance of the visualization techniques (PCA, t-SNE), and other clustering quality measures (e.g., silhouette coefficients, Davies-Bouldin index) must be conducted to validate such findings and assess the stability of the clustering techniques.

To evaluate the quality of the resultant clusters, several internal validation measures were computed like Silhouette Score, Davies-Bouldin Index (DBI) and Dunn Index.



These measures reflect the compactness, separation, and structure of clusters, allowing for a comparison between different clustering methods.

The Silhouette Score measures how well each sample is clustered by comparing intra-cluster cohesion and inter-cluster separation (Table 16). Higher values indicate well-defined clusters. Among the methods applied, PGMM (0.140) and MCLUST (0.104) performed satisfactorily, having the highest Silhouette Scores, indicating that among all methods their clusters may be the most distinct. The lowest Silhouette Score was observed for EMMIX-GENE (0.023), suggesting that the identified clusters in this model were poorly defined and highly overlapping.

The Davies-Bouldin Index (DBI) evaluates the ratio of intra-cluster dispersion to inter-cluster distance, with lower values indicating better clustering performance (Table 16). The worst-performing methods in terms of DBI was MCLUST (2.225). PUMA (2.535), and EMMIX-GENE (4.847) which had the highest DBI values, indicating that these methods produced clusters that were highly dispersed and lacked clear separation.

The Dunn Index measures the ratio of the smallest inter-cluster distance to the largest intra-cluster distance, where higher values indicate better clustering (Table 16). The highest Dunn Index was observed for PGMM (0.526) and MCLUST (0.556), while EMMIX-GENE (0.370) had the lowest value, confirming that its clustering results were poorly separated.

<i>Method</i>	<b>Silhouette Score</b>	<b>DB Index</b>	<b>Dunn Index</b>
PGMM	0.13954345	1.4518112	0.5260461
Mclust Clustering	0.10393583	2.2246834	0.5555132
PUMA Clustering	0.10072336	2.5353047	0.5539642
Emmix Clustering	0.02280419	4.8466939	0.3700892

Table 16 Internal Clustering Metrics per method

The results indicate that PGMM and MCLUST exhibited moderate performance, indicating some degree of separation but weaker cluster structures. PUMA-CLUST showed poor performance in Silhouette and DBI but had a moderate Dunn Index, suggesting that it might be capturing some biological signal despite weaker overall clustering. EMMIX-GENE consistently performed the worst across all metrics, indicating that it failed to form meaningful clusters for tissue samples.

The visualization results from PGMM, MCLUST, PUMA-CLUST, and EMMIX-GENE reveal notable differences in how these probabilistic clustering approaches



assign leukemia samples into clusters (Figure 38). These methods rely on latent variable models, Gaussian mixture models (GMMs), and Bayesian inference, which theoretically should provide more flexibility in capturing the data structure compared to distance-based approaches. However, their effectiveness varies significantly, as seen in the PCA and t-SNE projections. A key factor influencing the poor performance of these model-based methods is the high-dimensional nature of the dataset, where the number of features (genes) far exceeds the number of observations (samples). This imbalance creates computational and statistical challenges for these approaches, as they rely on estimating complex covariance structures, which become unstable when the number of features is much larger than the number of observations.

For PGMM, the clustering fails to effectively separate the leukemia subtypes, with a significant number of red and blue points (true labels) overlapping in both PCA and t-SNE plots. This aligns with the low ARI score observed for PGMM, indicating that the method struggled to capture the underlying distribution of leukemia samples. Most samples were assigned to a single cluster, reinforcing the poor performance of this model. Given the dataset's high dimensionality, PGMM's assumption that clusters can be captured using a small number of latent factors might be insufficient, resulting in weak separation.

The MCLUST results suggest a more complex clustering structure, as the model identified four distinct clusters instead of two. While some subgroups align with the true labels, over-clustering is evident, likely due to the BIC-driven model selection choosing a higher number of clusters than expected. The additional clusters introduce significant misclassification, making MCLUST less effective in this context despite its strong theoretical foundation. This over-segmentation could be a consequence of high feature dimensionality, where the model detects artificial structure in the noise rather than meaningful biological clusters. Despite this, MCLUST outperformed PGMM and EMMIX-GENE in separating leukemia subtypes, though its inability to consistently differentiate the two main groups suggests limitations in its application to this dataset. PUMA-CLUST identified three clusters, suggesting some separation of leukemia subtypes, but the additional cluster adds complexity to the interpretation. The introduction of an additional cluster suggests that PUMA-CLUST may have captured biological heterogeneity within the dataset, but further validation would be needed to confirm whether this cluster reflects a meaningful biological subgroup or a modeling artifact. The PCA and t-SNE plots indicate that some AML samples were split into



multiple clusters, which may suggest biological heterogeneity but also introduces uncertainty regarding the validity of the identified groups. This aligns with the moderate ARI score observed for PUMA-CLUST, where it performed better than PGMM. The observed difficulty in defining well-separated clusters could be due to noise and variability in gene expression data, which is exacerbated by the dataset's high-dimensional nature. Given the lower silhouette and higher Davies-Bouldin Index scores, it is likely that the clustering structure was not well-defined, with overlapping group boundaries. Nevertheless, the integration of probe-level uncertainty appears to provide an advantage over other model-based approaches.

Finally, EMMIX-GENE produced a clustering solution similar to PGMM, showing substantial overlap between the two leukemia subtypes. The PCA and t-SNE projections indicate that clusters do not align well with the true labels, reinforcing the low ARI and internal validation scores obtained for this method. A possible explanation is that the selected features by EMMIX-GENE may not have provided sufficient discriminatory power for subtype separation. The high-dimensional nature of the dataset likely exacerbated these limitations, leading to cluster merging and a lack of meaningful differentiation between leukemia subtypes.

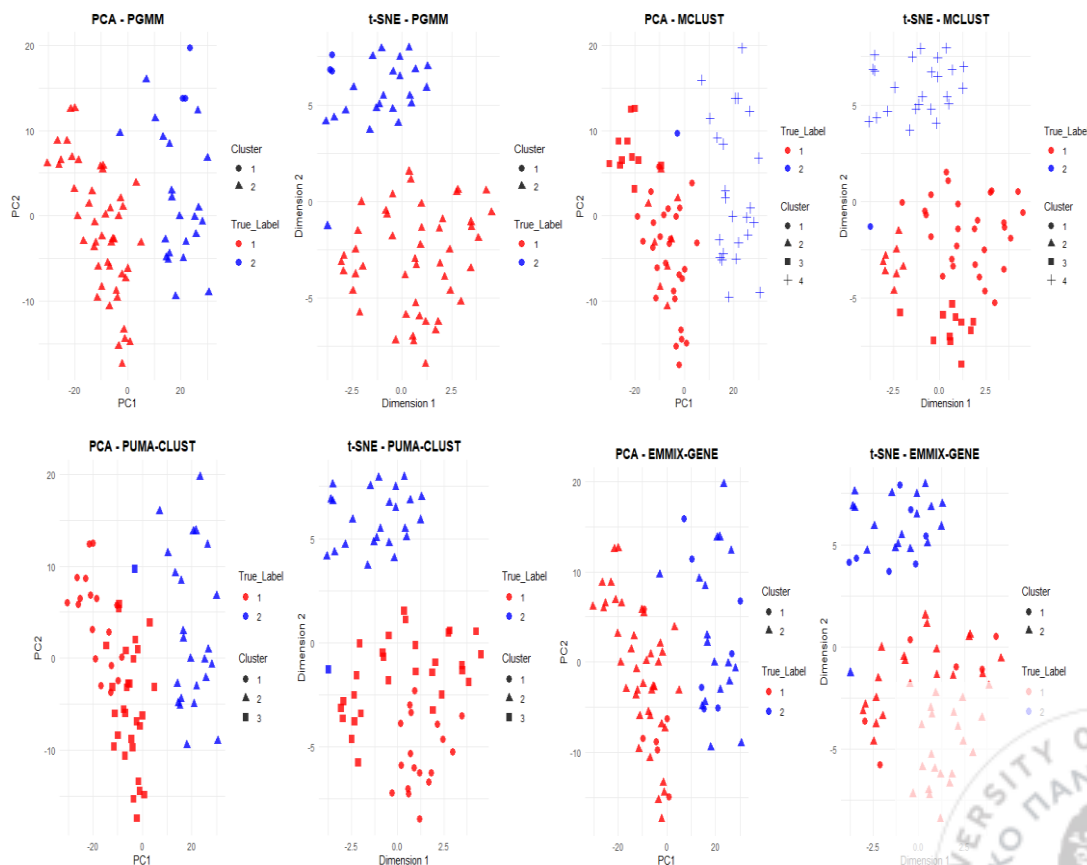


Figure 38 PCA and t-SNE visualization plot for Model based Clustering methods

### **Conclusion**

The clustering analysis of the leukemia dataset highlights the challenges of applying model-based clustering techniques to high-dimensional microarray data. While PUMA-CLUST and MCLUST demonstrated moderate success in separating leukemia subtypes, PGMM and EMMIX-GENE failed to generate meaningful clusters. The high-dimensional nature of the dataset, combined with the need to estimate complex covariance structures, posed significant challenges for model-based approaches, leading to cluster merging, over-segmentation, or poor differentiation.

MCLUST over-clustered the data, identifying four groups rather than two, likely due to its reliance on BIC-driven model selection. PUMA-CLUST showed the best separation but introduced an additional subgroup, the biological relevance of which remains unclear. These findings suggest that for tissue/sample clustering in high-dimensional microarray data, some probabilistic model-based approaches struggled more than others, under the specific conditions of the dataset. Future work should explore dimensionality reduction techniques (e.g., PCA, feature selection) or hybrid clustering strategies to improve model-based clustering performance. The results emphasize the need for careful preprocessing, selection of appropriate clustering methods, and validation using multiple metrics to ensure meaningful biological insights in high-dimensional gene expression data.



## APPENDIX

### Tables

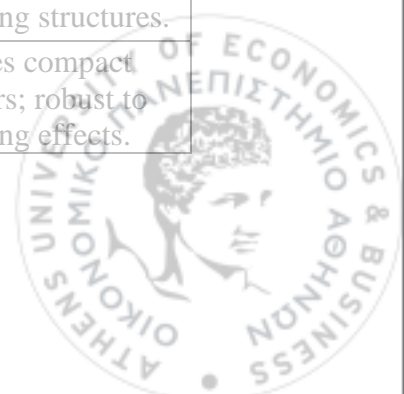
<i>Method</i>	<i>Formula</i>	<i>Primary Use</i>
<b>Euclidean Distance</b>	$d(x_i, x_j) = \sqrt{\sum_{k=1}^p (x_{ik} - x_{jk})^2}$	Measures straight-line distance; standard for continuous data clustering.
<b>Squared Euclidean</b>	$d(x_i, x_j) = \sum_{k=1}^p (x_{ik} - x_{jk})^2$	Computationally efficient variant used in k-means; minimizes within-cluster variance.
<b>Manhattan Distance</b>	$d(x_i, x_j) = \sum_{k=1}^p  x_{ik} - x_{jk} $	Emphasizes absolute differences; robust to outliers.
<b>Correlation-Based</b>	$d(x_i, x_j) = 1 - \text{corr}(x_i - x_j)$ where $\text{corr}(x_i - x_j)$ is Pearson correlation.	Captures co-expression relationships in gene expression data.
<b>Cosine Similarity</b>	$\text{cosine\_sim}(x_i, x_j) = \frac{\sum_{k=1}^p x_{ik} x_{jk}}{\sqrt{\sum_{k=1}^p (x_{ik})^2} \sqrt{\sum_{k=1}^p (x_{jk})^2}}$	Measures angular similarity; useful for expression patterns and directional trends.
<b>Mahalanobis Distance</b>	$d(x_i, x_j) = \sqrt{(x_i - x_j)^T S^{-1} (x_i - x_j)}$ where $S$ is the covariance matrix.	Adjusts for variable correlations; ideal for



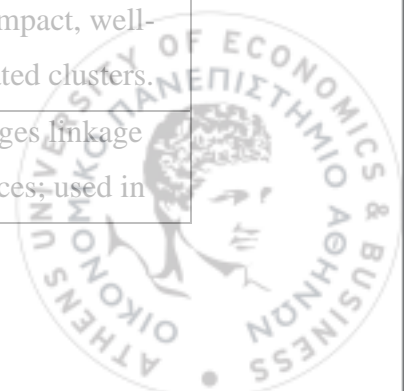
		multivariate datasets.
<b>Canberra Distance</b>	$d(x_i, x_j) = \sum_{k=1}^p \frac{ x_{ik} - x_{jk} }{ x_{ik}  +  x_{jk} }$	Highlights proportional differences; effective for datasets with variable scales.
<b>Dynamic Time Warping (DTW)</b>	N/A (Algorithm-based, aligns time-series data points dynamically.)	Aligns temporal patterns; used in time-series gene expression data.
<b>Jaccard Distance</b>	$d(x_i, x_j) = 1 - \frac{ x_i \cap x_j }{ x_i \cup x_j }$	Measures dissimilarity between binary attributes or categorical data.
<b>Mutual Information-Based</b>	N/A (Entropy-based, measures dependence between variables.)	Captures non-linear dependencies; useful for complex relationships in gene expression.

Table 17 Distance Metrics for clustering

<i>Method</i>	<b>Formula</b>	<i>Primary Use</i>
<b>Single Linkage</b>	$d(A, B) = \min_{x \in A, y \in B} d(x, y)$	Forms elongated clusters; suitable for detecting chaining structures.
<b>Complete Linkage</b>	$d(A, B) = \max_{x \in A, y \in B} d(x, y)$	Creates compact clusters; robust to chaining effects.



<p><b>Average Linkage</b></p>	$d(A,B) = \frac{1}{ A  B } \sum_{x \in A} \sum_{y \in B} d(x,y)$ <p>where <math> A </math> and <math> B </math> are the number of points in clusters <math>A</math> and <math>B</math></p>	<p>Balances compactness and chaining effects;                  Suitable for datasets where clusters have similar sizes and are relatively well-separated; Often used in gene expression analysis to cluster genes with balanced expression levels.</p>
<p><b>Ward's Linkage</b></p>	$d(A,B) = \frac{ A  B }{ A  +  B } \left\  \bar{x}_A - \bar{x}_B \right\ ^2$ <p>where <math>\bar{x}_A</math> and <math>\bar{x}_B</math> are the centroids of clusters <math>A</math> and <math>B</math>, and <math>\left\  \bar{x}_A - \bar{x}_B \right\ </math> represents the Euclidean distance between them</p>	<p>Minimizes within-cluster variance; Preferred for continuous datasets such as gene expression profiles; Effective for creating compact, spherical clusters and handling datasets with variable cluster sizes.</p>
<p><b>Centroid Linkage</b></p>	$d(A,B) = \left\  \bar{x}_A - \bar{x}_B \right\ $ <p>where <math>\bar{x}_A</math> and <math>\bar{x}_B</math> are the centroids of clusters <math>A</math> and <math>B</math></p>	<p>Links clusters based on their centroids; useful for compact, well-separated clusters.</p>
<p><b>Median Linkage</b></p>	$d(A,B) = \frac{\left\  \bar{x}_A - \bar{x}_B \right\  + \left\  \bar{x}_A - \bar{x}_C \right\  + \left\  \bar{x}_B - \bar{x}_C \right\ }{3}$	<p>Averages linkage distances; used in</p>



		balanced datasets with no extreme variability.
--	--	--

Table 18 Linkage methods for clustering

### ***Code***

#### **Application Chapter 2**

#Step 1: Load Required Libraries

```
library("leukemiasEset")
```

```
library(qvalue)
```

```
library(ggplot2)
```

```
library(fdrtool)
```

```
library(UpSetR)
```

# Step 2: Load the Leukemia Dataset (leukemiasEset)

```
data(leukemiasEset)
```

# Extract expression data and phenotype data

```
exprs_data <- exprs(leukemiasEset)
```

```
pheno_data <- pData(leukemiasEset)
```

# Subset for AML vs CML samples

```
aml_cml_samples <- which(pheno_data$LeukemiaType %in% c("AML", "CML"))
```

```
exprs_data_subset <- exprs_data[, aml_cml_samples]
```

```
group <- factor(pheno_data$LeukemiaType[aml_cml_samples])
```

# Compute mean expression per sample

```
mean_expression_per_sample <- colMeans(exprs_data_subset)
```

# Create dataframe with sample-wise mean expression

```
df_mean_expr <- data.frame(
```

```
  Sample = colnames(exprs_data_subset),
```

```
  MeanExpression = mean_expression_per_sample,
```

```
  LeukemiaType = group # AML or CML
```

```
)
```

# Boxplot of mean expression per group



```
ggplot(df_mean_expr, aes(x = LeukemiaType, y = MeanExpression, fill =  
LeukemiaType)) +  
  geom_boxplot(alpha = 0.7) +  
  theme_minimal() +  
  labs(title = "Mean Gene Expression per Sample (AML vs. CML)",  
        x = "Leukemia Type",  
        y = "Mean Expression Level") +  
  scale_fill_manual(values = c("AML" = "blue", "CML" = "red")) +  
  theme(legend.position = "none")  
# Compute mean expression per gene for AML and CML separately  
mean_expression_aml <- rowMeans(exprs_data_subset[, group == "AML"])  
mean_expression_cml <- rowMeans(exprs_data_subset[, group == "CML"])  
# Create dataframe for scatter plot  
df_gene_means <- data.frame(  
  GeneID = rownames(exprs_data_subset),  
  Mean_AML = mean_expression_aml,  
  Mean_CML = mean_expression_cml  
)  
# Scatter plot comparing AML vs. CML mean expression per gene  
ggplot(df_gene_means, aes(x = Mean_AML, y = Mean_CML)) +  
  geom_point(alpha = 0.5, color = "darkblue") +  
  theme_minimal() +  
  labs(title = "Mean Gene Expression: AML vs. CML",  
        x = "Mean Expression in AML",  
        y = "Mean Expression in CML") +  
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") #  
Reference line  
# Step 3: Perform Traditional t-tests for Each Gene  
p_values <- apply(exprs_data_subset, 1, function(gene_expression) {  
  t.test(gene_expression ~ group)$p.value  
})  
# Create results data frame  
results <- data.frame(GeneID = rownames(exprs_data_subset), P.Value =  
p_values)
```



```
# Step 4: Apply Multiple Hypothesis Testing Methods
# FWER Methods
results$Bonferroni <- p.adjust(results$P.Value, method = "bonferroni")
results$Holm <- p.adjust(results$P.Value, method = "holm")
results$Hochberg <- p.adjust(results$P.Value, method = "hochberg")
# FDR Methods
results$BH <- p.adjust(results$P.Value, method = "BH")
# Adaptive FDR (Storey's q-value)
qvals <- qvalue(results$P.Value)$qvalues
results$qvalue <- qvals
# Local FDR
t_statistics <- apply(exprs_data_subset, 1, function(gene_expression) {
  t.test(gene_expression ~ group)$statistic
})
fdrtool_results <- fdrtool(t_statistics, statistic = "normal", plot = TRUE)
results$local_fdr <- fdrtool_results$lfd

# Step 5: Compare Results
# Count significant genes by method (alpha = 0.05)
num_sig_bonferroni <- sum(results$Bonferroni < 0.05)
num_sig_holm <- sum(results$Holm < 0.05)
num_sig_hochberg <- sum(results$Hochberg < 0.05)
num_sig_bh <- sum(results$BH < 0.05)
num_sig_qvalue <- sum(results$qvalue < 0.05)
num_sig_local_fdr <- sum(results$local_fdr < 0.05)
cat("Bonferroni: ", num_sig_bonferroni, "\n")
cat("Holm: ", num_sig_holm, "\n")
cat("Hochberg: ", num_sig_hochberg, "\n")
cat("Benjamini-Hochberg (FDR): ", num_sig_bh, "\n")
cat("Adaptive FDR (q-value): ", num_sig_qvalue, "\n")
cat("Local FDR: ", num_sig_local_fdr, "\n")
# View the top results
head(results)
# Create a summary table for the number of significant genes
summary_table <- data.frame(
```



```
Method = c("Bonferroni", "Holm", "Hochberg", "Benjamini-Hochberg (FDR)"
           , "Adaptive FDR (q-value)", "Local FDR"),
Significant_Genes = c(num_sig_bonferroni, num_sig_holm, num_sig_hochberg,
                      num_sig_bh, num_sig_qvalue, num_sig_local_fdr)
)
print(summary_table)
# Define sets of significant genes for different methods
sig_genes_bonferroni <- rownames(results)[results$Bonferroni < 0.05]
sig_genes_holm <- rownames(results)[results$Holm < 0.05]
sig_genes_hochberg <- rownames(results)[results$Hochberg < 0.05]
sig_genes_bh <- rownames(results)[results$BH < 0.05]
sig_genes_qvalue <- rownames(results)[results$qvalue < 0.05]
sig_genes_local_fdr <- rownames(results)[results$local_fdr < 0.05]
# Create a bar plot for the number of significant genes detected by each method
sig_genes_count <- c(num_sig_bonferroni, num_sig_holm, num_sig_hochberg,
                    num_sig_bh, num_sig_qvalue, num_sig_local_fdr)
methods <- c("Bonferroni", "Holm", "Hochberg", "BH (FDR)", "q-value", "Local
FDR")
bar_data <- data.frame(Method = methods, Significant_Genes = sig_genes_count)
ggplot(bar_data, aes(x = Method, y = Significant_Genes)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  theme_minimal() +
  labs(title = "Number of Significant Genes Detected by Each Method",
       x = "Method", y = "Number of Significant Genes")
# Create a binary matrix for UpSetR
set_list <- list(
  Bonferroni = sig_genes_bonferroni,
  Holm = sig_genes_holm,
  Hochberg = sig_genes_hochberg,
  `BH (FDR)` = sig_genes_bh,
  `q-value` = sig_genes_qvalue,
  `Local FDR` = sig_genes_local_fdr
)
binary_matrix <- fromList(set_list)
```



```
# Create the UpSet plot
upset(binary_matrix, sets = c("Bonferroni", "Holm", "Hochberg", "BH (FDR)", "q-
value", "Local FDR"),
      main.bar.color = "blue", sets.bar.color = "purple", order.by = "freq")
# Step 1: Create a DE status for each method
results$DE_status_bonferroni <- ifelse(results$Bonferroni < 0.05, "DE Gene",
"DE Gene", "Not DE")
results$DE_status_holm <- ifelse(results$Holm < 0.05, "DE Gene", "Not DE")
results$DE_status_hochberg <- ifelse(results$Hochberg < 0.05, "DE Gene", "Not
DE")
results$DE_status_bh <- ifelse(results$BH < 0.05, "DE Gene", "Not DE")
results$DE_status_qvalue <- ifelse(results$qvalue < 0.05, "DE Gene", "Not DE")
results$DE_status_local_fdr <- ifelse(results$local_fdr < 0.05, "DE Gene", "Not
DE")
# Step 2: Run PCA on the expression data
pca <- prcomp(exprs_data_subset, scale = TRUE)
# Step 43: Create PCA plots with DE status for each method
# Bonferroni PCA plot
pca_df_bonferroni <- data.frame(PC1 = pca$x[, 1], PC2 = pca$x[, 2], DE_status =
results$DE_status_bonferroni)
ggplot(pca_df_bonferroni, aes(x = PC1, y = PC2, color = DE_status)) +
  geom_point() +
  labs(title = "PCA Plot - Bonferroni", x = "PC1", y = "PC2") +
  scale_color_manual(values = c("red", "gray")) +
  theme_minimal()
# Holm PCA plot
pca_df_holm <- data.frame(PC1 = pca$x[, 1], PC2 = pca$x[, 2], DE_status =
results$DE_status_holm)
ggplot(pca_df_holm, aes(x = PC1, y = PC2, color = DE_status)) +
  geom_point() +
  labs(title = "PCA Plot - Holm", x = "PC1", y = "PC2") +
  scale_color_manual(values = c("red", "gray")) +
  theme_minimal()
# Hochberg PCA plot
```



```
pca_df_hochberg <- data.frame(PC1 = pca$x[, 1], PC2 = pca$x[, 2], DE_status =
results$DE_status_hochberg)
ggplot(pca_df_hochberg, aes(x = PC1, y = PC2, color = DE_status)) +
  geom_point() +
  labs(title = "PCA Plot - Hochberg", x = "PC1", y = "PC2") +
  scale_color_manual(values = c("red", "gray")) +
  theme_minimal()
# BH (FDR) PCA plot
pca_df_bh <- data.frame(PC1 = pca$x[, 1], PC2 = pca$x[, 2], DE_status =
results$DE_status_bh)
ggplot(pca_df_bh, aes(x = PC1, y = PC2, color = DE_status)) +
  geom_point() +
  labs(title = "PCA Plot - BH (FDR)", x = "PC1", y = "PC2") +
  scale_color_manual(values = c("red", "gray")) +
  theme_minimal()
# q-value PCA plot
pca_df_qvalue <- data.frame(PC1 = pca$x[, 1], PC2 = pca$x[, 2], DE_status =
results$DE_status_qvalue)
ggplot(pca_df_qvalue, aes(x = PC1, y = PC2, color = DE_status)) +
  geom_point() +
  labs(title = "PCA Plot - q-value", x = "PC1", y = "PC2") +
  scale_color_manual(values = c("red", "gray")) +
  theme_minimal()
# Local FDR PCA plot
pca_df_local_fdr <- data.frame(PC1 = pca$x[, 1], PC2 = pca$x[, 2], DE_status =
results$DE_status_local_fdr)
ggplot(pca_df_local_fdr, aes(x = PC1, y = PC2, color = DE_status)) +
  geom_point() +
  labs(title = "PCA Plot - Local FDR", x = "PC1", y = "PC2") +
  scale_color_manual(values = c("red", "gray")) +
  theme_minimal()
```



### **Application 1 Part 1 Chapter 3**

```
# Parameters
set.seed(123) # Ensure reproducibility
n_genes <- 10000 # Total number of genes
n_samples <- 10 # Moderate sample size (5 per group)
prop_de <- 0.1 # Proportion of truly DE genes
effect_sizes <- c(0.5, 1.5, 3) # Small, moderate, and larger effect sizes
alpha <- 0.05 # Significance level for FDR
# Generate a consistent set of truly DE genes (1 for DE, 0 for non-DE)
is_de <- rbinom(n_genes, 1, prop_de)
simulate_data <- function(n_genes, n_samples, is_de, effect_size) {
  set.seed(123)
  variances <- 1 / rgamma(n_genes, shape = 3, scale = 1) # Gene-specific variances
  # Generate control data (mean = 0 for all genes)
  control_data <- matrix(rnorm(n_genes * (n_samples / 2), 0, sqrt(variances)), nrow
= n_genes)
  # Generate treatment data (mean shifted for DE genes)
  treatment_data <- matrix(rnorm(n_genes * (n_samples / 2), effect_size * is_de,
sqrt(variances)), nrow = n_genes)
  # Combine control and treatment data and return as a list
  data <- cbind(control_data, treatment_data)
  list(data = data, is_de = is_de, variances = variances)
}
apply_methods <- function(data, group, alpha) {
  # Limma Analysis
  design <- model.matrix(~group) # Design matrix for Limma
  fit <- lmFit(data, design) # Fit linear model for Limma
  fit <- eBayes(fit) # Apply empirical Bayes moderation
  limma_pvalues <- fit$p.value[, 2] # Extract p-values for the group effect
  limma_bh <- p.adjust(limma_pvalues, method = "BH") < alpha # Adjust using
BH
  # T-test Analysis
  t_test_pvalues <- apply(data, 1, function(row) t.test(row ~ group)$p.value)
```



```
t_bh <- p.adjust(t_test_pvalues, method = "BH") < alpha # Adjust using BH
# Return results as a list
list(
  limma_p = limma_pvalues,
  limma_bh = limma_bh,
  t_test_p = t_test_pvalues,
  t_bh = t_bh
)
}
calculate_metrics <- function(true_labels, predicted_labels) {
  # True Positives: correctly predicted DE genes
  tp <- sum(predicted_labels & true_labels)
  # False Positives: predicted DE genes that are not truly DE
  fp <- sum(predicted_labels & !true_labels)
  # False Negatives: truly DE genes that are not predicted
  fn <- sum(!predicted_labels & true_labels)
  # True Negatives: correctly predicted non-DE genes
  tn <- sum(!predicted_labels & !true_labels)
  # Sensitivity (Recall): True Positives / (True Positives + False Negatives)
  sensitivity <- ifelse((tp + fn) > 0, tp / (tp + fn), 0)
  # Specificity: True Negatives / (True Negatives + False Positives)
  specificity <- ifelse((tn + fp) > 0, tn / (tn + fp), 0)
  # False Discovery Rate (FDR): False Positives / (True Positives + False Positives)
  fdr <- ifelse((tp + fp) > 0, fp / (tp + fp), 0)
  # Return the metrics as a named vector
  c(Sensitivity = sensitivity, Specificity = specificity, FDR = fdr)
}

# Store results for plotting, Venn diagram, and table creation
results <- data.frame()
metrics_table <- data.frame()
venn_data <- list()
# Main loop for three effect sizes (small, moderate, large)
for (effect_size in effect_sizes) {
```



```
# Simulate data for each effect size
sim_data <- simulate_data(n_genes, n_samples, is_de, effect_size)
data <- sim_data$data
group <- factor(c(rep(0, n_samples / 2), rep(1, n_samples / 2)))
# Apply FDR adjustment methods
test_results <- apply_methods(data, group, alpha)
# Collect detected genes for Venn diagram
limma_detected <- which(test_results$limma_bh)
t_test_detected <- which(test_results$t_bh)
true_de_genes <- which(is_de == 1)
venn_data[[paste0("Effect_Size_", effect_size)]] <- list(
  True_DE = true_de_genes,
  Limma = limma_detected,
  T_test = t_test_detected
)
# Add counts to the summary table
results <- rbind(results, data.frame(
  Effect_Size = effect_size,
  Truly_DE = length(true_de_genes),
  Limma_Detected = length(limma_detected),
  T_test_Detected = length(t_test_detected)
))
# Calculate and store metrics for each method
metrics_limma <- calculate_metrics(is_de, test_results$limma_bh)
metrics_t_test <- calculate_metrics(is_de, test_results$t_bh)
metrics_table <- rbind(metrics_table,
  data.frame(Method = "limma", Effect_Size = effect_size,
    Sensitivity = metrics_limma["Sensitivity"],
    Specificity = metrics_limma["Specificity"],
    FDR = metrics_limma["FDR"]),
  data.frame(Method = "t-test", Effect_Size = effect_size,
    Sensitivity = metrics_t_test["Sensitivity"],
    Specificity = metrics_t_test["Specificity"],
    FDR = metrics_t_test["FDR"]))
```



```
}  
# Print summary table of detected DE counts  
print("Summary Table of DE Gene Counts:")  
print(results)  
# Print metrics table  
print("Metrics Table:")  
print(metrics_table)  
# Create Venn diagrams for each effect size and display them  
for (effect_size in effect_sizes) {  
  effect_size_label <- paste0("Effect_Size_", effect_size)  
  
  # Check if the data exists for the current effect size  
  if (!is.null(venn_data[[effect_size_label]])) {  
    venn.plot <- venn.diagram(  
      x = venn_data[[effect_size_label]],  
      category.names = c("Truly DE", "Limma", "T-test"),  
      filename = NULL,  
      output = TRUE,  
      main = paste("Venn Diagram for Effect Size =", effect_size),  
      fill = c("red", "green", "blue"),  
      alpha = 0.5,  
      cex = 1.5,  
      cat.cex = 1.2  
    )  
    # Display the Venn diagram  
    grid.newpage() # Ensure each plot is on a new page  
    grid.draw(venn.plot)  
  }  
}  
  
# Plot Sensitivity/Power vs. Effect Size  
ggplot(metrics_table, aes(x = Effect_Size, y = Sensitivity, color = Method)) +  
  geom_line() +  
  geom_point() +
```



```
labs(title = "Sensitivity (Power) vs. Effect Size", x = "Effect Size", y =
"Sensitivity / Power") +
  theme_minimal()
# Plot Specificity vs. Effect Size
ggplot(metrics_table, aes(x = Effect_Size, y = Specificity, color = Method)) +
  geom_line() +
  geom_point() +
  labs(title = "Specificity vs. Effect Size", x = "Effect Size", y = "Specificity") +
  theme_minimal()
# Plot FDR vs. Effect Size
ggplot(metrics_table, aes(x = Effect_Size, y = FDR, color = Method)) +
  geom_line() +
  geom_point() +
  labs(title = "False Discovery Rate vs. Effect Size", x = "Effect Size", y = "FDR")
+
  theme_minimal()
# Initialize a list for storing plots
limma_plots <- list()
limma_plot_data_list <- list() # Store data for each effect size
for (effect_size in effect_sizes) {
  # Simulate data for the current effect size
  sim_data <- simulate_data(n_genes = 10000, n_samples = 10, is_de = is_de,
effect_size = effect_size)
  data <- sim_data$data
  group <- factor(c(rep(0, n_samples / 2), rep(1, n_samples / 2)))

  # Apply Limma and T-Test methods
  test_results <- apply_methods(data, group, alpha)

  # Calculate mean differences and standard deviations
  mean_diff <- rowMeans(data[, 6:10]) - rowMeans(data[, 1:5]) # Treatment
Control
  sd_values <- apply(data, 1, sd) # Standard deviation of each gene
```



```
# Store data in a list so we don't overwrite results
limma_plot_data_list[[paste("Effect_Size", effect_size, sep = "_")] <-
data.frame(
  MeanDifference = mean_diff,
  StandardDeviation = sd_values,
  Is_DE = factor(is_de, levels = c(1, 0), labels = c("True DE", "Non-DE")),
  Significant = factor(test_results$limma_bh, levels = c(FALSE, TRUE), labels =
c("Not Significant", "Significant"))
)

# Create Limma Plot
limma_plot <- ggplot(limma_plot_data_list[[paste("Effect_Size", effect_size, sep
= "_")]],
  aes(x = MeanDifference, y = StandardDeviation, color = Is_DE,
shape = Significant)) +
  geom_point(alpha = 0.6) +
  scale_color_manual(values = c("red", "blue"), labels = c("True DE", "Non-
DE")) +
  scale_shape_manual(values = c(1, 16), labels = c("Not Significant",
"Significant")) + # Open circles = Not Significant
  labs(
    title = paste("Mean Difference vs Standard Deviation (Limma, Effect Size =",
effect_size, ")"),
    x = "Mean Difference",
    y = "Standard Deviation",
    color = "Gene Status",
    shape = "Significance"
  ) +
  theme_minimal()

# Store the plot in the list
limma_plots[[paste("Effect_Size", effect_size, sep = "_")] <- limma_plot
}

# Display all plots for Limma
```



```
for (plot_name in names(limma_plots)) {  
  print(limma_plots[[plot_name]])  
}  
### Section 2 Varying levels of Variance  
# Parameters  
set.seed(123) # Ensure reproducibility  
n_genes <- 10000      # Total number of genes  
n_samples <- 10       # Moderate sample size (5 per group)  
prop_de <- 0.1        # Proportion of truly DE genes  
effect_size <- 3      # Fixed effect size for this part  
alpha <- 0.05         # Significance level for FDR  
variance_levels <- list(  
  Low = list(shape = 8, scale = 0.5), Moderate = list(shape = 4, scale = 0.9),  
  High = list(shape = 2, scale = 3)  
)  
# Function to simulate data with varying levels of variance  
simulate_data_with_variance <- function(n_genes, n_samples, is_de, effect_size,  
shape, scale) {  
  set.seed(123) # Ensure reproducibility  
  variances <- 1 / rgamma(n_genes, shape = shape, scale = scale) # Gene-specific  
variances  
  # Generate control data (mean = 0 for all genes)  
  control_data <- matrix(rnorm(n_genes * (n_samples / 2), 0, sqrt(variances)), nrow  
= n_genes)  
  # Generate treatment data (mean shifted for DE genes)  
  treatment_data <- matrix(rnorm(n_genes * (n_samples / 2), effect_size * is_de,  
sqrt(variances)), nrow = n_genes)  
  # Combine control and treatment data and return as a list  
  data <- cbind(control_data, treatment_data)  
  list(data = data, is_de = is_de, variances = variances)  
}  
# Store results for plotting, Venn diagrams, and table creation  
results <- data.frame()  
metrics_table <- data.frame()
```



```
venn_data <- list()
# Main loop for varying levels of variance (Low, Moderate, High)
for (level in names(variance_levels)) {
  # Get the shape and scale for the current variance level
  shape <- variance_levels[[level]]$shape
  scale <- variance_levels[[level]]$scale
  # Simulate data for the current variance level
  sim_data <- simulate_data_with_variance(n_genes, n_samples, is_de, effect_size,
shape, scale)
  data <- sim_data$data
  group <- factor(c(rep(0, n_samples / 2), rep(1, n_samples / 2)))
  # Apply FDR adjustment methods
  test_results <- apply_methods(data, group, alpha)
  # Collect detected genes for Venn diagram
  limma_detected <- which(test_results$limma_bh)
  t_test_detected <- which(test_results$t_bh)
  true_de_genes <- which(is_de == 1)
  venn_data[[level]] <- list(
    True_DE = true_de_genes,
    Limma = limma_detected,
    T_test = t_test_detected
  )
  # Add counts to the summary table
  results <- rbind(results, data.frame(
    Variance_Level = level,
    Truly_DE = length(true_de_genes),
    Limma_Detected = length(limma_detected),
    T_test_Detected = length(t_test_detected)
  ))
  # Calculate and store metrics for each method
  metrics_limma <- calculate_metrics(is_de, test_results$limma_bh)
  metrics_t_test <- calculate_metrics(is_de, test_results$t_bh)
  metrics_table <- rbind(metrics_table,
    data.frame(Method = "limma", Variance_Level = level,
```



```
        Sensitivity = metrics_limma["Sensitivity"],
        Specificity = metrics_limma["Specificity"],
        FDR = metrics_limma["FDR"]),
    data.frame(Method = "t-test", Variance_Level = level,
               Sensitivity = metrics_t_test["Sensitivity"],
               Specificity = metrics_t_test["Specificity"],
               FDR = metrics_t_test["FDR"]))
}
# Print summary table of detected DE counts
print("Summary Table of DE Gene Counts:")
print(results)
# Print metrics table
print("Metrics Table:")
print(metrics_table)
# Create Venn diagrams for each variance level
for (level in names(venn_data)) {
  venn.plot <- venn.diagram(
    x = venn_data[[level]],
    category.names = c("Truly DE", "Limma", "T-test"),
    filename = NULL,
    output = TRUE,
    main = paste("Venn Diagram for Variance Level:", level),
    fill = c("red", "green", "blue"),
    alpha = 0.5,
    cex = 1.5,
    cat.cex = 1.2
  )
  # Display the Venn diagram
  grid.newpage()
  grid.draw(venn.plot)
}
# Ensure 'Method' is treated as a factor in the metrics_table
metrics_table$Method <- as.factor(metrics_table$Method)
metrics_table$Variance_Level <- as.factor(metrics_table$Variance_Level)
```



```
# Plot Sensitivity/Power vs. Variance Level
ggplot(metrics_table, aes(x = Variance_Level, y = Sensitivity, color = Method,
group = Method)) +
  geom_line(aes(group = Method)) +
  geom_point() +
  labs(title = "Sensitivity (Power) vs. Variance Level", x = "Variance Level", y =
"Sensitivity / Power") +
  theme_minimal()

# Plot Specificity vs. Variance Level
ggplot(metrics_table, aes(x = Variance_Level, y = Specificity, color = Method,
group = Method)) +
  geom_line(aes(group = Method)) +
  geom_point() +
  labs(title = "Specificity vs. Variance Level", x = "Variance Level", y =
"Specificity") +
  theme_minimal()

# Plot FDR vs. Variance Level
ggplot(metrics_table, aes(x = Variance_Level, y = FDR, color = Method, group =
Method)) +
  geom_line(aes(group = Method)) +
  geom_point() +
  labs(title = "False Discovery Rate vs. Variance Level", x = "Variance Level", y =
"FDR") +
  theme_minimal()

# Main loop for varying levels of variance (Low, Moderate, High)
# Initialize lists for storing plots and data
limma_plots <- list()
limma_plot_data_list <- list()

# Loop through variance levels (Low, Moderate, High)
for (level in names(variance_levels)) {
  shape <- variance_levels[[level]]$shape
  scale <- variance_levels[[level]]$scale
```



```
# Simulate data for current variance level
sim_data <- simulate_data_with_variance(n_genes, n_samples, is_de, effect_size,
shape, scale)
data <- sim_data$data
group <- factor(c(rep(0, n_samples / 2), rep(1, n_samples / 2)))
# Apply Limma method
test_results <- apply_methods(data, group, alpha)
# Calculate mean differences and standard deviations
mean_diff <- rowMeans(data[, (n_samples / 2 + 1):n_samples]) -
rowMeans(data[, 1:(n_samples / 2)])
sd_values <- apply(data, 1, sd)

# Store data in a list
limma_plot_data_list[[level]] <- data.frame(
  MeanDifference = mean_diff,
  StandardDeviation = sd_values,
  Is_DE = factor(is_de, levels = c(1, 0), labels = c("True DE", "Non-DE")),
  Significant = factor(test_results$limma_bh, levels = c(FALSE, TRUE), labels =
c("Not Significant", "Significant"))
)
# Create Limma plot
limma_plot <- ggplot(limma_plot_data_list[[level]],
  aes(x = MeanDifference, y = StandardDeviation, color = Is_DE,
shape = Significant)) +
  geom_point(alpha = 0.6) +
  scale_color_manual(values = c("red", "blue"), labels = c("True DE", "Non-
DE")) +
  scale_shape_manual(values = c(1, 16), labels = c("Not Significant",
"Significant")) +
  labs(
    title = paste("Mean Difference vs Standard Deviation (Limma, Variance Level
=", level, ")"),
    x = "Mean Difference",
    y = "Standard Deviation",
```



```
    color = "Gene Status",
    shape = "Significance"
  ) +
  theme_minimal()

limma_plots[[level]] <- limma_plot
}
# Display all Limma plots
for (plot_name in names(limma_plots)) {
  print(limma_plots[[plot_name]])
}
### Section 3 Varying sample sizes
# Parameters
set.seed(123)
n_genes <- 10000      # Total number of genes
prop_de <- 0.1        # Proportion of truly DE genes
effect_size <- 3      # Fixed effect size for this part
alpha <- 0.05         # Significance level for FDR
sample_sizes <- c(5, 10, 20) # Small, moderate, and large sample sizes
shape <- 3            # Moderate variance level
scale <- 1            # Moderate variance level
# Function to simulate data with varying sample sizes
simulate_data_with_sample_size <- function(n_genes, n_samples, is_de,
effect_size, shape, scale) {
  set.seed(123)
  variances <- 1 / rgamma(n_genes, shape = shape, scale = scale) # Gene-specific
variances
  # Generate control data (mean = 0 for all genes)
  control_data <- matrix(rnorm(n_genes * (n_samples / 2), 0, sqrt(variances)), nrow
= n_genes)
  # Generate treatment data (mean shifted for DE genes)
  treatment_data <- matrix(rnorm(n_genes * (n_samples / 2), effect_size * is_de,
sqrt(variances)), nrow = n_genes)
  # Combine control and treatment data and return as a list
```



```
data <- cbind(control_data, treatment_data)
list(data = data, is_de = is_de, variances = variances)
}
# Store results for plotting, Venn diagrams, and table creation
results <- data.frame()
metrics_table <- data.frame()
venn_data <- list()
# Main loop for varying sample sizes (Small, Moderate, Large)
for (n_samples in sample_sizes) {
  # Simulate data for the current sample size
  sim_data <- simulate_data_with_sample_size(n_genes, n_samples * 2, is_de,
effect_size, shape, scale)
  data <- sim_data$data
  group <- factor(c(rep(0, n_samples), rep(1, n_samples)))
  # Apply FDR adjustment methods
  test_results <- apply_methods(data, group, alpha)
  # Collect detected genes for Venn diagram
  limma_detected <- which(test_results$limma_bh)
  t_test_detected <- which(test_results$t_bh)
  true_de_genes <- which(is_de == 1)
  venn_data[[paste0("Sample_Size_", n_samples)]] <- list(
    True_DE = true_de_genes,
    Limma = limma_detected,
    T_test = t_test_detected
  )
  # Add counts to the summary table
  results <- rbind(results, data.frame(
    Sample_Size = n_samples,
    Truly_DE = length(true_de_genes),
    Limma_Detected = length(limma_detected),
    T_test_Detected = length(t_test_detected)
  ))
  # Calculate and store metrics for each method
  metrics_limma <- calculate_metrics(is_de, test_results$limma_bh)
```



```
metrics_t_test <- calculate_metrics(is_de, test_results$t_bh)
metrics_table <- rbind(metrics_table,
  data.frame(Method = "limma", Sample_Size = n_samples,
    Sensitivity = metrics_limma["Sensitivity"],
    Specificity = metrics_limma["Specificity"],
    FDR = metrics_limma["FDR"]),
  data.frame(Method = "t-test", Sample_Size = n_samples,
    Sensitivity = metrics_t_test["Sensitivity"],
    Specificity = metrics_t_test["Specificity"],
    FDR = metrics_t_test["FDR"]))
}
# Print summary table of detected DE counts
print("Summary Table of DE Gene Counts:")
print(results)
# Print metrics table
print("Metrics Table:")
print(metrics_table)
# Create Venn diagrams for each sample size
for (n_samples in sample_sizes) {
  sample_size_label <- paste0("Sample_Size_", n_samples)
  venn.plot <- venn.diagram(
    x = venn_data[[sample_size_label]],
    category.names = c("Truly DE", "Limma", "T-test"),
    filename = NULL,
    output = TRUE,
    main = paste("Venn Diagram for Sample Size:", n_samples),
    fill = c("red", "green", "blue"),
    alpha = 0.5,
    cex = 1.5,
    cat.cex = 1.2
  )
# Display the Venn diagram
grid.newpage()
grid.draw(venn.plot)
```



```
}  
# Plot Sensitivity/Power vs. Sample Size  
ggplot(metrics_table, aes(x = as.factor(Sample_Size), y = Sensitivity, color =  
Method, group = Method)) +  
  geom_line() +  
  geom_point() +  
  labs(title = "Sensitivity (Power) vs. Sample Size", x = "Sample Size", y =  
"Sensitivity / Power") +  
  theme_minimal()  
# Plot Specificity vs. Sample Size  
ggplot(metrics_table, aes(x = as.factor(Sample_Size), y = Specificity, color =  
Method, group = Method)) +  
  geom_line() +  
  geom_point() +  
  labs(title = "Specificity vs. Sample Size", x = "Sample Size", y = "Specificity") +  
  theme_minimal()  
# Plot FDR vs. Sample Size  
ggplot(metrics_table, aes(x = as.factor(Sample_Size), y = FDR, color = Method,  
group = Method)) +  
  geom_line() +  
  geom_point() +  
  labs(title = "False Discovery Rate vs. Sample Size", x = "Sample Size", y =  
"FDR") +  
  theme_minimal()  
# Initialize lists for storing plots  
limma_plots <- list()  
limma_plot_data_list <- list()  
# Loop through sample sizes (5, 10, 20)  
for (n_samples in sample_sizes) {  
  # Simulate data for the current sample size  
  sim_data <- simulate_data_with_sample_size(n_genes, n_samples * 2, is_de,  
effect_size, shape, scale)  
  data <- sim_data$data  
  group <- factor(c(rep(0, n_samples), rep(1, n_samples)))
```



```
# Apply Limma and T-Test methods
test_results <- apply_methods(data, group, alpha)
# Calculate mean differences and standard deviations
mean_diff <- rowMeans(data[, (n_samples/2 + 1):n_samples]) - rowMeans(data[,
1:(n_samples/2)])
sd_values <- apply(data, 1, sd)
# Store data in a list so we don't overwrite results
limma_plot_data_list[[paste0("Sample_Size_", n_samples)]] <- data.frame(
  MeanDifference = mean_diff,
  StandardDeviation = sd_values,
  Is_DE = factor(is_de, levels = c(1, 0), labels = c("True DE", "Non-DE")),
  Significant = factor(test_results$limma_bh, levels = c(FALSE, TRUE), labels =
c("Not Significant", "Significant"))
)
# Create Limma plot
limma_plot <- ggplot(limma_plot_data_list[[paste0("Sample_Size_",
n_samples)]]),
  aes(x = MeanDifference, y = StandardDeviation, color = Is_DE,
shape = Significant)) +
  geom_point(alpha = 0.6) +
  scale_color_manual(values = c("red", "blue"), labels = c("True DE", "Non-
DE")) +
  scale_shape_manual(values = c(1, 16), labels = c("Not Significant",
"Significant")) +
  labs(
    title = paste("Mean Difference vs Standard Deviation (Limma, Sample Size =",
n_samples, ")"),
    x = "Mean Difference",
    y = "Standard Deviation",
    color = "Gene Status",
    shape = "Significance"
  ) +
  theme_minimal()
```



```
limma_plots[[paste0("Sample_Size_", n_samples)]] <- limma_plot
}
# Display all Limma plots
for (plot_name in names(limma_plots)) {
  print(limma_plots[[plot_name]])
}
## Section 4 varying levels of alpha
# Parameters
set.seed(123)
n_genes <- 10000      # Total number of genes
n_samples <- 10       # 10 samples per group
prop_de <- 0.1        # Proportion of truly DE genes
effect_size <- 3      # Fixed effect size
alpha_levels <- c(0.01, 0.05, 0.1) # Varying alpha levels
shape <- 4            # Moderate variance level
scale <- 0.9         # Moderate variance level
# Function to simulate data with fixed sample size and variance level
simulate_data_fixed <- function(n_genes, n_samples, is_de, effect_size, shape,
scale) {
  set.seed(123)
  variances <- 1 / rgamma(n_genes, shape = shape, scale = scale) # Gene-specific
variances
  # Generate control data (mean = 0 for all genes)
  control_data <- matrix(rnorm(n_genes * (n_samples / 2), 0, sqrt(variances)), nrow
= n_genes)
  # Generate treatment data (mean shifted for DE genes)
  treatment_data <- matrix(rnorm(n_genes * (n_samples / 2), effect_size * is_de,
sqrt(variances)), nrow = n_genes)
  # Combine control and treatment data and return as a list
  data <- cbind(control_data, treatment_data)
  list(data = data, is_de = is_de, variances = variances)
}
# Simulate data
```



```
sim_data <- simulate_data_fixed(n_genes, n_samples * 2, is_de, effect_size, shape,
scale)
data <- sim_data$data
group <- factor(c(rep(0, n_samples), rep(1, n_samples)))
# Store results for plotting, Venn diagrams, and table creation
results <- data.frame()
metrics_table <- data.frame()
venn_data <- list()
# Main loop for varying alpha levels (0.01, 0.05, 0.1)
for (alpha in alpha_levels) {
  # Apply FDR adjustment methods
  test_results <- apply_methods(data, group, alpha)
  # Collect detected genes for Venn diagram
  limma_detected <- which(test_results$limma_bh)
  t_test_detected <- which(test_results$t_bh)
  true_de_genes <- which(is_de == 1)
  venn_data[[paste0("Alpha_", alpha)]] <- list(
    True_DE = true_de_genes,
    Limma = limma_detected,
    T_test = t_test_detected
  )
  # Add counts to the summary table
  results <- rbind(results, data.frame(
    Alpha = alpha,
    Truly_DE = length(true_de_genes),
    Limma_Detected = length(limma_detected),
    T_test_Detected = length(t_test_detected)
  ))
  # Calculate and store metrics for each method
  metrics_limma <- calculate_metrics(is_de, test_results$limma_bh)
  metrics_t_test <- calculate_metrics(is_de, test_results$t_bh)
  metrics_table <- rbind(metrics_table,
    data.frame(Method = "limma", Alpha = alpha,
      Sensitivity = metrics_limma["Sensitivity"]),
```



```
        Specificity = metrics_limma["Specificity"],
        FDR = metrics_limma["FDR"]),
data.frame(Method = "t-test", Alpha = alpha,
        Sensitivity = metrics_t_test["Sensitivity"],
        Specificity = metrics_t_test["Specificity"],
        FDR = metrics_t_test["FDR"]))
}
# Print summary table of detected DE counts
print("Summary Table of DE Gene Counts:")
print(results)
# Print metrics table
print("Metrics Table:")
print(metrics_table)
# Create Venn diagrams for each alpha level
for (alpha in alpha_levels) {
  alpha_label <- paste0("Alpha_", alpha)
  venn.plot <- venn.diagram(
    x = venn_data[[alpha_label]],
    category.names = c("Truly DE", "Limma", "T-test"),
    filename = NULL,
    output = TRUE,
    main = paste("Venn Diagram for Alpha Level:", alpha),
    fill = c("red", "green", "blue"),
    alpha = 0.5,
    cex = 1.5,
    cat.cex = 1.2
  )
  # Display the Venn diagram
  grid.newpage()
  grid.draw(venn.plot)
}
# Plot Sensitivity/Power vs. Alpha Level
ggplot(metrics_table, aes(x = as.factor(Alpha), y = Sensitivity, color = Method,
group = Method)) +
```



```
geom_line() +  
geom_point() +  
labs(title = "Sensitivity (Power) vs. Alpha Level", x = "Alpha Level", y =  
"Sensitivity / Power") +  
theme_minimal()  
# Plot Specificity vs. Alpha Level  
ggplot(metrics_table, aes(x = as.factor(Alpha), y = Specificity, color = Method,  
group = Method)) +  
geom_line() +  
geom_point() +  
labs(title = "Specificity vs. Alpha Level", x = "Alpha Level", y = "Specificity") +  
theme_minimal()  
# Plot FDR vs. Alpha Level  
ggplot(metrics_table, aes(x = as.factor(Alpha), y = FDR, color = Method, group =  
Method)) +  
geom_line() +  
geom_point() +  
labs(title = "False Discovery Rate vs. Alpha Level", x = "Alpha Level", y =  
"FDR") +  
theme_minimal()
```

### **Application 1 Part 2 Chapter 3**

```
# Parameters for simulation  
set.seed(123)  
n_genes <- 10000      # Total number of genes  
n_samples_per_group <- 4 # Small sample size per group  
prop_de <- 0.1       # Proportion of truly DE genes  
effect_size_a <- 1   # Mean shift for Treatment A  
effect_size_b <- 1.4 # Mean shift for Treatment B  
alpha <- 0.01       # Significance level  
shape <- 2          # Higher variance level  
scale <- 3          # Higher variance level  
# Generate a consistent set of truly DE genes  
set.seed(123) # Ensure reproducibility
```



```
is_de <- rbinom(n_genes, 1, prop_de)
# Function to simulate data for three groups
simulate_three_group_data <- function(n_genes, n_samples_per_group, is_de,
effect_size_a, effect_size_b, shape, scale) {
  set.seed(123)
  variances <- 1 / rgamma(n_genes, shape = shape, scale = scale) # Gene-specific
variances
  # Generate control data (mean = 0 for all genes)
  control_data <- matrix(rnorm(n_genes * n_samples_per_group, 0,
sqrt(variances))), nrow = n_genes)
  # Generate treatment data with mean shifts for DE genes
  treatment_a_data <- matrix(rnorm(n_genes * n_samples_per_group,
effect_size_a * is_de, sqrt(variances))), nrow = n_genes)
  treatment_b_data <- matrix(rnorm(n_genes * n_samples_per_group,
effect_size_b * is_de, sqrt(variances))), nrow = n_genes)
  # Combine data for all three groups
  data <- cbind(control_data, treatment_a_data, treatment_b_data)
  list(data = data, is_de = is_de, variances = variances)
}
# Simulate data
sim_data <- simulate_three_group_data(n_genes, n_samples_per_group, is_de,
effect_size_a, effect_size_b, shape, scale)
data <- sim_data$data
group <- factor(rep(1:3, each = n_samples_per_group))
# Apply limma for global testing across all three groups
design <- model.matrix(~ 0 + group) # Design matrix without intercept for limma
colnames(design) <- c("Control", "TreatmentA", "TreatmentB")
fit <- lmFit(data, design)
fit <- eBayes(fit) # Apply eBayes without contrasts for a global F-test
limma_pvalues <- fit$F.p.value # Global test p-values from limma
limma_significant <- p.adjust(limma_pvalues, method = "fdr") < alpha
# Apply ANOVA F-test for global testing
anova_pvalues <- apply(data, 1, function(x) {
  anova_result <- aov(x ~ group)
```



```
summary(anova_result)[[1]][["Pr(>F)"]][1] # Extract p-value
})
anova_significant <- p.adjust(anova_pvalues, method = "fdr") < alpha
# Collect true DE genes and results for Venn diagrams
true_de_genes <- which(is_de == 1)
limma_detected <- which(limma_significant)
anova_detected <- which(anova_significant)
venn_data <- list(
  True_DE = true_de_genes,
  Limma = limma_detected,
  ANOVA_F = anova_detected
)
# Print results
results <- data.frame(
  Method = c("Truly_DE", "Limma", "ANOVA_F"),
  Count = c(length(true_de_genes), length(limma_detected),
length(anova_detected))
)
print("Summary Table of DE Gene Counts:")
print(results)
# Metrics calculation function for sensitivity, specificity, and FDR
calculate_metrics <- function(true_labels, predictions) {
  tp <- sum(predictions & true_labels) # True positives
  fp <- sum(predictions & !true_labels) # False positives
  fn <- sum(!predictions & true_labels) # False negatives
  tn <- sum(!predictions & !true_labels) # True negatives

  sensitivity <- ifelse((tp + fn) > 0, tp / (tp + fn), 0) # Sensitivity
  specificity <- ifelse((tn + fp) > 0, tn / (tn + fp), 0) # Specificity
  fdr <- ifelse((tp + fp) > 0, fp / (tp + fp), 0) # False Discovery Rate

  c(Sensitivity = sensitivity, Specificity = specificity, FDR = fdr)
}
# Calculate metrics for each method
```



```
metrics_table <- data.frame(  
  Method = c("limma", "ANOVA_F"),  
  Sensitivity = c(  
    calculate_metrics(is_de, limma_significant)["Sensitivity"],  
    calculate_metrics(is_de, anova_significant)["Sensitivity"]  
  ),  
  Specificity = c(  
    calculate_metrics(is_de, limma_significant)["Specificity"],  
    calculate_metrics(is_de, anova_significant)["Specificity"]  
  ),  
  FDR = c(  
    calculate_metrics(is_de, limma_significant)["FDR"],  
    calculate_metrics(is_de, anova_significant)["FDR"]  
  )  
)  
  
# Print metrics table  
print("Metrics Table:")  
print(metrics_table)  
  
# Create a Venn diagram for the comparison  
venn.plot <- venn.diagram(  
  x = venn_data,  
  category.names = c("Truly DE", "Limma", "ANOVA F-test"),  
  filename = NULL,  
  output = TRUE,  
  main = "Venn Diagram for Global Test of Three Groups",  
  fill = c("red", "green", "blue"),  
  alpha = 0.5,  
  cex = 1.5,  
  cat.cex = 1.2  
)  
  
# Display the Venn diagram  
grid.newpage()  
grid.draw(venn.plot)  
  
# Scores for ROC curve
```



```
limma_scores <- -log10(limma_pvalues) # Use -log10(p-values) as scores
anova_scores <- -log10(anova_pvalues) # Similarly for ANOVA F-test
labels <- is_de
library(pROC)
# ROC curve for Limma
roc_limma <- roc(labels, limma_scores)
auc_limma <- auc(roc_limma)
# Calculate ROC and AUC for ANOVA F-test
roc_anova <- roc(labels, anova_scores)
auc_anova <- auc(roc_anova)
# Plot ROC curve for Limma
plot(roc_limma, col = "blue", main = "ROC Curve for Limma and ANOVA F-test")
text(0.6, 0.3, paste("Limma AUC =", round(auc_limma, 3)), col = "blue", cex =
1.2, font = 2) # Add AUC to plot
# Add ROC curve for ANOVA F-test
plot(roc_anova, col = "red", add = TRUE)
text(0.6, 0.2, paste("ANOVA F-test AUC =", round(auc_anova, 3)), col = "red",
cex = 1.2, font = 2) # Add AUC to plot
# Add legend
legend("bottomright", legend = c("Limma", "ANOVA F-test"), col = c("blue",
"red"), lty = 1, cex = 1.2)
# Store pairwise plots
limma_pairwise_plots <- list()
# Define pairwise group comparisons
pairwise_comparisons <- list(
  "Control_vs_TreatmentA" = c("Control", "TreatmentA"),
  "Control_vs_TreatmentB" = c("Control", "TreatmentB"),
  "TreatmentA_vs_TreatmentB" = c("TreatmentA", "TreatmentB")
)
# Loop through comparisons
for (comparison in names(pairwise_comparisons)) {
  groups <- pairwise_comparisons[[comparison]]
  # Get column indices for each group
  group1_index <- which(colnames(design) == groups[1])
```



```
group2_index <- which(colnames(design) == groups[2])
# Get sample indices for each group
group1_cols <- ((group1_index - 1) * n_samples_per_group + 1):(group1_index
* n_samples_per_group)
group2_cols <- ((group2_index - 1) * n_samples_per_group + 1):(group2_index
* n_samples_per_group)
# Compute Mean Difference and Standard Deviation
mean_diff <- rowMeans(data[, group2_cols]) - rowMeans(data[, group1_cols])
sd_values <- apply(data[, c(group1_cols, group2_cols)], 1, sd)
# Use global F-test significance results
significant_genes <- factor(limma_significant, levels = c(TRUE, FALSE), labels
= c("Significant", "Not Significant"))
# Create dataframe for plotting
plot_data <- data.frame(
  MeanDifference = mean_diff,
  StandardDeviation = sd_values,
  Is_DE = factor(is_de, levels = c(1, 0), labels = c("True DE", "Non-DE")),
  Significant = significant_genes
)
# Generate plot
limma_plot <- ggplot(plot_data, aes(x = MeanDifference, y = StandardDeviation,
color = Is_DE, shape = Significant)) +
  geom_point(alpha = 0.6) +
  scale_color_manual(values = c("red", "blue"), labels = c("True DE", "Non-
DE")) +
  scale_shape_manual(values = c(16, 1), labels = c("Significant", "Not
Significant")) +
  labs(
    title = paste("Mean Difference vs Standard Deviation (Limma:", comparison,
    ")"),
    x = "Mean Difference",
    y = "Standard Deviation",
    color = "Gene Status",
    shape = "Significance"
```



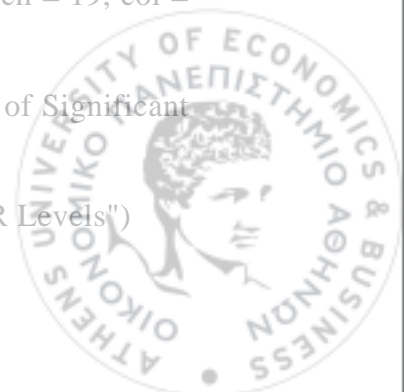
```
) +  
  theme_minimal()  
  
# Store plot  
limma_pairwise_plots[[comparison]] <- limma_plot  
}  
# Display all pairwise comparison plots  
for (plot_name in names(limma_pairwise_plots)) {  
  print(limma_pairwise_plots[[plot_name]])  
}
```

### **Application 2 Chapter 3**

```
library("leukemiasEset")  
library(limma)  
library(ggplot2)  
library(VennDiagram)  
# Global F-test with Limma  
# Step 1: Prepare the data  
expression_data <- exprs(leukemiasEset)  
phenotype_data <- pData(leukemiasEset)  
expression_data <- as.matrix(expression_data) # Expression matrix  
group <- phenotype_data$LeukemiaType      # Group information  
# Subset data for the selected groups  
groups <- c("ALL", "AML", "CLL", "CML", "NoL")  
subset_idx <- group %in% groups  
expression_data_subset <- expression_data[, subset_idx]  
subtype_subset <- factor(group[subset_idx], levels = groups)  
# Step 2: Design matrix for limma  
design <- model.matrix(~ subtype_subset) # Includes intercept  
colnames(design) <- levels(subtype_subset)  
# Step 3: Fit the linear model  
fit <- lmFit(expression_data_subset, design)  
# Step 4: Perform the global F-test  
# Test for any difference across the groups  
fit_f_test <- eBayes(fit)
```



```
f_test_results <- topTable(fit_f_test, coef = 2:ncol(design), number = Inf, sort.by =
"F", adjust.method = "fdr")
# Step 5: Filter significant genes
f_test_significant <- subset(f_test_results, adj.P.Val < 0.05)
# Print the number of significant genes
cat("Number of significant genes (Global F-test):", nrow(f_test_significant), "\n")
# Step 6: Visualization
# Volcano Plot for Global F-Test
f_test_results$Significant <- ifelse(
  f_test_results$adj.P.Val < 0.05, "Significant", "Not Significant"
)
ggplot(f_test_results, aes(x = AveExpr, y = -log10(adj.P.Val), color = Significant))
+
  geom_point(alpha = 0.8, size = 1.5) +
  scale_color_manual(values = c("red", "grey")) +
  theme_minimal() +
  labs(
    title = "Volcano Plot: Global F-Test",
    x = "Average Expression",
    y = "-Log10 Adjusted P-Value"
  ) +
  geom_hline(yintercept = -log10(0.05), linetype = "dashed", color = "blue")
# MA Plot for F-Test
plotMA(fit_f_test, main = "MA Plot: Global F-Test Across Groups")
## Proportion of Significant Genes Plot
alpha_levels <- seq(0.01, 0.1, by = 0.01)
prop_significant <- sapply(alpha_levels, function(alpha) {
  mean(f_test_results$adj.P.Val < alpha)
})
proportion_plot <- plot(alpha_levels, prop_significant, type = "b", pch = 19, col =
"blue",
  xlab = "FDR Threshold (alpha)", ylab = "Proportion of Significant
Genes",
  main = "Proportion of Significant Genes Across FDR Levels")
```



```
# Display proportions at each alpha level
cat("Proportion of Significant Genes at Different Alpha Levels:\n")
print(data.frame(FDR_Threshold = alpha_levels, Proportion_Significant =
prop_significant))
#Pairwise comparisons
# Step 1: Prepare the data
expression_data <- exprs(leukemiasEset) # Expression matrix
phenotype_data <- pData(leukemiasEset) # Phenotype data
group <- phenotype_data$LeukemiaType # Group information
# Subset data for NoL vs. each leukemia subtype
groups <- c("ALL", "AML", "CLL", "CML", "NoL")
subset_idx <- group %in% groups
expression_data_subset <- expression_data[, subset_idx]
subtype_subset <- factor(group[subset_idx], levels = groups)
# Step 2: Design matrix for limma
design <- model.matrix(~ 0 + subtype_subset) # No intercept to avoid collinearity
colnames(design) <- levels(subtype_subset)
# Step 3: Fit the linear model
fit <- lmFit(expression_data_subset, design)
# Step 4: Define contrasts for NoL vs. each subtype
contrast_matrix <- makeContrasts(
  ALL_vs_NoL = ALL - NoL,
  AML_vs_NoL = AML - NoL,
  CLL_vs_NoL = CLL - NoL,
  CML_vs_NoL = CML - NoL,
  levels = design
)
fit2 <- contrasts.fit(fit, contrast_matrix)
fit2 <- eBayes(fit2)
# Step 5: Extract results for each comparison
nol_results <- list()
for (contrast in colnames(contrast_matrix)) {
  results <- topTable(fit2, coef = contrast, number = Inf, adjust.method = "fdr")
  nol_results[[contrast]] <- results
}
```



```
    cat("Number of significant genes for", contrast, ":", sum(results$adj.P.Val <
0.05), "\n")
  }
# Step 6: Volcano Plots for Each Comparison
for (contrast in names(nol_results)) {
  results <- nol_results[[contrast]]
  # Add significance status for volcano plot
  results$Significant <- ifelse(
    results$adj.P.Val < 0.05 & abs(results$logFC) > 1, "Significant", "Not
Significant"
  )
  # Volcano plot
  p <- ggplot(results, aes(x = logFC, y = -log10(adj.P.Val), color = Significant)) +
    geom_point(alpha = 0.8, size = 1.5) +
    scale_color_manual(values = c("red", "grey")) +
    labs(
      title = paste("Volcano Plot:", contrast),
      x = "Log Fold Change",
      y = "-Log10 Adjusted P-Value",
      color = "Gene Status"
    ) +
    geom_hline(yintercept = -log10(0.05), linetype = "dashed", color = "blue") +
    geom_vline(xintercept = c(-1, 1), linetype = "dashed", color = "blue") +
    theme_minimal()

  print(p)
}
# Step 7
# Mean vs. Variance (or Standard Deviation) Plot for Pairwise Comparisons
# Initialize an empty list to store plots
mean_diff_variance_plots <- list()
# Loop through each pairwise comparison
for (contrast_name in names(nol_results)) {
  # Extract results for the current comparison
```



```
results <- nol_results[[contrast_name]]
# Calculate mean difference and standard deviation for each gene
mean_diff <- rowMeans(expression_data_subset[, group == sub("_(.*)", "",
contrast_name)]) -
  rowMeans(expression_data_subset[, group == "NoL"])
std_dev <- apply(expression_data_subset, 1, sd)
# Add these to the results table
results$Mean_Difference <- mean_diff[rownames(results)]
results$Standard_Deviation <- std_dev[rownames(results)]
# Add differential expression status and point size (-log10(p-value))
results$DE_Status <- ifelse(results$adj.P.Val < 0.05 & abs(results$logFC) > 1,
"DE", "Non-DE")
results$Point_Size <- -log10(results$adj.P.Val) # Point size based on -log10(p-
value)
# Create the plot for the current comparison
plot <- ggplot(results, aes(
  x = Mean_Difference,
  y = Standard_Deviation,
  color = DE_Status,
  size = Point_Size
)) +
  geom_point(alpha = 0.6) +
  scale_color_manual(values = c("Non-DE" = "blue", "DE" = "red")) +
  scale_size_continuous(range = c(1, 6), guide = "none") + # Scale point size,
hide size guide
  theme_minimal() +
  labs(
    title = paste("Mean Difference vs. Standard Deviation:", contrast_name),
    x = "Mean Difference",
    y = "Standard Deviation",
    color = "Gene Status"
  )
# Save the plot to the list
mean_diff_variance_plots[[contrast_name]] <- plot
```



```
}  
# Display all plots  
for (contrast_name in names(mean_diff_variance_plots)) {  
  print(mean_diff_variance_plots[[contrast_name]])  
}  
# Extract Significant Genes from Pairwise Comparisons  
significant_genes <- list()  
for (contrast_name in names(nol_results)) {  
  significant_genes[[contrast_name]] <-  
rownames(subset(nol_results[[contrast_name]], adj.P.Val < 0.05))  
}  
# Venn Diagram for Overlaps  
# Generate a Venn diagram  
venn.plot <- venn.diagram(  
  x = significant_genes,  
  category.names = names(significant_genes),  
  filename = NULL, # To display on screen instead of saving to file  
  output = TRUE,  
  fill = c("red", "blue", "green", "purple"),  
  alpha = 0.5,  
  cex = 1.5,  
  cat.cex = 1.2,  
  main = "Overlap of DE Genes Across Comparisons"  
)  
# Display the Venn diagram  
grid.newpage()  
grid.draw(venn.plot)  
# Identify Shared and Unique Genes  
# Shared genes across all comparisons  
shared_genes <- Reduce(intersect, significant_genes)  
cat("Number of shared genes across all comparisons:", length(shared_genes), "\n")  
# Unique genes for each comparison  
unique_genes <- lapply(significant_genes, function(genes) {
```



```
    setdiff(genes,          unlist(significant_genes[names(significant_genes)
names(genes)]))
  })
# Print counts of unique genes per comparison
unique_gene_counts <- sapply(unique_genes, length)
cat("Number of unique genes for each comparison:\n")
print(unique_gene_counts)
```

#### **Application 1 Chapter 4**

```
# Load required libraries
library(GEOquery)
library(dplyr)
library(limma)
library(tidyverse)
library(Biobase) # For handling ExpressionSet
library(puma)
library(mclust)
library(pgmm)
library(fabMix)
library(flexclust)
library(cluster)
library(clusterCrit)
library(ggplot2)
library(plotly)
library(RColorBrewer)
# Step 1: Load the Dataset
# Download the GDS912 dataset
gds912 <- getGEO("GDS912")
# Convert GDS to ExpressionSet
eset <- GDS2eSet(gds912)
# Check the structure of the ExpressionSet
print(eset)
# Step 1: Extract Expression Data
expression_matrix <- exprs(eset) # Extract the expression matrix
```



```
cat("Dimensions of the expression matrix: ", dim(expression_matrix), "\n")
# Extract descriptions and filter synchronized samples
# Extract the descriptions
descriptions <- pData(eset)$description
# Correct regex to handle "day", "week", "month", and "year"
time_points <- gsub(".*Postnatal (day [0-9]+|[0-9]+-[a-z]+),.*", "\\1",
descriptions)
# Check the unique time points to verify correct extraction
unique_time_points <- unique(time_points)
cat("Extracted time points:\n")
print(unique_time_points)
# Define synchronized ages
synchronized_ages <- c("day 1", "day 6", "day 14", "day 17", "day 23")
# Subset the dataset for samples matching synchronized ages
synchronous_samples <- which(time_points %in% synchronized_ages)
# Subset the ExpressionSet
eset_sync <- eset[, synchronous_samples]
# Verify the dimensions of the subset
cat("Dimensions of the dataset after extracting synchronized ages:",
dim(exprs(eset_sync)), "\n")
# Extract expression data and synchronized time points
expression_data <- exprs(eset_sync)
time_points_sync <- time_points[synchronous_samples]
# Create a design matrix for the synchronized time points
design <- model.matrix(~ 0 + factor(time_points_sync))
colnames(design) <- unique(time_points_sync)
# Normalize the expression data (z-score normalization)
normalized_matrix <- t(apply(expression_data, 1, function(x) {
(x - mean(x)) / sd(x)
}))
set.seed(123)
# Run the F-test on the normalized data
fit <- lmFit(normalized_matrix, design)
fit <- eBayes(fit)
```



```
# Identify significant genes
f_test_results <- topTable(fit, coef = NULL, number = Inf, sort.by = "F")
significant_genes <- f_test_results[f_test_results$adj.P.Val < 0.05, ]
# Select top 500 genes
top_genes <- rownames(head(significant_genes, 1100))
filtered_matrix <- normalized_matrix[top_genes, ]
# Create a new ExpressionSet
filtered_eset_sync <- ExpressionSet(
  assayData = filtered_matrix,
  phenoData = phenoData(eset_sync),
  featureData = featureData(eset_sync)[top_genes, ],
  experimentData = experimentData(eset_sync),
  annotation = annotation(eset_sync)
)
# Verify the dimensions
cat("Dimensions of the filtered dataset (normalized and filtered):",
dim(exprs(filtered_eset_sync)), "\n")
# Extract expression data from the filtered and normalized ExpressionSet
expression_data <- exprs(filtered_eset_sync)
# Set measurement error
measurement_error <- NULL
# Define the range of clusters to test
cluster_range_puma <- 2:35
# Initialize storage for BIC values
bic_values_puma <- numeric(length(cluster_range_puma))
set.seed(123)
# Loop over cluster numbers and run PUMA-CLUST
for (k in cluster_range_puma) {
  cat("Running PUMA-CLUST with", k, "clusters...\n")
  # Run pumaClust
  puma_result <- pumaClust(
    e = expression_data,
    se = measurement_error,
    clusters = k
```



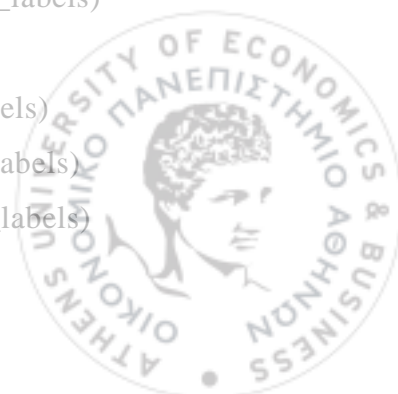
```
)  
# Store the BIC value  
bic_values_puma[k - 1] <- puma_result$bic  
}  
# Identify the optimal number of clusters  
optimal_clusters_puma <- cluster_range_puma[which.min(bic_values_puma)]  
cat("Optimal number of clusters:", optimal_clusters_puma, "\n")  
# Plot the BIC values for visualization  
plot(cluster_range_puma, bic_values_puma, type = "b", col = "blue", pch = 19,  
      xlab = "Number of Clusters", ylab = "BIC", main = "BIC vs. Number of  
Clusters")  
abline(v = optimal_clusters_puma, col = "red", lty = 2)  
# Run PUMA-CLUST with the optimal number of clusters  
puma_result_optimal <- pumaClust(  
  e = expression_data,  
  se = measurement_error,  
  clusters = optimal_clusters_puma  
)  
# Extract cluster assignments  
puma_clusters <- puma_result_optimal$cluster  
cat("Cluster assignments:\n")  
print(table(puma_clusters))  
set.seed(123)  
# Perform model-based clustering using MCLUST  
mclust_result <- Mclust(expression_data, G=2:35)  
# Extract the optimal number of clusters based on BIC  
optimal_clusters_mclust <- mclust_result$G # Optimal number of clusters  
cat("Optimal number of clusters (MCLUST):", optimal_clusters_mclust, "\n")  
# Print model details  
summary(mclust_result)  
plot(mclust_result, what = "BIC")  
# Extract cluster assignments  
mclust_clusters <- mclust_result$classification  
cat("Cluster distribution (MCLUST):\n")
```



```
print(table(mclust_clusters))
# Print best model type
cat("Best covariance model:", mclust_result$modelName, "\n")
# Define the range of clusters
cluster_range_pgmm <- 2:35
# Run PGMM
pgmm_result <- pgmmEM(
  x = expression_data,    # Data matrix (genes x samples)
  rG = cluster_range_pgmm, # Range of clusters to test
  rq = 1:5,               # latent factors
  icl = FALSE,            # Use BIC for model selection
  zstart = 2,             # K-means initialization
  seed=123,
  loop = 1                # Single run
)
# Extract results
optimal_model_pgmm <- pgmm_result$model # Best model name
optimal_clusters_pgmm <- pgmm_result$g # Optimal number of clusters
optimal_factors_pgmm <- pgmm_result$q # Optimal number of factors
cluster_assignments_pgmm <- pgmm_result$map # Cluster assignments for each
gene
# Print results
cat("Best model:", optimal_model_pgmm, "\n")
cat("Optimal number of clusters:", optimal_clusters_pgmm, "\n")
cat("Optimal number of factors:", optimal_factors_pgmm, "\n")
cat("Cluster distribution:\n")
print(table(cluster_assignments_pgmm))
# Factor analytic mixture model (fabMix)
nChains <- 2
Kmax <- 35
qRange <- 1:2
mCycles <- 500
burnCycles <- 50
```



```
fabmix_result <- fabMix(nChains = nChains, Kmax = Kmax, mCycles = mCycles,  
burnCycles = burnCycles, q = qRange, rawData = expression_data, outDir =  
"tmp20")  
# Extract cluster assignments  
fabmix_clusters <- fabmix_results$class # This gives the cluster labels  
# Print cluster distribution  
print(table(fabmix_clusters))  
str(fabmix_result)  
# Extract the number of clusters identified  
optimal_clusters_fabmix <- length(unique(fabmix_result$class))  
cat("Optimal number of clusters:", optimal_clusters_fabmix, "\n")  
# Extract the selected model information  
selected_model_fabmix <- fabmix_result$selected_model  
print(selected_model_fabmix)  
# Convert clustering results to numeric vectors  
pgmm_labels <- as.numeric(cluster_assignments_pgmm)  
puma_labels <- as.numeric(puma_clusters)  
mclust_labels <- as.numeric(mclust_clusters)  
fabmix_labels <- as.numeric(fabmix_clusters)  
# Jaccard Similarity Function  
jaccard_similarity <- function(clusters1, clusters2) {  
  table_clusters <- table(clusters1, clusters2)  
  intersection <- sum(apply(table_clusters, 1, max))  
  union <- length(clusters1)  
  return(intersection / union)  
}  
# Compute Jaccard Similarity  
jaccard_puma_mclust <- jaccard_similarity(puma_labels, mclust_labels)  
jaccard_puma_fabmix <- jaccard_similarity(puma_labels, fabmix_labels)  
jaccard_mclust_fabmix <- jaccard_similarity(mclust_labels, fabmix_labels)  
# Compute Jaccard Similarity for PGMM Comparisons  
jaccard_pgmm_puma <- jaccard_similarity(pgmm_labels, puma_labels)  
jaccard_pgmm_mclust <- jaccard_similarity(pgmm_labels, mclust_labels)  
jaccard_pgmm_fabmix <- jaccard_similarity(pgmm_labels, fabmix_labels)
```



```
cat("\n--- Jaccard Similarity Scores ---\n")
cat("PUMA-CLUST vs. MCLUST:", jaccard_puma_mclust, "\n")
cat("PUMA-CLUST vs. FABMix:", jaccard_puma_fabmix, "\n")
cat("MCLUST vs. FABMix:", jaccard_mclust_fabmix, "\n")
cat("PGMM vs. PUMA-CLUST:", jaccard_pgmm_puma, "\n")
cat("PGMM vs. MCLUST:", jaccard_pgmm_mclust, "\n")
cat("PGMM vs. FABMix:", jaccard_pgmm_fabmix, "\n")
# Compute Silhouette Score
silhouette_score <- function(data, clusters) {
  sil <- silhouette(clusters, dist(data))
  mean(sil[, 3]) # Extract the average silhouette score
}
sil_puma <- silhouette_score(expression_data, puma_clusters)
sil_mclust <- silhouette_score(expression_data, mclust_clusters)
sil_fabmix <- silhouette_score(expression_data, fabmix_clusters)
sil_pgmm <- silhouette_score(expression_data_pgmm,
cluster_assignments_pgmm)
# Print Cluster Quality Metrics
cat("\n--- Silhouette Score ---\n")
cat("PUMA-CLUST:", sil_puma, "\n")
cat("MCLUST:", sil_mclust, "\n")
cat("FABMix:", sil_fabmix, "\n")
cat("PGMM:", sil_pgmm, "\n")
# Convert clustering results to integer vectors
puma_clusters_int <- as.integer(as.factor(puma_clusters))
mclust_clusters_int <- as.integer(as.factor(mclust_clusters))
fabmix_clusters_int <- as.integer(as.factor(fabmix_clusters))
pgmm_clusters_int <- as.integer(as.factor(cluster_assignments_pgmm))
# Compute Dunn Index (Extract first value from list)
dunn_index_puma <- unlist(intCriteria(expression_data, puma_clusters_int,
"Dunn"))
dunn_index_mclust <- unlist(intCriteria(expression_data, mclust_clusters_int,
"Dunn"))
```



```
dunn_index_fabmix <- unlist(intCriteria(expression_data, fabmix_clusters_int,
"Dunn"))
dunn_index_pgmm <- unlist(intCriteria(expression_data_pgmm,
pgmm_clusters_int, "Dunn"))
# Compute Davies-Bouldin Index (Extract first value from list)
db_index_puma <- unlist(intCriteria(expression_data, puma_clusters_int,
"Davies_Bouldin"))
db_index_mclust <- unlist(intCriteria(expression_data, mclust_clusters_int,
"Davies_Bouldin"))
db_index_fabmix <- unlist(intCriteria(expression_data, fabmix_clusters_int,
"Davies_Bouldin"))
db_index_pgmm <- unlist(intCriteria(expression_data_pgmm, pgmm_clusters_int,
"Davies_Bouldin"))
# Print Cluster Quality Metrics
cat("\n--- Dunn Index ---\n")
cat("PUMA-CLUST:", dunn_index_puma, "\n")
cat("MCLUST:", dunn_index_mclust, "\n")
cat("FABMix:", dunn_index_fabmix, "\n")
cat("PGMM:", dunn_index_pgmm, "\n")
cat("\n--- Davies-Bouldin Index ---\n")
cat("PUMA-CLUST:", db_index_puma, "\n")
cat("MCLUST:", db_index_mclust, "\n")
cat("FABMix:", db_index_fabmix, "\n")
cat("PGMM:", db_index_pgmm, "\n")
# Function to generate PCA plots
plot_pca <- function(expression_data, clusters, method_name) {
  # Run PCA
  pca_result <- prcomp(expression_data, scale. = TRUE) # No transposition needed
  # Extract principal components for genes
  pca_df <- as.data.frame(pca_result$x)
  # Ensure cluster labels match PCA output
  if (nrow(pca_df) != length(clusters)) {
    stop("Error: Cluster assignments and PCA points do not match in length!")
  }
}
```



```
# Add cluster labels
pca_df$Cluster <- as.factor(clusters)
# Plot PCA
ggplot(pca_df, aes(x = PC1, y = PC2, color = Cluster)) +
  geom_point() +
  theme_minimal() +
  ggtitle(paste("PCA of Clustered Data (", method_name, ")", sep = ""))
}
# Generate PCA plots for each method
plot_pca(expression_data, puma_clusters, "PUMA-CLUST")
plot_pca(expression_data, mclust_clusters, "MCLUST")
plot_pca(expression_data, fabmix_clusters, "FABMix")
plot_pca(expression_data, cluster_assignments_pgmm, "PGMM") # PGMM uses
1500 genes
# Function to generate 3D PCA plots for any clustering method
plot_3d_pca <- function(expression_data, clusters, method_name) {
  # Run PCA
  pca_result <- prcomp(expression_data, scale. = TRUE)
  pca_df <- as.data.frame(pca_result$x)
  # Add cluster labels
  pca_df$Cluster <- as.factor(clusters)
  # Define a larger color palette dynamically
  num_clusters <- length(unique(clusters))
  if (num_clusters <= 12) {
    colors <- brewer.pal(num_clusters, "Set3")
  } else {
    colors <- rainbow(num_clusters)
  }
  # 3D PCA plot
  fig <- plot_ly(pca_df, x = ~PC1, y = ~PC2, z = ~PC3, color = ~Cluster, colors =
colors, type = 'scatter3d', mode = 'markers') %>%
  layout(title = paste("3D PCA Plot of Clustered Data (", method_name, ")", sep
= ""))
}
```



```
    return(fig)
  }
# Generate 3D PCA plots for each method
fig_puma <- plot_3d_pca(expression_data, puma_clusters, "PUMA-CLUST")
fig_mclust <- plot_3d_pca(expression_data, mclust_clusters, "MCLUST")
fig_fabmix <- plot_3d_pca(expression_data, fabmix_clusters, "FABMix")
fig_pgmm <- plot_3d_pca(expression_data, cluster_assignments_pgmm,
"PGMM") # PGMM uses 1500 genes
# Display results
fig_puma
fig_mclust
fig_fabmix
fig_pgmm
```

#### **Application 2 Chapter 4**

```
#Leukemia data
# Load the Golub dataset
library(EMMIXgene)
data(golub_data)
# Load required libraries
library(tidyverse) # For data manipulation
library(Rtsne) # For t-SNE
library(ggplot2) # For enhanced visualizations
library(limma)
library(pgmm)
library(mclust)
library(puma)
library(cluster) # For silhouette and k-medoids
library(factoextra)
library(fpc) # For clustering validation
library(clValid)
library(clusterSim) # For Davies-Bouldin Index
library(ggplot2) # For plotting
library(gridExtra) # For arranging multiple plots
# Check dimensions
```



```
print(dim(golub_data)) # 3731 genes x 72 samples
# Create group labels for ALL (1) and AML (2)
group_labels <- factor(c(rep(1, 47), rep(2, 25))) # 47 ALL, 25 AML
design <- model.matrix(~ 0 + group_labels) # Create design matrix
colnames(design) <- c("ALL", "AML") # Name the groups
# Fit linear model
fit <- lmFit(golub_data, design)
# Create contrast (AML vs ALL)
contrast <- makeContrasts(AML - ALL, levels = design)
# Fit the contrast model
fit2 <- contrasts.fit(fit, contrast)
fit2 <- eBayes(fit2)
# Select the top 1000 differentially expressed genes based on adjusted p-value
significant_genes <- topTable(fit2, adjust = "BH", number = 1000) # Select 1000
DEGs
# Extract the row indices of the top genes
top_gene_indices <- as.numeric(rownames(significant_genes))
# Ensure rownames of golub_data match numeric indices
rownames(golub_data) <- 1:nrow(golub_data)
# Subset golub_data using the selected top 1000 gene indices
subset_data <- golub_data[top_gene_indices, ]
# Check dimensions
print(dim(subset_data)) # Should be 1000 genes x 72 samples
# Transpose for clustering
reduced_data <- t(subset_data)
# Check the dimensions of the reduced dataset
print(dim(reduced_data)) # 72 samples x 1000 genes
# Apply PGMM clustering
set.seed(42)
pgmm_result <- pgmmEM(
  x = reduced_data,      # Transposed reduced dataset
  rG = 2:4,             # Number of clusters
  rq = 1:4,             # Range of factors
  zstart = 1,          # Use random initialization
```



```
loop = 1,          # Random starts
icl = FALSE,      # Use BIC for model selection
seed = 123456,    # Seed for reproducibility
tol = 0.1        # Convergence tolerance
)
# Extract cluster assignments from pgmm_result
predicted_labels_pgmm <- pgmm_result$map
# Display the cluster counts
print(table(predicted_labels_pgmm)) # Shows how many samples are in each
cluster
pgmm_model <- pgmm_result$model
clusters_pgmm <- pgmm_result$g # Optimal number of clusters
factors_pgmm <- pgmm_result$q # Optimal number of factors
# Print results
cat("Best model:", pgmm_model, "\n")
cat("Optimal number of clusters:", clusters_pgmm, "\n")
cat("Optimal number of factors:", factors_pgmm, "\n")
# Define true labels (1 = ALL, 2 = AML)
true_labels <- c(rep(1, 47), rep(2, 25))
# Run MCLUST clustering
set.seed(42) # Ensure reproducibility
mclust_result <- Mclust(reduced_data)
# Display summary of the model
summary(mclust_result)
# Extract cluster assignments
mclust_labels <- mclust_result$classification
print(table(mclust_labels)) # Check how many samples are in each cluster
# Apply PUMA clustering
# Define the range of clusters to test
cluster_range_puma <- 2:5
# Initialize storage for BIC values
bic_values <- numeric(length(cluster_range_puma))
set.seed(123)
# Loop over cluster numbers and run PUMA-CLUST
```



```
for (k in cluster_range_puma) {
  cat("Running PUMA-CLUST with", k, "clusters...\n")
  # Run pumaClust
  puma_result <- pumaClust(
    e = reduced_data,
    clusters = k
  )
  # Store the BIC value
  bic_values[k - 1] <- puma_result$bic
}
# Identify the optimal number of clusters
optimal_clusters_puma <- cluster_range_puma[which.min(bic_values)]
cat("Optimal number of clusters:", optimal_clusters_puma, "\n")
set.seed(42) # For reproducibility
puma_result_Optimal <- pumaClust(
  e = reduced_data, # Gene expression matrix
  clusters = 3,     # Number of clusters
  iter.max = 100,   # Maximum iterations for parameter initialization
  nstart = 10,      # Number of random initializations
  eps = 1.0e-6,     # Convergence tolerance
)
print(summary(puma_result_Optimal))
# Extract cluster assignments
puma_clusters <- puma_result_Optimal$cluster
cat("Cluster assignments:\n")
print(table(puma_clusters))
set.seed(42)
# Select genes using EMMIX-GENE
selected_genes <- select_genes(dat = subset_data) # Filter based on variability
print(sum(selected_genes$selected)) # Check the number of selected genes
# Cluster selected genes into 2 clusters (biological groups or patterns)
gene_clustering <- cluster_genes(
  gen = selected_genes, # Object returned by select_genes()
  g = 2                 # Number of clusters
)
```



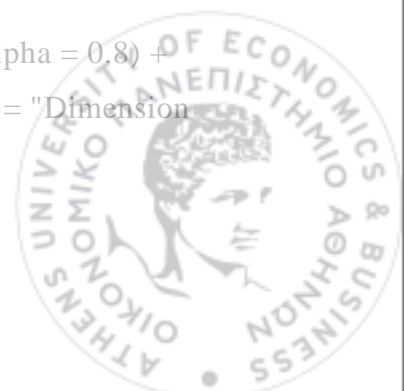
```
)  
# View the clustering results  
print(table(gene_clustering)) # Distribution of genes across clusters  
# Use the selected genes object to cluster tissues  
tissue_clustering <- cluster_tissues(  
  gen = selected_genes, # Selected genes object  
  clusters = gene_clustering, # Results from cluster_genes()  
  q = 2, # Number of latent factors  
  G = 2 # Number of clusters (e.g., ALL vs. AML)  
)  
# View tissue clustering results  
# Extract cluster assignments from the tissue clustering matrix  
predicted_labels_emmix <- apply(tissue_clustering, 2, which.max) # Assign based  
on max probability  
# Print the assigned clusters  
print(table(predicted_labels_emmix)) # Count of samples in each cluster  
# Compute silhouette score  
silhouette_score <- function(data, labels) {  
  dist_matrix <- dist(data) # Compute distance matrix  
  sil <- silhouette(labels, dist_matrix)  
  avg_sil <- mean(sil[, 3]) # Mean silhouette score  
  return(avg_sil)  
}  
# Compute Davies-Bouldin Index  
davies_bouldin_score <- function(data, labels) {  
  dbi_value <- index.DB(data, labels)$DB  
  return(dbi_value)  
}  
dunn_score <- function(data, labels) {  
  return(cIValid::dunn(Data = data, clusters = labels))  
}  
# Compute validation metrics for PGMM  
sil_pgmm <- silhouette_score(reduced_data, predicted_labels_pgmm)  
dbi_pgmm <- davies_bouldin_score(reduced_data, predicted_labels_pgmm)
```



```
dunn_pgmm <- dunn_score(reduced_data, predicted_labels_pgmm)
# Compute validation metrics for MCLUST
sil_mclust <- silhouette_score(reduced_data, mclust_labels)
dbi_mclust <- davies_bouldin_score(reduced_data, mclust_labels)
dunn_mclust <- dunn_score(reduced_data, mclust_labels)
# Compute validation metrics for PUMA-CLUST
sil_puma <- silhouette_score(reduced_data, puma_clusters)
dbi_puma <- davies_bouldin_score(reduced_data, puma_clusters)
dunn_puma <- dunn_score(reduced_data, puma_clusters)
# Compute validation metrics for EMMIX-GENE
sil_emmix <- silhouette_score(reduced_data, predicted_labels_emmix)
dbi_emmix <- davies_bouldin_score(reduced_data, predicted_labels_emmix)
dunn_emmix <- dunn_score(reduced_data, predicted_labels_emmix)
# Create a summary table
internal_metrics <- data.frame(
  Method = c("PGMM", "MCLUST", "PUMA", "EMMIX-GENE", "k-Means", "k-
Medoids", "Hierarchical (Complete)", "Hierarchical (Average)", "Hierarchical
(Single)"),
  Silhouette = c(sil_pgmm, sil_mclust, sil_puma, sil_emmix)
  DBI = c(dbi_pgmm, dbi_mclust, dbi_puma, dbi_emmix),
  Dunn = c(dunn_pgmm, dunn_mclust, dunn_puma, dunn_emmix),
)
# Print results
print(internal_metrics)
# Compute ARI for PGMM
ari_pgmm <- adjustedRandIndex(predicted_labels_pgmm, true_labels)
# Compute ARI for MCLUST
ari_mclust <- adjustedRandIndex(mclust_labels, true_labels)
# Compute ARI for PUMA-CLUST
ari_puma <- adjustedRandIndex(puma_clusters, true_labels)
# Compute ARI for EMMIX-GENE
ari_emmix <- adjustedRandIndex(predicted_labels_emmix, true_labels)
# Create a summary table of ARI scores
ari_results <- data.frame(
```



```
Method = c(
  "PGMM", "MCLUST", "PUMA-CLUST", "EMMIX-GENE",),
ARI_Score = c(
  ari_pgmm, ari_mclust, ari_puma, ari_emmix)
)
# Print ARI scores for all methods
print(ari_results)
# Compute PCA for the dataset
pca_result <- prcomp(reduced_data, center = TRUE, scale. = TRUE)
# Create PCA dataframe with true labels
pca_df <- data.frame(PC1 = pca_result$x[,1], PC2 = pca_result$x[,2], True_Label
= factor(true_labels))
# Compute t-SNE
set.seed(42)
tsne_result <- Rtsne(reduced_data, dims = 2, perplexity = 20)
# Create t-SNE dataframe with true labels
tsne_df <- data.frame(Dim1 = tsne_result$Y[,1], Dim2 = tsne_result$Y[,2],
True_Label = factor(true_labels))
plot_clusters_with_truth <- function(df, method_name, cluster_labels) {
  df$Cluster <- factor(cluster_labels) # Add predicted clusters
  # PCA Plot (True Labels vs. Clusters)
  pca_plot <- ggplot(df, aes(x = PC1, y = PC2)) +
    geom_point(aes(color = True_Label, shape = Cluster), size = 3, alpha = 0.8) +
    labs(title = paste("PCA -", method_name), x = "PC1", y = "PC2") +
    theme_minimal() +
    scale_color_manual(values = c("red", "blue")) + # True labels color
    theme(plot.title = element_text(hjust = 0.5, face = "bold"))
  # t-SNE Plot (True Labels vs. Clusters)
  tsne_df$Cluster <- factor(cluster_labels) # Assign predicted clusters
  tsne_plot <- ggplot(tsne_df, aes(x = Dim1, y = Dim2)) +
    geom_point(aes(color = True_Label, shape = Cluster), size = 3, alpha = 0.8) +
    labs(title = paste("t-SNE -", method_name), x = "Dimension 1", y = "Dimension
2") +
    theme_minimal() +
```



```
scale_color_manual(values = c("red", "blue")) + # True labels color
theme(plot.title = element_text(hjust = 0.5, face = "bold"))
# Return both plots arranged side-by-side
return(grid.arrange(pca_plot, tsne_plot, ncol = 2))
}
plot_clusters_with_truth(pca_df, "PGMM", predicted_labels_pgmm)
plot_clusters_with_truth(pca_df, "MCLUST", mclust_labels)
plot_clusters_with_truth(pca_df, "PUMA-CLUST", puma_clusters)
plot_clusters_with_truth(pca_df, "EMMIX-GENE", predicted_labels_emmix)
```

## ***References***

**Benjamini, Y. and Hochberg, Y. (1995).** Controlling the false discovery rate: a practical and powerful approach to multiple testing, *Journal of the Royal Statistical Society: Series B (Methodological)*, 57, 289–300.

**Broberg, P. (2003).** Statistical methods for ranking differentially expressed genes, *Genome Biology*, 4, R41.

**Dudoit, S. and Fridlyand, J. (2003).** Classification in microarray experiments, *Statistical Analysis of Gene Expression Microarray Data*, 1, 93–158.

**Dudoit, S., Shaffer, J. P., and Boldrick, J. C. (2003).** Multiple hypothesis testing in microarray experiments, *Statistical Science*, 71–103.

**Efron, B., Tibshirani, R., Storey, J. D., and Tusher, V. (2001).** Empirical Bayes analysis of a microarray experiment, *Journal of the American Statistical Association*, 96, 1151–1160.

**Ghosh, D. and Chinnaiyan, A. M. (2002).** Mixture modelling of gene expression data from microarray experiments, *Bioinformatics*, 18, 275–286.

**Lönnstedt, I. and Speed, T. (2002).** Replicated microarray data, *Statistica Sinica*, 31–46.

**McLachlan, G. J., Do, K.-A., and Ambrose, C. (2005).** Analyzing microarray gene expression data.

**McNicholas, P. D. and Murphy, T. B. (2010).** Model-based clustering of microarray expression data via latent Gaussian mixture models, *Bioinformatics*, 26, 2705–2712.

**Pan, W., Lin, J., and Le, C. T. (2002).** Model-based cluster analysis of microarray gene-expression data, *Genome Biology*, 3, 1–8.



**Ritchie, M. E., Phipson, B., Wu, D., Hu, Y., Law, C. W., Shi, W., and Smyth, G. K. (2015).** limma powers differential expression analyses for RNA-sequencing and microarray studies, *Nucleic Acids Research*, 43, e47–e47, URL: <https://doi.org/10.1093/nar/gkv007>

**Schena, M., Shalon, D., Davis, R. W., and Brown, P. O. (1995).** Quantitative monitoring of gene expression patterns with a complementary DNA microarray, *Science*, 270, 467–470.

**Smyth, G. K. (2004).** Linear models and empirical Bayes methods for assessing differential expression in microarray experiments, *Statistical Applications in Genetics and Molecular Biology*, 3.

**Stephens, M. (2017).** False discovery rates: a new deal, *Biostatistics*, 18, 275–294.

**Storey, J. D. (2002).** A direct approach to false discovery rates, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64, 479–498.

**Storey, J. D. (2003).** The positive false discovery rate: a Bayesian interpretation and the q-value, *Annals of Statistics*, 31, 2013–2035.

**Storey, J. D. and Tibshirani, R. (2003).** Statistical significance for genomewide studies, *Proceedings of the National Academy of Sciences*, 100, 9440–9445.

**Wei, C., Li, J., and Bumgarner, R. E. (2004).** Sample size for detecting differentially expressed genes in microarray experiments, *BMC Genomics*, 5, 87.

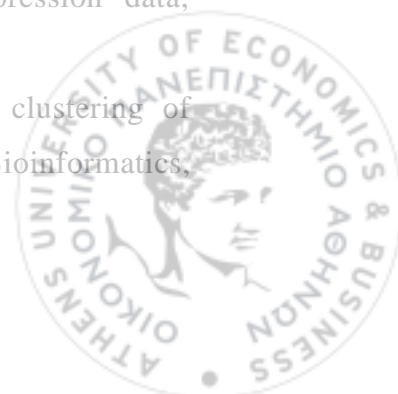
**Yip, W.-K., Amin, S. B., and Li, C. (2011).** A survey of classification techniques for microarray data analysis, in *Handbook of Statistical Bioinformatics*, Springer, 193–223.

**Liu, X., Lin, K. K., Andersen, B., and Rattray, M. (2007).** Including probe-level uncertainty in model-based gene expression Clustering, *BMC Bioinformatics*, 8, 98.

**Liu, X., Lin, K. K., Andersen, B., and Rattray, M. (2007).** Including probe-level uncertainty in model-based gene expression Clustering, *BMC Bioinformatics*, 8, 98.

**Yeung, K. Y., Fraley, C., Murua, A., Raftery, A. E., and Ruzzo, W. L. (2001).** Model-based clustering and data transformations for gene expression data, *Bioinformatics*, 17(10), 977–987.

**McNicholas, P. D., and Murphy, T. B. (2010).** Model-based clustering of microarray expression data via latent Gaussian mixture models, *Bioinformatics*, 26(21), 2705–2712.



**Ghosh, D., and Chinnaiyan, A. M. (2002).** Mixture modelling of gene expression data from microarray experiments, *Bioinformatics*, 18(2), 275–286.

**Hochberg, Yosef. 1988.** A Sharper Bonferroni Procedure for Multiple Tests of Significance, *Biometrika* 75 (4): 800–802

**Lex, A., Gehlenborg, N., Strobel, H., Vuillemot, R., & Pfister, H. (2014).** UpSet: Visualization of intersecting sets. *IEEE Transactions on Visualization and Computer Graphics*, 20(12), 1983–1992.

