Abstract

RNA-sequencing (RNA-seq) is a sequencing technique used to study gene expression in biological samples. It has fueled many discoveries in recent years and brought the study of differentially expressed genes (DEG) to the forefront of biomedical research. However, there is no consensus on the way to analyze RNA-seq experimental data, and many tools using different methods are available.

Differential Gene Expression (DGE) analysis pipelines are used to identify varying gene expressions between experimental groups. In this report, we have outlined implementations of edgeR, DESeq2, limma+voom, and NOISeq to study DGE on a synthetic dataset. Using the recommended manual settings, edgeR consistently discovered the most DEG, with 7,974 across the whole dataset, compared to 4,513 for DESeq2, 2,802 for limma+voom, and 2,350 for NOISeq. We observed a high level of agreement between the tools, with 57% of all DEG being detected by at least three tools.

Since the raw number of DEG gives no indication of the true nature of the difference, we studied the sensitivity and specificity of each method. Overall, we observed that edgeR found the highest proportion of DEG (sensitivity of 57.7% vs. DESeq2: 44.5% vs. NOISeq: 18.6% and limma+voom: 29.7%), but at the cost of more false positive instances (specificity of 96.6 vs. DESeq2: 99.2% vs. NOISeq: 99.2% and limma+voom: 99.8%).

For all tools except NOISeq, increasing the number of samples increased the total number of genes discovered and the sensitivity (DESeq: ×2, edgeR: ×1.43, NOIseq: ×0.77, limma: ×12.82). All tools performed the same on upregulated genes vs. downregulated genes.

Introduction

Genomics is the study of the genetic sequence information of organisms to understand the function of these sequences and downstream possible products. The genetic material, DNA, is shared among all cells within an individual. Nevertheless, cells do not share the same functions and express diverse proteins to fulfil their respective roles. DNA is transcribed into RNA, which, in turn, is translated into proteins. The study of RNA expression can thus provide insights into cellular biology. RNA levels exhibit correlations with protein expression and serve as a valuable tool for the examination of gene expression.

Since the inception of sequencing in the 1970s, there has been a substantial increase in both the quantity and speed at which scientists were capable of sequencing genomes. In recent years, the development of sequencing technology has led to the production of huge datasets. Those new techniques required the development of new computational tools to study biological significance. Some methodologies, such as single-cell RNA sequencing (single-seq RNA-seq), require a minimum of 10 million reads per sample. Its primary application lies in the investigation of gene expression. RNA levels exhibit correlations with protein expression and serve as a valuable tool for the examination of gene expression. Differential gene expression (DGE) refers to the examination of how gene expression (or activity) varies across different conditions, tissues, developmental stages, or individuals[1] . Here we will provide an overview and comparison of some tools used in the study of DGE.

Although the utility and importance of RNA-seq have grown, there is no consensus on the best way to analyse RNA-seq data. A primary challenge stems from the limited number of samples available for such experiments. Various factors other than biological ones might influence the number of detected sequences, including experimental conditions, sequence length, and individual variability.

DGE pipeline analyses RNA-seq data with a series of steps. Raw reads are aligned to reference the genome, such as Bowtie2[2]. Aligned reads are assigned to genes and annotated using tools such as featureCounts. The resulting count data then need to be normalized to enable comparison of gene expression between samples. Different normalisation methods exist to correct for technical biases caused by gene length, sequencing batches, or other protocol artifacts. The normalization step helps to compare the sample, eliminate biases, and to filter out low read count data which is usually attributed to noise in the data. Normalised counts are then analysed using statistical models to identify differentially expressed genes (DEG) [3]. Models assume count distributions follow a particular distribution such as negative binomial, Log-Normal, or empirical Bayes. Differentially expressed genes are identified using a threshold for the probability of being differentially expressed. In a biological context, small changes of expression between conditions are not necessarily reflective of any change

in the cell. That's why it is also recommended to also set a threshold for expression changes (LogFoldChange).

Previously, it has been shown that DGE analysis is heavily influenced by statistical models, and there is no consensus as to which DGE methodology provides the best results. Comparing the DGE pipeline is complicated by the absence of gold standard datasets: with biological samples, the ground truth is never known, and we are unable to identify gene expression patterns. Studies utilize synthetic reads derived *in silico*[4] .

Model performances can be assessed based on multiple criteria, some studies have looked at the total number of DEG found, but this does not consider the proportion of false positive genes discovered (FDR). Another method to analyse performance is to rank DGE methods based on concordance with other models. It is thus critical to develop a method to compare DGE analysis tools.

Given the wide range of factors that can affect results, we investigated four DGE pipelines for gene-level analysis: DESeq2, edgeR, NOISeq, and voom-limma. To assess the performance of each model based on metrics other than the total number of differentially expressed genes, we utilized a dataset composed of count matrices from nine synthetic experiments.

In the initial phase of our study, we evaluated the effectiveness of each tool in identifying Differentially Expressed Genes (DEG). Our focus was on examining the total number of DEG detected by each tool under various experimental conditions. Specifically, we aimed to investigate factors that might improve reproducibility, such as filtering out genes with low expression, genes where the difference in expression was deemed non-significant and varying the number of samples.

To facilitate a comparison among the tools, we analysed the extent of agreement in their outcomes, particularly in terms of identifying genes as DEG. We explored whether different tools were identifying distinct sets of genes as DEG.

Following this initial assessment, we proceeded to evaluate the classification performance of these tools. Our evaluation criteria included measuring the sensitivity, which quantifies the proportion of truly Differentially Expressed Genes (DEG) correctly identified, and the specificity, which indicates the proportion of genes that are not differentially expressed but were accurately recognized as such.

Additionally, we investigated how varying sample sizes and the proportion of upregulated DEG influenced the ultimate results.

Literature Review

Fatemeh, Asta & Laura 2013[5]performed a systematic comparison of eight Bioconductor software packages and pipelines for detecting differentially expressed genes between conditions and has given some guidelines regarding how to choose a robust pipeline. From their study it was found that there can be differences in pipelines and there is no single tool/pipeline that performs best under all circumstances. They also acknowledge the differences in the documentation quality and detail of the pipelines; Packages like DEseq, edgeR, limma have very comprehensive manuals and user-guides with practical examples, whereas NOISeq provides very limited instructions and parameter value descriptions.

A. Stupnikov et al. 2021 [4] investigated five software packages DESeq2, voom + limma, edgeR, EBSeq, and NOISeq for gene-level detection to compare robustness to sequencing alterations using a controlled analysis of fixed count matrices. Robustness of different DGE tools was compared between filtering regimes and for different expression levels (high, low) using unbiased metrics. They also examined sensitivity as relative False Discovery Rate (FDR), concordance between DGE results and comparisons of a 'population' of slopes of relative FDRs across different library sizes generated using linear regressions, etc. Results from the study helped to identify reliable workflows at different library sizes for genes with different expression levels, and also provided important guidelines for DGE tool package selection for molecular diagnostics.

Alyssa et al. 2018 [6] evaluated the impact of different read depth(total number of reads obtained for each sample) and sample level on the performance of DGE packages, based on precision, or the fraction of genes correctly identified as differentially expressed, and by recall, or the fraction of differentially expressed genes identified. They analysed 30 high-performing DGE workflows, with different read depth and number of biological samples of patient monocyte samples provided as input. They found that read depth had little effect on the performance of most of the workflows when the read depth was above two million reads per sample, however performance was reduced below this threshold for the workflows. This reduced performance was particularly significant for inputs with less than seven samples per group, when more heterogeneity in workflow performance is observed. They also suggest that the selection of DGE tools, in particular, has a large impact on the response to limited inputs.

Material and Methods

Software Packages Implementation

DGE analysis was conducted using R (4.3.1) software Bioconductor(3.17) packages namely edgeR, DESeq2, Limma, and NOISeq. Package implementations were based on the developer manuals, which involved using default parameters and normalization methods as per the instructions provided [3]. Versions, normalisation methods, and differential expression tools used in the packages are listed in Table 1.

edgeR uses a negative binomial model based on empirical Bayes estimation and exact tests for performing differential expression analysis. edgeR works best with a small number of replicates. Overdispersion between genes is moderated by sharing information between genes which conforms to the empirical Bayes method. The exact test used for differential expression testing is similar to Fisher's exact test, though modified to suit over-dispersed data in this context. The default TMM normalization method was used to account for the varying sequence depth among samples. The false positive rate (FPR) was regulated by the Benjamini–Hochberg procedure[5].

DESeq2 also uses a negative binomial model for read count assumption similar to edgeR, however, dispersion estimates are modelled on the mean, and variance relationships observed. This data-driven parametric approach helps to even up the selection of differentially expressed genes from a wide range of data. DESeq2 also uses a size factor normalisation approach and Benjamini–Hochberg procedure-based FDR controlling similarly to edgeR. DESeq2 works well with fewer replicates and can even work without any biological replicates; however, it is not recommended[5].

NOIseq is a "data adaptive non-parametric method, which empirically models the noise distribution from the actual data by contrasting fold-change differences and absolute expression differences among samples within the same condition"[5]. The FPR is regulated by adapting to data size and the default normalization process is RPKM (reads per kilobase per million mapped reads). The shrinking of variance estimates in NOISeqBIO involves combination of a non-parametric model and empirical Bayes approach irrespective of sample number. However, null distribution is created in a different method when the number of samples per condition is less than five and NOISeqBIO demonstrates unusual behaviour at these sample levels. NOISeqBIO employs, k-means clustering at these lower sample levels to identify genes with similar expression patterns and information is shared between these genes when creating the null distribution [6]. K-means clustering is not performed at higher sample levels. This deference should be taken into consideration when analysing NOISeqBIO's performance for different number of sample levels. The probability of differential expression is equivalent to 1 − FDR in NOISeqBIO where FDR is similar to adjusted p-value, so the developers

recommend for q to use values around 0.95 when identifying differentially expressed genes[3]. Sonia et al. 2015 [7] also suggest that the FDR defined by Benjamini and Hochberg is closely connected to the *a posteriori* probability $p0(z) = 1 - p1(z)$ calculated by NOIseq. Therefore, $p1(z) = 1 - FDR$ and so $1 - p1(z)$ can be considered as an adjusted *P*-value. This should be taken into consideration when using NOISeq that DE probability returned by the tool is not equivalent to a *P*-value. The CPM value per sample was set to one for low count filtering as recommended by developers in order to be conservative and not exclude too many genes.

Limma package[8] was originally developed for differential expression analysis of microarray. Later it was extended to RNA seq by adding the function voom. Limma package uses linear models to assess differential expression. A Study from Ritchie *et al* [8] mentions that Limma accepts RNA-seq data in the form of a matrix of read counts, with rows for genomic features and columns for RNA samples. However, the current user guide suggests passing the RNA-seq data as a DGEList object from the edgeR package. The read counts are processed by the voom function in Limma to convert them into log2 counts per million (logCPM) with associated precision weights. These precision weights are estimated using their mean–variance relationship. The logCPM values can be normalized between samples by the voom function or can be pre-normalized by adding normalization factors within edgeR. The current recommendation according to the Limma user guide is to use TMM normalization of the edgeR package before passing the read counts into the voom function. By default, the Benjamini–Hochberg procedure is used to estimate the FDR, and the Empirical Bayes method is used for differential expression tests.

Table 1 Software Packages for Detecting Differential Expression, Adapted from[2]

Method	Version	Normalization	Read count distribution assumption	Statistical Test
edgeR	3.42.4	TMM	Negative binomial distribution	Exact test
DESeq2	1.40.2	DESeq sizeFactors	Negative binomial distribution	Exact test
NOIseq	2.42.0	RPKM	Nonparametric method	Contrasts fold changes and absolute differences within a condition to determine the null distribution and then compares the observed differences to this null.
Limma	3.56.2	TMM	voom transformation of counts	Empirical Bayes method

Dataset

A total of 9 files were used where each file contains a synthetic count matrix that represents a theoretical RNA-seq experiment with two conditions. All files have the same genes but different numbers of biological replicates and ratios of up and down-regulated genes as outlined in Table 2.

The filenames are structured as n_up_down.tsv, where:

- n is the number of samples per condition. The first n samples are condition 1, the next n samples are condition 2.
- up is the number of genes that are upregulated in condition 2 compared to condition 1.
- down is the number of genes that are down-regulated in condition 2 compared to condition 1.

For instance, 3_500_500.tsv corresponds to an experiment with 3 samples in each condition, 500 upregulated genes, and 500 downregulated genes. For each of the experiments, a metadata file (3_500_500_meta.tsv, etc.) was also provided that lists, for each gene, whether the gene is differentially expressed (and if it is, whether it is upregulated or downregulated). Meta files were used to measure the different types of efficiency matrices such as sensitivity, specificity, etc.

Table 2 Experiment Design

Parameter	***Design***
Conditions	*2*
Samples/condition	*3/6/9*
Genes	*10 000*
Differentially expressed genes	*1000*
Proportion of upregulated gene	*50%/75%/100%*

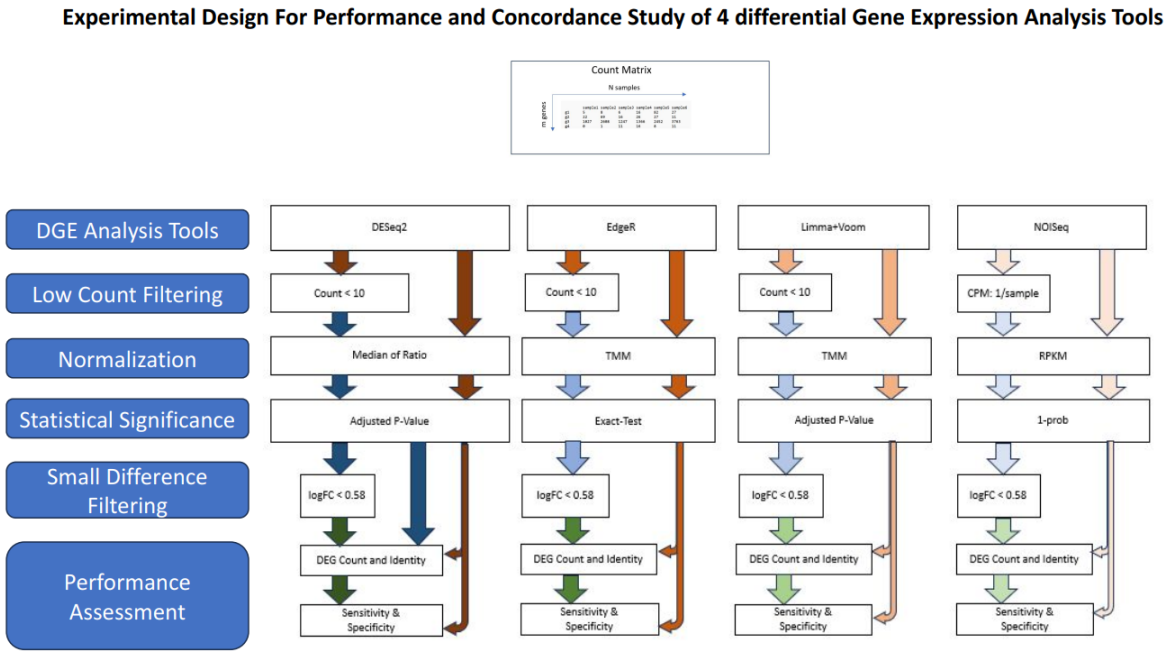


Figure 1: Study Design Overview The study involves the analysis of input data in the form of count matrices using four distinct Differential Gene Expression analysis tools. Our analysis includes comparing the total number of Differentially Expressed Genes (DEG) detected by each tool, both with and without the application of low count filtering, as well as with and without filtering for minor changes in gene expression. Following best practices, which involve the removal of low counts and implementing a Log-Fold Change (LFC) threshold of 0.58, we then conducted a comparative assessment of sensitivity and specificity results across these different tools. Furthermore, we investigated the influence of sample size and the ratio of upregulated DEG on the outcomes of our analysis.

Read Count Data for DGE analysis

The starting point of our analysis was the counts matrix, with a line for each gene i and a column for each sample j. The matrix entry K_{ij} represents the number of sequence reads that have been unambiguously mapped to genes in the sample.

Low-count filtering

For DESeq2, genes with counts below 10 across all samples of a given condition were filtered. NOIseq uses CPM (method 1: counts per million) to filter genes with low expression counts. In a condition with s samples, the cutoff is set as CPM × s, and those genes are removed where the expression counts were below the cutoff threshold in all conditions. The cutoff for the coefficient of variation (in percentage) was set to 100 to eliminate features with inconsistent expression values[3]. limma also uses CPM to filter genes with low expression counts and is implemented using filterByExpr function in the edgeR package. Genes with 10 or more read counts are kept by default in a minimum number of samples, with the number of samples selected based on the minimum group sample size[9].

Concordance of DEG pipeline outputs

DGE pipelines were compared for unfiltered (no-fold) and 1.5-fold filtered DEGs. For each sample size and each tool, the total number of DEGs was aggregated across all experiments. Each data point in the measurement of the effect of log-fold change and low count filtering represents the aggregate results of experiments with a 1:1, 2:1, and 1:0 ratio of upregulated vs. downregulated genes, totalling 30,000 genes and 3,000 DEGs. To compare the output of different tools, we examined the identity of the genes labelled by each tool and investigated how sample size and log-fold filtering affected the results.

We used a single experiment with a 1:1 ratio for each condition, as no effect of the ratio on the outcome was detected. We present results for 3 and 9 samples because most tools performed best in terms of the number of DEGs with 9 samples, but NOISeq performed better with 3. In all conditions, filtering for low count was applied.

Theoretical Classification Performance of DEG pipelines

Sensitivity measures the probability of genes identified as DE being truly DE also known as 'True Positives'. Specificity is the probability gene identified as not DE being not DE also known as 'True Negatives'.

$$sensitivity = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

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

Determining Differentially Expressed Genes

The count matrices were subjected to Differential gene expression analysis using the aforementioned Bioconductor packages. Differentially expressed genes were identified from different packages using the FDR value of 0.05 for all packages except NOIseq, where 1-prob was used as the threshold with rate of 0.05 as NOIseq does not have an FDR component as recommended by method. An absolute

log2 fold change exceeding a threshold of 0.58 was also used for filtering genes that exhibited a 1.5-fold or greater change in gene expression.

Estimating Time Efficiency

Runtimes were compared between different packages for 3, 6, and 9 samples. The runtime was computed using a Windows system with 1.1 GHz CPU, and 8 GB RAM configuration. Each experiment was analysed 10 times by each package, for a total of 30 repetitions for each sample size. Pipelines were tested from the start to the identification of differentially expressed genes.

Results and Discussion

Count and Agreement Study

Sensibility to low count removal

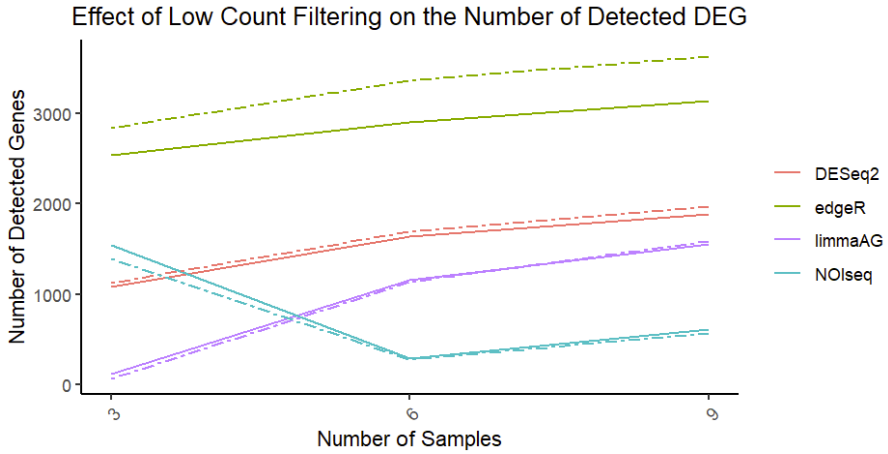


Figure 2 : Effect of Removing genes with low total counts. *We present the total number of DEG across all experiments for a given sample size, 30 000 genes total. The dash line (----) represents the number of DEG in the unfiltered dataset for each differential gene expression analysis tool.*

Figure 2 depicts the total number of differentially expressed genes (DEGs) in all experiments for a given sample size. As expected, with most packages, the number of detected DEG increased with the number of replicates. However, with NOISeq, we observed a noticeable decrease from around 1500 detections with 3 replicates, to 300 with 6 replicates. This might be because when performing low-count filtering, NOISeq considers the experimental design and removes features below the threshold from all the conditions in the dataset and similar results were observed by Fatemeh, Asta & Laura 2013 [5]. The count results are presented in Table I in the appendix. Except for edgeR, implementing the method prescribed by the developer to remove low counts had little effect on the number of discovered DEGs. Regardless of the sample size, 10% fewer DEGs were found by edgeR after removing genes with low counts. This might indicate more robustness to low counts for limma, DESeq2, and NOISeq. Those results are in contradiction with results from a previous study [10], where removing between 15% and 30% of low-expression genes increased the number of DEG. In our study, the number of genes removed varied widely depending on the tool, and at most 10% of the genes were

removed. Our goal was not to find the best parameter, but rather explore what the results would be if we used the default parameters for each tool. However, tools replicate the results cited previously, we could look at different methods of low-gene expression removal to obtain better results.

Sensibility to Small changes in Expression

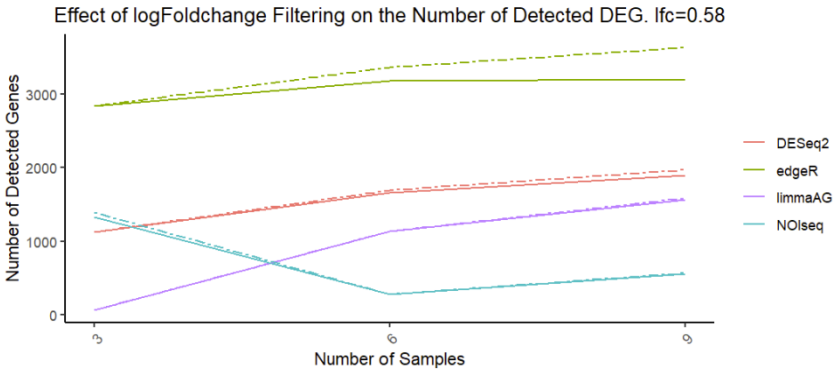
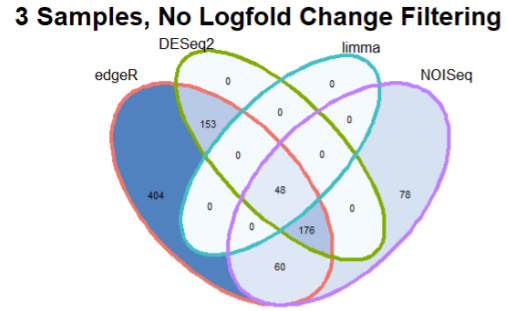
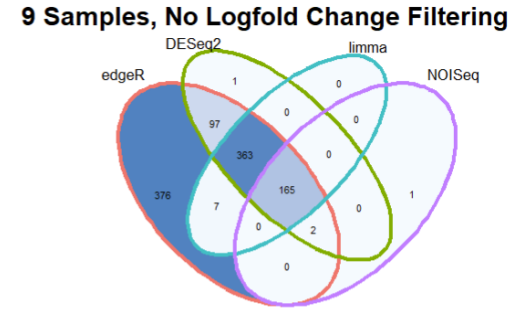


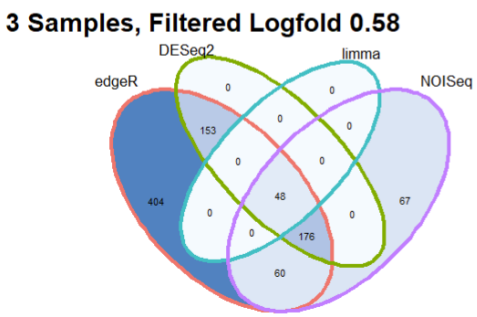
Figure 3 : Effect of filtering for small changes in expression between conditions*. We present the total number of DEG across all experiments for a given sample size, 30 000 genes total. The dash line(----) represents the number of DEG in the unfiltered dataset for each differential gene expression analysis tool. A gene with a difference in expression lower than 1.5 between conditions was considered not DEG.*

As for genes with low counts, detecting only those with an absolute expression difference of 1.5 times between condition 1 and 2 had no impact on the number of discovered DEGs, except in the case of edgeR. With 6 and 9 samples, 5% and 12% fewer genes were found. This might indicate that, by default, edgeR is more sensitive to lower levels of difference.

Agreement Between Tools







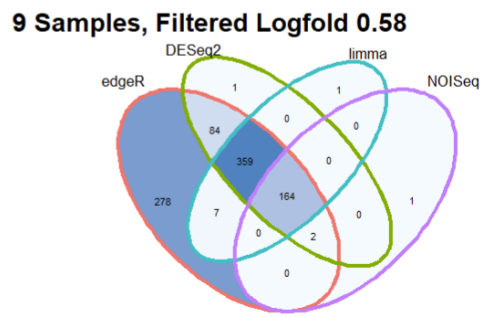


Figure 4 : Venn diagrams presenting the number of genes commonly identified with different combinations of tools. *The values are the number of DEG genes. We studied the experiments with a ration of 1:1 of upregulated vs downregulated genes, for each sample size. All datasets were filtered for low counts, and included around 10 000 genes with 10% of DEG..*

In Figure 4, we selected the highest and lowest numbers of samples to investigate whether the tools agreed on which genes were differentially expressed. Using the recommended settings, including filtering for a logfold change under 0.58, we observed no differences in results with 3 samples. Specifically, with 3 samples, 919 unique DEGs were found, compared to 908 after LFC filtering. All tools agreed on 48 genes, and edgeR, DESeq2, and NOISeq agreed on an additional 176 genes. Notably, edgeR and NOISeq discovered 404 and 67 unique genes, respectively.

The situation changed when we increased the sample size. With 9 samples, 1012 DEGs were found without LFC filtering, and 897 with. Filtering removed 98 unique genes from edgeR, indicating that this tool might be more susceptible to small variations in expression levels. Moreover, the level of agreement increased, with all tools concurring on 164 genes, and DESeq2, edgeR, and limma agreeing on 359 genes after filtering. EdgeR discovered 278 unique genes, while all other tools identified one unique gene each.

Stupnikov et al., 2021[4] reported no concordance between tools but they used a logfold filtering of 2, more than 2 times the one we applied. We might have reproduced their results without filtering if we had obtained better results with limma at 3 samples or NOISeq at 9 samples. Those discrepancies highlight the need for a consensus on what is considered a small difference in gene expression.

It was expected that DESeq2 and edgeR would have the most similar results, since they share the same underlying model.

It is worth noting that in our study, the number of DEGs discovered by NOISeq decreased with an increasing number of samples. Nearly all genes identified by NOISeq were also identified by all other tools with 9 samples. These results suggest that the results produced with different tools are more comparable with a higher number of samples. The results are reproduced among tools and are more robust, with edgeR performing consistently well, regardless of the number of samples.

Overall Sensitivity and Specificity

Analysing all experiments, we were able to estimate the overall performance of the different DGE analysis pipelines. We took the average of specificity, sensitivity, false positive rate, and false negative rate across all the sample files (Table 3 and Figure 5). When we filtered the DEG for a fold Change of 0.58, edgeR performed better with a sensitivity of 59%, followed by DESeq2 (45%), Limma (30%) and NOISeq performed the worst at 19%. However, we observe the better sensitivity obtained with edgeR was obtained at the cost of more false positive results, with a specificity of 96.4%, compared to the other model. The Limma package provides the least false positive results (specificity: 99.8%) followed

by DESeq2(99.2%), and NOISeq (99%). Nonetheless, NOISeq and Limma have the highest false negative rate implying these packages failed to identify a greater number of Differentially expressed genes compared to DESeq2 and edgeR. However, we need to analyse if this difference is true for different numbers of samples. Other studies compared the rate of false discoveries between tools, and also found that out of those 4 tools, edgeR performed the worst in terms of false positive rate. However, they reported NOISeq had comparable results to NOISeq, with limma and DESeq performing the best[5].

Table 3: Mean Model Performances Across all Experiments

Parameters	edgeR	DESeq2	NOISeq	Limma
Sensitivity (True Positive Rate) (%)	57.7	44.5	18.6	29.7
Specificity (True Negative Rate) (%)	96.6	99.2	99.2	99.8

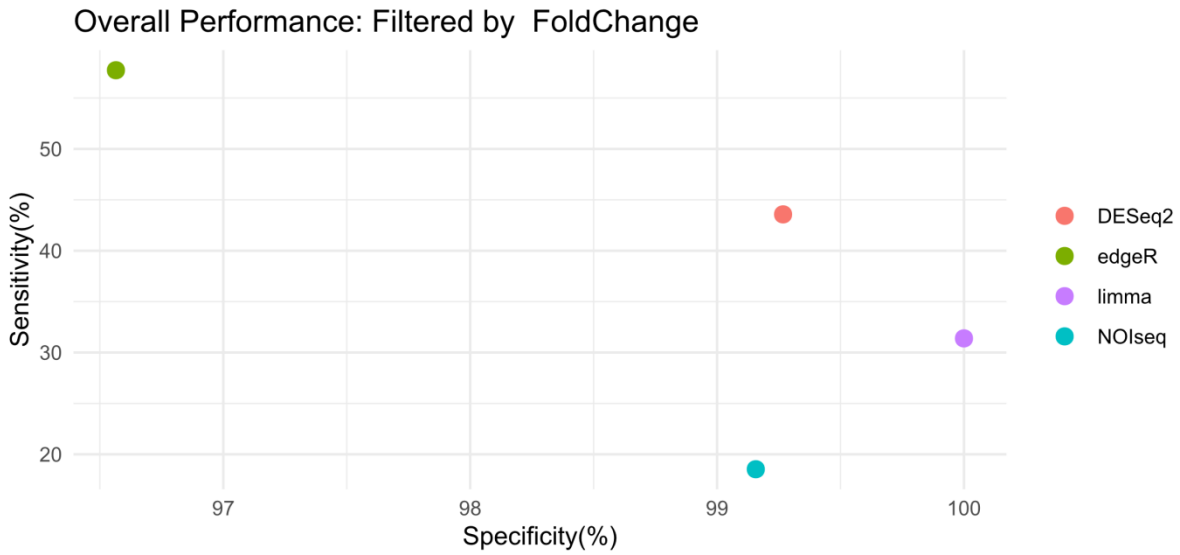


Figure 5: Mean Model Performances Across all Experiments. *Results across all 9 experiments were averaged for each tool. The datasets were filtered for low counts and a logFoldChange threshold of 0.58 was applied. Values of sensitivity and specificity for the different tools are as follow: Deseq2(0.45; 99.2), edgeR(0.58; 96.6), NOIseq(0.19; 99) and limma (0.3;1)*

Effect of Sample Size

This section will cover how the effectiveness of various methods varies with different sample sizes. With respect to **DESeq2**, sensitivity and specificity increased linearly as the number of samples increased (Figure 6). Sensitivity sharply increased from 28% to 57% (Table 4), while changes in specificity were negligible, going from 99% to 99.2%. Similar observations can be made for **limma**:

larger sample sizes had a negligible effect on specificity, changing from 100% to 99.7%, but they sharply increased sensitivity. Indeed, sensitivity increased from 3.8% to 36.8% as the sample size increased from 3 to 6, and further increased to 48.7% with 9 samples.

EdgeR displayed a similar improvement in sensitivity, increasing from 46.1% to 66.2%. However, we observed a more noticeable, albeit still small, increase of 1.5% in specificity from 95.7% to 97.3%.

NOISeq displayed a different trend from the other three tools. When the number of samples was increased from 3 to 6, specificity for NOISeq showed the most improvement among all the tools, rising from 97% to 100%. However, sensitivity decreased by 16% during the same period, going from 26% to 9.7%. Further increases in the sample size had no effect on specificity but seemed to restore sensitivity closer to its previous value, reaching 20.1%.

Increasing the sample size had a small effect on the ability of DESeq2 and limma to correctly identify genes that are not differentially expressed, but it had a more noticeable positive effect for NOISeq and EdgeR. Except for NOISeq, increasing the sample size increased the sensitivity of all tools. In conclusion, DESeq2, EdgeR, and Limma are more conservative with small sample sizes, resulting in fewer positive predictions and more false negative predictions (corresponding to 1-sensitivity). On the other hand, the NOISeq tool displayed an opposite trend, being more conservative with a larger sample size.

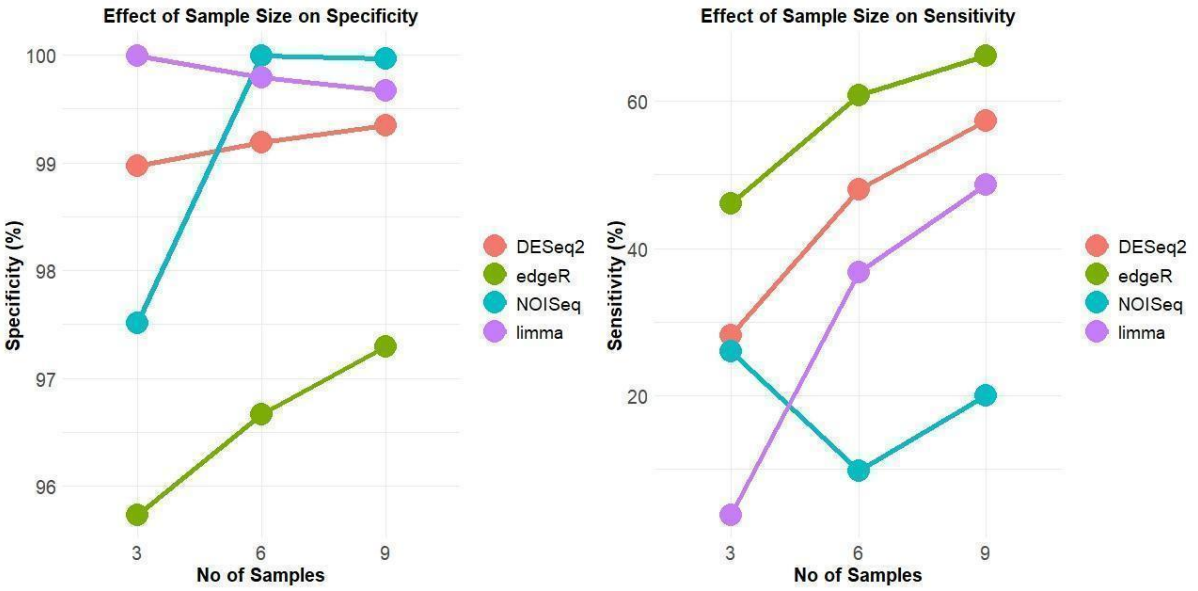


Figure 6: Effect of the number of samples on the sensitivity and specificity on different DEG analysis tools.*Sensitivity and specificity were compared for each tool. Average results for all experiments with a given sample size are shown. The dataset was filtered for low counts and LFC < 0.58.*

Table 4: Effect of the number of samples on the sensitivity and specificity of different DEG analysis tools

Number of Samples	Tool	Specificity	Sensitivity
3	DESeq2	0.990	0.282
	EdgeR	0.957	0.461
	NOISeq	0.971	0.260
	Limma	1.000	0.038
6	DESeq2	0.992	0.479
	EdgeR	0.967	0.608
	NOISeq	1.000	0.097
	Limma	0.998	0.368
9	DESeq2	0.993	0.574
	EdgeR	0.973	0.662
	NOISeq	1.000	0.201
	Limma	0.997	0.487

Effect of DE gene ratio

By comparing the average tool performance for a given ratio across various samples, we were able to determine how well the tools could distinguish between genes that were overexpressed or upregulated and those that were under-expressed or downregulated. We specifically looked at how the ratio affected both specificity and sensitivity. There were 1000 DE genes in each file, with varying ratios of upregulated to downregulated genes. Three of the files had a 1:1 ratio, three had a 3:1 ratio, and three did not contain any genes that were downregulated. From Figure 7, for all four methods, the increased number of elevated DE genes had no effect on specificity.

We observe no noticeable differences in sensitivity for various ratios for DSEq2, edgeR, and Limma. NOISeq seems to perform worse with an equal ratio, and increasing the ratio of upregulated genes vs downregulated genes from 1:1 to 3:1 increased the sensitivity from 14.3% to 21.5% (Table II, Appendix). However, further increasing the proportion of upregulated genes had no effects. It is interesting to observe that for all tools, the sensitivity was at the highest for a ratio 3:1, and might be worth investigating why.

Overall, all tools seem to have the same ability to detect up and down-regulated genes. We might speculate that, given the sensitivity was lower when half of the DE genes were downregulated, the NOISeq package is more cautious for upregulated genes. However, the fact that the sensitivity does not keep increasing when all genes are upregulated goes against the previous speculation.

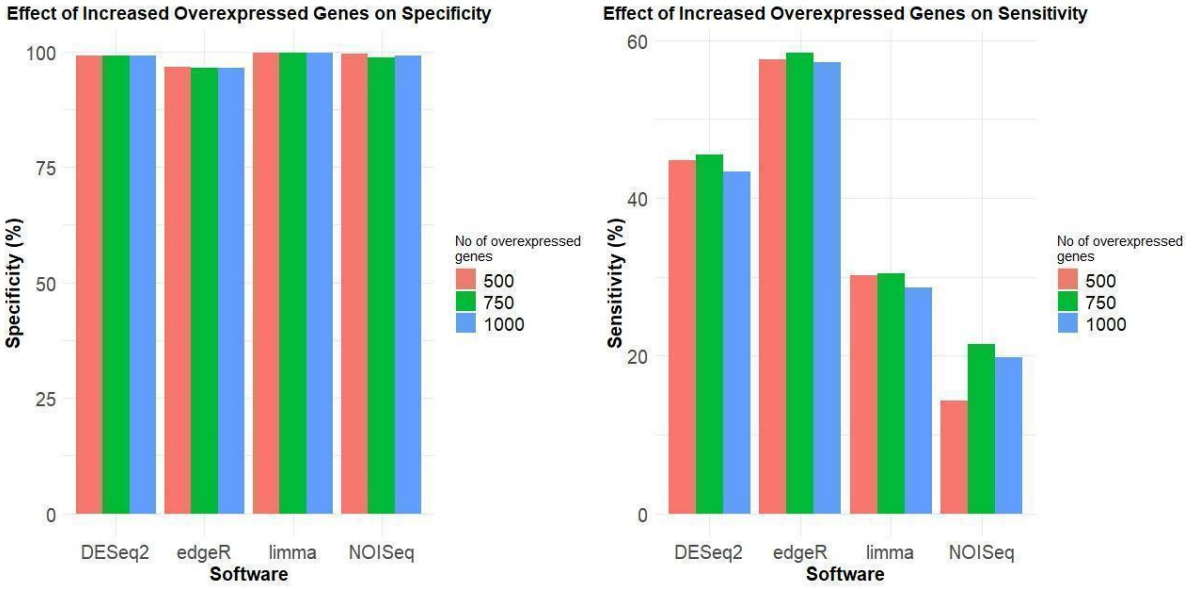


Figure 7: Effect of the ratio of upregulated differentially expressed genes. *Mean sensitivity(right panel) and specificity(left panel) for all experiments with a given ratio of up vs down regulated genes. Each tool evaluated 30 000 genes, 3000 were DE. Filters for low count and LFC<0.58 were applied.*

Runtime

The 10-fold average runtime for each experiment of each pipeline is shown in the table below(Table 5). From our initial analyses, we can observe differences between tools and between experiments with different sample sizes (Table 5 and Figure 8). NOISeq has the slowest runtime with 3 samples (29.57 s) but is in line with results for DESeq2 at 9 samples (19.72 sec vs 14.18). We observe that Limma is not impacted by the number of samples, with time always lower than one second. Here we experimented with 10,000 genes, and this analysis shed light on the necessity to consider the runtime of each tool, and not only the accuracy. Experiments might involve more than 10,000 or 9 samples, and it's crucial to consider the time needed to perform the analysis. Overall, all tools had comparable time efficiency, requiring less than a minute to complete an analysis. Those results are similar to those presented by Seyednasrollah et al. They also reported that limma had the best efficiency, with edgeR second, followed by DESeq and NOISeq. However, for every tool, they observed an increase of runtime, contrary to us. NOiSeq might have taken more time to perform the analysis at 3 samples, because when the number of samples is lower than 5, an additional step using k-mean clustering is added to identify genes with similar expression patterns. All the R/Bioconductor packages were run on a single core at a time.

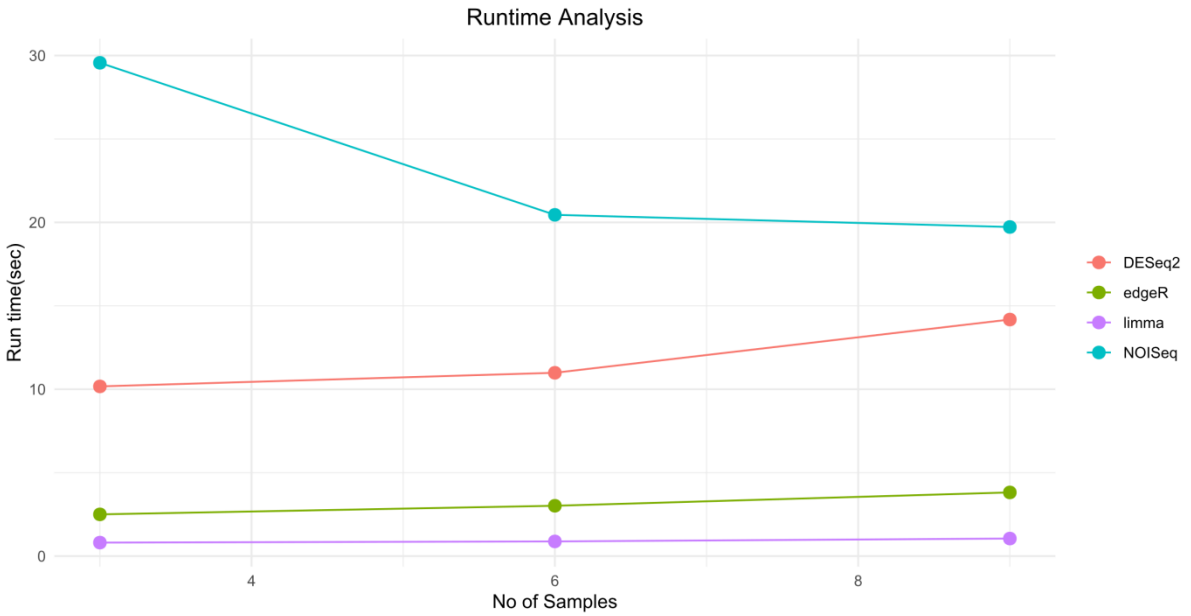


Figure 8: 10-fold Average Runtime in seconds. *All experiments were evaluated for differential gene expression, 10 times, by each tool, and the average is presented for each sample size. Filters for low count and LFC<0.58 were applied.*

Table 5: 10-fold Average Runtime in seconds

Number of Samples	Tool	10-fold Average Runtime(s)
3	DESeq2	10.17
	EdgeR	2.50
	NOISeq	29.57
	Limma	0.81
6	DESeq2	10.99
	EdgeR	3.01
	NOISeq	20.45
	Limma	0.88
9	DESeq2	14.18
	EdgeR	3.82
	NOISeq	19.72
	Limma	1.05

NOISeq identifies fewer genes as the number of samples per condition increases. Since NOISeq contrasts changes among conditions and changes within a condition to decrease the false positive rate, it may not be suitable for diverse datasets. NOISeq might be a suitable tool for identifying differentially expressed genes in data without distributional assumptions, provided the number of

samples per condition is above ten [5]. Additionally, it is suggested that while parametric methods tend to yield more significant results in highly replicated but variable data, NOISeqBIO strongly penalizes values with a high coefficient of variation. On the contrary, when data variability is lower, NOISeqBIO might be more effective in calling differentially expressed genes [7]. NOISeqBIO corrects the statistics for the biological variability specific to each gene.

In summary, edgeR performed the best, identifying more differentially expressed genes (DEGs) than all other tools. However, all the tools tended to identify the same genes as differentially expressed. The removal of lowly expressed genes and filtering for small differences in expression mainly impacted edgeR. These results suggest that edgeR is more sensitive to these factors, and considering these steps might be more important when using edgeR. EdgeR also displayed the best sensitivity in our study but at the cost of more false positive results. NOISeq performed the worst, but these results might be due to technical errors in our pipeline. For all tools except NOISeq, increasing the sample size increased the number of correctly identified DEGs, while the ratio of upregulated to downregulated genes had no effect. EdgeR also demonstrated one of the best time efficiencies, whereas NOISeq performed the worst.

Limitations

The primary limitation of this study is that the differential gene expression analysis is conducted using synthetic data. When we perform DGE analysis using real-world data, the tools' performance may differ from what we have found here. Additionally, only 10% of the genes with differential expression were present in the synthetic data that we used for the study. The fact that the study was limited to a straightforward two-group design is another drawback. In other words, condition 1 and condition 2 are the only two conditions that the study was intended to compare. When we apply complex design to compare more than two conditions, the tools perform differently. Certain tools, like limma and egdeR, are made to work with such intricate designs[1]. The quality and thoroughness of the pipelines' documentation varies significantly as well. While some, like DEseq, edgeR, and limma, have very detailed user manuals and guides with real-world examples, others only offer more condensed instructions and descriptions of parameter values (like NOISeq).

Conclusion

Different software packages available for DGE analysis to identify differentially expressed genes with read count data files containing different numbers of replicates were compared in the study. We compared the softwares DESeq2, edgeR, NOISeq and limma package and analysed their overall performance. We also analysed the sensitivity and specificity of the tools for varying sample sizes and different upregulated gene ratios along with runtime analysis and concordance. From the study, it

was observed that tools exhibit differences in performance measures in different contexts. It was identified that overall edgeR performed better than other tools. NOISeq software is sensitive toward lower sample sizes and the Limma package is sensitive to higher sample sizes. In NOIseq, as the number of replicates per condition increased above five, the number of differentially expressed genes reduced considerably. However, this might be due to data diversity.

Reference

[1] A. Gondane and H. M. Itkonen, “Revealing the History and Mystery of RNA-Seq,” *Curr. Issues Mol. Biol.*, vol. 45, no. 3, pp. 1860–1874, Feb. 2023, doi: 10.3390/cimb45030120.

[2] B. Langmead and S. L. Salzberg, “Fast gapped-read alignment with Bowtie 2,” *Nat. Methods*, vol. 9, no. 4, pp. 357–359, Apr. 2012, doi: 10.1038/nmeth.1923.

[3] S. Tarazona, P. Furió-Tarı, A. Ferrer, and A. Conesa, “NOISeq: Differential Expression in RNA-seq.” 2013. Accessed: Oct. 14, 2023. [Online]. Available: http://bioconductor.jp/packages/3.7/bioc/vignettes/NOISeq/inst/do[1]c/NOISeq.pdf

[4] A. Stupnikov *et al.*, “Robustness of differential gene expression analysis of RNA-seq,” *Comput. Struct. Biotechnol. J.*, vol. 19, pp. 3470–3481, 2021, doi: 10.1016/j.csbj.2021.05.040.

[5] F. Seyednasrollah, A. Laiho, and L. L. Elo, “Comparison of software packages for detecting differential expression in RNA-seq studies,” *Brief. Bioinform.*, vol. 16, no. 1, pp. 59–70, Jan. 2015, doi: 10.1093/bib/bbt086.

[6] A. Baccarella, C. R. Williams, J. Z. Parrish, and C. C. Kim, “Empirical assessment of the impact of sample number and read depth on RNA-Seq analysis workflow performance,” *BMC Bioinformatics*, vol. 19, no. 1, p. 423, Dec. 2018, doi: 10.1186/s12859-018-2445-2.

[7] S. Tarazona *et al.*, “Data quality aware analysis of differential expression in RNA-seq with NOISeq R/Bioc package,” *Nucleic Acids Res.*, p. gkv711, Jul. 2015, doi: 10.1093/nar/gkv711.

[8] M. E. Ritchie *et al.*, “limma powers differential expression analyses for RNA-sequencing and microarray studies,” *Nucleic Acids Res.*, vol. 43, no. 7, pp. e47–e47, Apr. 2015, doi: 10.1093/nar/gkv007.

[9] “RNA-seq analysis is easy as 1-2-3 with limma, Glimma and edgeR.” Accessed: Oct. 14, 2023. [Online]. Available: https://www.bioconductor.org/packages/devel/workflows/vignettes/RNAseq123/inst/doc/limmaWorkflow.html

[10] Ying Sha, J. H. Phan, and M. D. Wang, “Effect of low-expression gene filtering on detection of differentially expressed genes in RNA-seq data,” in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Milan: IEEE, Aug. 2015, pp. 6461–6464. doi: 10.1109/EMBC.2015.7319872.

Appendix

Supplementary Tables

Table I : Total DEG Count for Each Tools, Across All Experiments, With and Without Low Counts and LogFoldChange Filtering

Tool	N	No Filter	Low Count Filtering	LogFold Filtering	Low Count and LogFold Filtering
	3	2844	2534	2844	2534
edgeR	6	3359	2906	3185	2723
	9	3631	3144	3205	2717
	3	1123	1084	1123	1084
DESeq2	6	1687	1638	1657	1612
	9	1972	1890	1899	1817
	3	66	117	1326	117
limma	6	1137	1158	1136	1156
	9	1580	1549	1562	1529
	3	1396	1542	1326	1450
NOISeq	6	276	292	276	292
	9	562	615	556	608

Table II : Mean Sensitivity by Ratio and Tools

Tool	Upreguated genes	Sensitivity
	500	0.575
edgeR	750	0.585
	1000	0.572
	500	0.437
DESeq2	750	0.444
	1000	0.425
	500	0.321
limma	750	0.318
	1000	0.303
	500	0.143
NOISeq	750	0.215
	1000	0.198

Table III: Combine DEG gene identified, by number of samples.

Tools Combinations	Combined Number of DEG identified (out of 1000)		
	9	6	3
DESeq2	584	472	284
edgeR	658	611	456
Limma	502	357	46
NOISeq	166	41	223
DESeq2+edgeR	685	623	456
DESeq2+limma	589	474	284

DESeq2+NOISeq	585	474	318
Limma+NOISeq	504	358	223
Limma+edgeR	661	611	456
NOISeq+edgeR	659	611	466
DESeq2+edgeR+limma	688	623	456
DESeq2+edgeR+NOISeq	686	623	466
Limma+edgeR+NOISeq	688	623	456
All	688	623	466

Code

DGE analysis using limma

```
library(edgeR)
library(limma)
library(RColorBrewer)
library(dplyr)
library(caret)
library(tidyr)
## Data Import ####
sample3_dge <- function (filename){
  #Importing the count matrix
  count <- read.delim(paste0("data/",filename,".tsv"))

  #Creating Dgelist object
  dg <- DGEList(count)

  #Creating groups
  if (substring(filename, 1, 1) == "3"){
    group <- as.factor(c("Condition1", "Condition1", "Condition1",
                         "Condition2", "Condition2", "Condition2"))
  }
  else if (substring(filename, 1, 1) == "6"){
    group <- as.factor(c("Condition1", "Condition1", "Condition1",
                         "Condition1", "Condition1", "Condition1",
                         "Condition2", "Condition2", "Condition2",
                         "Condition2", "Condition2", "Condition2"))
  }
  else if (substring(filename, 1, 1) == "9"){
    group <- as.factor(c("Condition1", "Condition1", "Condition1",
                         "Condition1", "Condition1", "Condition1",
                         "Condition1", "Condition1", "Condition1",
                         "Condition2", "Condition2", "Condition2",
                         "Condition2", "Condition2", "Condition2",
                         "Condition2", "Condition2", "Condition2"))
  }

  dg$samples$group <- group


  ## Data preprocessing ####

  # Removing genes that are lowly expressed
  keep.exprs <- filterByExpr(dg, group=group)
```

```
  dg <- dg[keep.exprs,, keep.lib.sizes=FALSE]
  dim(dg)

  #normalisation
  dg <- calcNormFactors(dg, method = "TMM")
  dg$samples$norm.factors

  #plotting - unsupervised clustering of samples
  lcpm <- cpm(dg, log=TRUE)

  plotMDS(lcpm,  col=as.numeric(group))
  title(main="Condition group")

  #sample 1, 2 and 3 has condition 1 and sample 4,5 and 6 has condition 2.
A clear disticntion between samples is visible in the plot

  ## Differential expression analysis ##

  #Design matrix
  design <- model.matrix(~0+group)
  colnames(design) <- gsub("group", "", colnames(design))
  design


  contr.matrix <- makeContrasts(
    Cnd2vsCnd1 = Condition2-Condition1,
    levels = colnames(design))
  contr.matrix

  vdg <- voom(dg, design, plot=TRUE)
  vdg

  #Fitting linear model
  vfit <- lmFit(vdg, design)
  vfit <- contrasts.fit(vfit, contrasts=contr.matrix)
  efit <- eBayes(vfit)

  #Checking number of up and down regulated genes
  print(summary(decideTests(efit)))


  #Table with pvalue and mapping differentially expressed genes
  top.table <- topTable(efit, sort.by = "P", n = Inf) %>%
    mutate(upregulation=ifelse(adj.P.Val <0.05  & logFC >0.58, 1, 0)) %>%
    mutate(downregulation=ifelse(adj.P.Val <0.05  & logFC<-0.58, 1, 0)) %>%
    mutate(differential.expression=ifelse(upregulation ==1|
                                          downregulation ==1, 1, 0))


  #reading meta data
  count <- read.delim(paste0("data/",filename,"_meta.tsv"))
  result <- left_join(count,top.table,by="X")

  #top.table$index <- as.numeric(row.names(top.table))
  result <- result %>%

select(c("X","upregulation.y","downregulation.y","differential.expression.y
")) %>%
    rename_with( ~gsub(".y", "", .x, fixed = TRUE))

  result[is.na(result)] <- 0
```

```
  #writing into file
  write.table(result, file =
paste0("Results/limma/",filename,"_result.txt"), row.names = F, sep = "\t",
quote = F)
}


result_3_500_500 <- sample3_dge("3_500_500")
result_3_750_250 <- sample3_dge("3_750_250")
result_3_1000_0 <- sample3_dge("3_1000_0")
result_6_500_500 <- sample3_dge("6_500_500")
result_6_750_250 <- sample3_dge("6_750_250")
result_6_1000_0 <- sample3_dge("6_1000_0")
result_9_500_500 <- sample3_dge("9_500_500")
result_9_750_250 <- sample3_dge("9_750_250")
result_9_1000_0 <- sample3_dge("9_1000_0")
```

DGE analysis using edgeR

```
library(tidyverse)

library(RColorBrewer)
library(edgeR)

library(pheatmap)
library(DEGreport)
library(ashr)
library(ggplot2)
library(ggrepel)
library(DEGreport)
library(dplyr)
create_result_file <-function(resTable, resultFile, lfcCutoff=0,outputname)
{
  #takes the table produced after DGE, the true result file to which it
will be compared, and the lfc cutoff to be used
  p_cutoff <- 0.05
  lfc_cutoff <- lfcCutoff
  trueResult <-read.table(paste("Meta/",resultFile,sep = ""), header=T,
row.names=1)
  #filter by p value and lfc
  resTable <- as.data.frame(resTable)
  res_table_filtered <- resTable %>%
    filter(PValue < p_cutoff & abs(logFC) > lfc_cutoff)
  #create final output file
  result_log_filtered <- data.frame(
    gene = rownames(trueResult),
    upregulation = rep(0, nrow(trueResult)),
    downregulation = rep(0, nrow(trueResult)),
    differential.expression = rep(0, nrow(trueResult))
  )
  #check results in res_tableCE_TB and update result_meta_df
  for (i in 1:nrow(res_table_filtered))
  {
    if (!is.na(res_table_filtered$PValue[i]))
    {
      #The following line are to make sure the loop works even if the genes
are not ordered
      numeric_value <- as.numeric(sub("^g([0-9]+).*",
"\\1",rownames(res_table_filtered[i,])))
      result_log_filtered$differential.expression[numeric_value] <- 1
      if(res_table_filtered$logFC[i]>0)
      {
```

```
        result_log_filtered$upregulation[numeric_value] <- 1
      }
      else
      {
        result_log_filtered$downregulation[numeric_value] <- 1
      }
    }
  }
  string_output <- substr(resultFile, 1, nchar(resultFile) - 4)
  #save the result_file
  output_file <-  paste0("Results/edger/edgeR_",
string_output,'_',outputname, "_Results.txt")
  write.table(  result_log_filtered, file = output_file, sep = "\t", quote
= FALSE, row.names = FALSE)
}

EdgeREvaluator <-function(dataFile, metaFile, resultFile)
{
  data <-read.table(paste("Data/",dataFile,sep = ""), header=T,
row.names=1)
  meta <- read.table(paste("MetaFull/",metaFile,sep = ""), header=T,
row.names=1)

  #create the DGEList object, takes at leat a count matrix
  samples <- substr(dataFile, 1, nchar(dataFile) - 4)
  split_values <- strsplit(dataFile, "_")[[1]]
  #extract the forst number
  num_samples <- as.integer(split_values[1])
  num_conditions <- 2
  groups <-c(rep(1, num_samples), rep(2, num_samples))
  y<- DGEList(counts = data, group=groups)
  #filter genes with low counts, remove rows a small count(the smallest
group size)
  keep <- filterByExpr(y)
  y_filterd_low_count <- y
  y_filtered_low_count <- y[keep, , keep.lib.sizes=FALSE]
  #normalization by the size of the library, TMM
  y <- normLibSizes(y)
  y_filtered_low_count<-normLibSizes(y_filtered_low_count)
  #estimate dispersion
  y <- estimateDisp(y)
  y_filtered_low_count<-estimateDisp(y_filtered_low_count)
  #Testing for DE genes, if a single factor exact test is possible
  et <- exactTest(y)
  et_filtered_low_count<-exactTest(y_filtered_low_count)

  create_result_file(et$table,resultFile, 0.58,'log_filtered')
  create_result_file(et_filtered_low_count$table,resultFile, 0.58,
'low_log_filtered')
  create_result_file(et$table,resultFile, 0,'no_filtered')
  create_result_file(et_filtered_low_count$table,resultFile,
0,'low_filtered')
}
```

DGE analysis using DESeq2

```
library(tidyverse)
```

```
library(RColorBrewer)
library(DESeq2)

library(pheatmap)
library(DEGreport)
library(ashr)
library(ggplot2)
library(ggrepel)
library(DEGreport)
create_result_file <-function(resTable, resultFile, lfcCutoff=0,outputname)
{

  padj.cutoff <- 0.05
  lfc.cutoff <- lfcCutoff
  trueResult <-read.table(paste("Meta/",resultFile,sep = ""), header=T,
row.names=1)

  resTable <- as.data.frame(resTable)
  res_table_filtered <- resTable %>%
    filter(padj < padj.cutoff & abs(log2FoldChange) > lfc.cutoff)

  result_log_filtered <- data.frame(
    gene = rownames(trueResult),
    upregulation = rep(0, nrow(trueResult)),
    downregulation = rep(0, nrow(trueResult)),
    differential.expression = rep(0, nrow(trueResult))
  )
  #check results in res_tableCE_TB and update result_meta_df
  for (i in 1:nrow(res_table_filtered))
  {

    if (!is.na(res_table_filtered$padj[i]))
    {
      numeric_value <- as.numeric(sub("^g([0-9]+).*",
"\\1",rownames(res_table_filtered[i,])))
      result_log_filtered$differential.expression[numeric_value] <- 1
      if(res_table_filtered$log2FoldChange[i]>0)
      {
        result_log_filtered$upregulation[numeric_value] <- 1
      }
      else
      {
        result_log_filtered$downregulation[numeric_value] <- 1
      }
    }

  }
  string_output <- substr(resultFile, 1, nchar(resultFile) - 4)
  #save the result_file
  output_file <-  paste0("Results/DESeq2/DESeq2_",
string_output,'_',outputname, "_Results.txt")
  write.table(  result_log_filtered, file = output_file, sep = "\t", quote
= FALSE, row.names = FALSE)
}

DESeqEvaluator <-function(dataFile, metaFile,resultFile)
  {
  min_count <- 10

  data <- read.table(paste("Data/",dataFile,sep = ""), header=T,
row.names=1)
```

```
  meta <- read.table(paste("MetaFull/",metaFile,sep = ""), header=T,
row.names=1)
  max_group <- ncol(data)/2

  ## Check that sample names match in both files
  all(colnames(data) %in% rownames(meta))
  all(colnames(data) == rownames(meta))
  ## Create DESeq2Dataset object
  dds <- DESeqDataSetFromMatrix(countData = data, colData = meta, design =
~Condition)
  # View the original counts matrix

  # when have to perform the median of ratio of normalization
  dds <- estimateSizeFactors(dds)
  #filter low gene with total count less than specified accross all genes
  keep= rowSums( counts(dds) >= min_count ) >= max_group
  filtered_dds <-dds[keep,]
  dds <- DESeq(dds)

  dds_filtered <-DESeq(filtered_dds)
  #### Define contrasts, extract results table, and shrink the log2 fold
changes
  contrast_ce <- c("Condition", "cond2", "cond1")
  res_tableCE_unshrunken <- results(dds, contrast=contrast_ce, alpha =
0.05)
  res_tableCE_unshrunken_filtered <-results(dds_filtered,
contrast=contrast_ce, alpha = 0.05)

  res_tableCE <- lfcShrink(dds, contrast=contrast_ce,
res=res_tableCE_unshrunken,type="normal")
  res_tableCE_filtered <- lfcShrink(dds_filtered, contrast=contrast_ce,
res=res_tableCE_unshrunken_filtered,type="normal")

  create_result_file(res_tableCE_filtered,resultFile, 0, 'low_filtered')
  create_result_file(res_tableCE_filtered,resultFile, 0.58,
'low_log_filtered')
  create_result_file(res_tableCE,resultFile, 0.58, 'log_filtered')
  create_result_file(res_tableCE,resultFile, 0, 'no_filtered')
  print("Over")

}
```

DGE analysis using NOISeq

The same approach is followed for all files except for the 'nclust' parameter in 'noiseqbio' function for files with 3 samples per condition. Other files don't have this parameter used in the 'noiseqbio' function.

```
padj.cutoff <- 0.05
lfc.cutoff <- 0.58
padj.cutoff <- 0.05
lfc.cutoff <- 0.58
## Load count matrix data
count_data_3_500_500 <- read.table("data/3_500_500.tsv",header=T,
row.names=1)
head(count_data_3_500_500)# to view the first 6 rows of data
# Load meta data file for count data
meta_data_3_500_500 <- read.table("meta/3_500_500_m.txt")
head(meta_data_3_500_500)# to view the first 6 rows of data
# Check that sample names match in count and meta files
all(colnames(count_data_3_500_500) %in% rownames(meta_data_3_500_500))
```

```
all(colnames(count_data_3_500_500) == rownames(meta_data_3_500_500))
#Convert generic data frame into a NOISeq object
#readData function returns an object of Biobase's eSet class

NoisData_3_500_500<-readData(data=count_data_3_500_500,
factors=meta_data_3_500_500)
NoisData_3_500_500
# Normalization
myRPKM_3_500_500 = rpkm(assayData(NoisData_3_500_500)$exprs, k = 0, lc = 1)
# Low-count filtering
filtered_counts_3_500_500 = filtered.data(count_data_3_500_500, factor =
meta_data_3_500_500$condition, norm = TRUE,  depth = NULL, method = 1,
cv.cutoff = 100, cpm = 1, p.adj = "fdr")
## Differential Expression :NOISeq with biological replicates
noiseqbio_3_500_500 = noiseqbio(NoisData_3_500_500, k = 0.5, norm = "rpkm",
factor = "condition",lc = 1, r = 20, adj = 1.5, nclust = 15,plot = FALSE,
a0per = 0.9, random.seed = 12345)

res_tb_3_500_500<-noiseqbio_3_500_500@results %>%
          data.frame() %>%
          rownames_to_column(var="gene") %>%
          as_tibble()
res_tb_3_500_500

#q : threshold for q ,default 0.95 for Noiseqbio)
noiseqbio.deg_3_500_500 = degenes(noiseqbio_3_500_500, q = 0.95, M = NULL)
noiseqbio.deg1_3_500_500 = degenes(noiseqbio_3_500_500, q = 0.95, M = "up")
noiseqbio.deg2_3_500_500 = degenes(noiseqbio_3_500_500, q = 0.95, M =
"down")

noiseqbio.deg_3_500_500 <-noiseqbio.deg_3_500_500%>%
           data.frame() %>%
           rownames_to_column(var="gene") %>%
           as_tibble()
 #View(noiseqbio.deg_3_500_500)

noiseqbio.deg1_3_500_500 <-noiseqbio.deg1_3_500_500%>%
           data.frame() %>%
           rownames_to_column(var="gene") %>%
           as_tibble()
 #View(noiseqbio.deg1_3_500_500)

 noiseqbio.deg2_3_500_500 <-noiseqbio.deg2_3_500_500%>%
           data.frame() %>%
           rownames_to_column(var="gene") %>%
           as_tibble()
 View(noiseqbio.deg2_3_500_500)

res_tb_3_500_500 <- res_tb_3_500_500 %>%
      mutate(upregulation= ifelse((gene %in% noiseqbio.deg1_3_500_500$gene
&(((1-prob)<padj.cutoff) & abs(log2FC) >= lfc.cutoff)), 1, 0),
            downregulation = ifelse((gene %in%
noiseqbio.deg2_3_500_500$gene &(((1-prob)<padj.cutoff) & abs(log2FC) >=
lfc.cutoff)), 1, 0),
            differential.expression = ifelse((upregulation == 1 |
                                                 downregulation ==1) ,
1, 0))
#View(res_tb_3_500_500)
resultfile_3_500_500<- subset(res_tb_3_500_500, select = -
c(X1_mean,X2_mean,theta,prob, log2FC) )
```

```
### write the results to a file
fwrite(resultfile_3_500_500, "results/3_500_500_Results.txt", sep="\t")
```

Low count filtering+LFC study

```
library(ggplot2)
library(ggrepel)
library(gridExtra)
library(dplyr)
library(ggVennDiagram)
library(dplyr)
library(ggpubr)
#########################################################################
#folders
#########################################################################
##figure 1 , will plot filtered vs unfiltered for low cout, without log fold
#########################################################################

extract_number_deg<-function(result_file)
{
  #taxe a result file and count the number of 1 in DEG, does not compare with gound
truth

  data<-read.table(paste("Results/",result_file,sep = ""), header=T, row.names=1,)
  number_DEG <- sum(data$differential.expression)
  return (number_DEG)
}

tools <- c("edgeR", "DESeq2", "NOIseq","limmaAG")
ratios <- c("500_500", "750_250", "1000_0")
methods <-c("low_filtered","log_filtered","low_log_filtered","no_filtered" )
sample_sizes<-c(3,6,9)

# Create an empty list to store tables for each sample size
result_df <- data.frame(tools=c("edgeR", "DESeq2", "NOIseq","limmaAG","edgeR",
"DESeq2", "NOIseq","limmaAG","edgeR", "DESeq2", "NOIseq","limmaAG"))
result_df$sample <- c(3,3,3,3,6,6,6,6,9,9,9,9)
result_df$low_filtered <-c(0,0,0,0,0,0,0,0,0,0,0,0)
result_df$log_filtered <-c(0,0,0,0,0,0,0,0,0,0,0,0)
result_df$low_log_filtered <-c(0,0,0,0,0,0,0,0,0,0,0,0)
result_df$no_filtered <-c(0,0,0,0,0,0,0,0,0,0,0,0)

# Loop through each sample size
for (sample_size in sample_sizes)
  {
  # Initialize an empty data frame to store results for this sample size
  # Loop through each tool
  for (tool in tools)
    {
    # Loop through each method
    for (method in methods)
      {

      number_DEG <- 0

      for (ratio in ratios)
        {
        # Create the formatted string
        formatted_string <- paste(tool,paste(tool,sample_size, ratio,"meta",
method, "Results.txt", sep = "_"), sep="/")
        # Call the extract_number_deg function and store the result
        number_DEG = number_DEG +extract_number_deg(formatted_string)
        }
```

```
      result_df[result_df$tools == tool & result_df$sample == sample_size, method]
<- number_DEG
    }
  }
}
print(result_df)
#################################################################
#filtered low count vs unfiltered low count

custom_colors <- c("edgeR" = "#7CAE00", "DESeq2" = "#F8766D", "NOIseq" =
"#00BFC4","limmaAG" = "#C77CFF" )
filtered_low_plot = ggplot(result_df, aes(x = sample,group=tools, color=tools)) +
  geom_line(aes(y = low_filtered)) +
  geom_line(aes(y = no_filtered),linetype="twodash")+
  scale_x_continuous(breaks = c(3, 6, 9))+# Use geom_line to create a line plot
  labs(
    title = "Effect of Low Count Filtering on the Number of Detected DEG",
    x = "Number of Samples",
    y = "Number of Detected Genes")+
  theme_classic() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5),
        panel.grid.minor = element_blank(),
        legend.title.align = 0.5)+
  scale_color_manual(values = custom_colors)+
  guides(color = guide_legend(title = NULL))

print(filtered_low_plot)
ggsave("figures/countFiltering.svg", plot = filtered_low_plot, device = "svg", dpi
= 300)
##############################
#filtered logfold count vs unfiltered log
filtered_log_plot = ggplot(result_df, aes(x = sample,group=tools, color=tools)) +
  geom_line(aes(y = log_filtered)) +
  geom_line(aes(y = no_filtered),linetype="twodash")+
  scale_x_continuous(breaks = c(3, 6, 9)) +# Use geom_line to create a line plot
  labs(
    title = "Effect of logFoldchange Filtering on the Number of Detected DEG.
lfc=0.58",
    x = "Number of Samples",
    y = "Number of Detected Genes")+
  theme_classic() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5),
        panel.grid.minor = element_blank(),
        legend.title.align = 0.5)+
  scale_color_manual(values = custom_colors)+
  guides(color = guide_legend(title = NULL))

print(filtered_log_plot)
ggsave("figures/FOldFiltering.svg", plot = filtered_log_plot, device = "svg", dpi =
300)
```

Venn Diagrams

```
#create summarary input 4 df with the result, one for each tool, and output 1 df
create_summary_result_nsamples <- function(edge, deseq2, limma, Noiseq)
{
  number_row <-10000
  tool_columns <- c("edgeR", "DESeq2", "limma", "NOIseq")
  gene_tool_matrix <- data.frame(matrix(0, nrow = number_row, ncol = 4))
  colnames(gene_tool_matrix) <- tool_columns
  # Set the row names to "g1" to "g10000"
  row.names(gene_tool_matrix) <- paste("g", 1:number_row, sep = "")

  for (i in 1:number_row) {
    gene_name <- paste("g", i, sep = "")
```

```
    gene_tool_matrix[gene_name, "edgeR"] <- ifelse( edge[gene_name,
"differential.expression"] == 1, 1, 0)
    gene_tool_matrix[gene_name, "DESeq2"] <- ifelse(deseq2[gene_name,
"differential.expression"] == 1, 1, 0)
    gene_tool_matrix[gene_name, "limma"] <- ifelse( limma[gene_name,
"differential.expression"] == 1, 1, 0)
    gene_tool_matrix[gene_name, "NOIseq"] <- ifelse( Noiseq[gene_name,
"differential.expression"] == 1, 1, 0)
  }
  return(gene_tool_matrix)

}
##################################################
#create a df for N3,6,9, filtered by log and low

n_9_result_df_filtered =
create_summary_result_nsamples(result_edgeR_9_500_500_filtered,
result_deseq2_9_500_500_filtered,result_limma_9_500_500_filtered,result_noiseq_9_50
0_500_filtered)
n_3_result_df_filtered =
create_summary_result_nsamples(result_edgeR_3_500_500_filtered,
result_deseq2_3_500_500_filtered,result_limma_3_500_500_filtered,result_noiseq_3_50
0_500_filtered)
n_9_result_df = create_summary_result_nsamples(result_edgeR_9_500_500,
result_deseq2_9_500_500,result_limma_9_500_500,result_noiseq_9_500_500)
n_3_result_df = create_summary_result_nsamples(result_edgeR_3_500_500,
result_deseq2_3_500_500,result_limma_3_500_500,result_noiseq_3_500_500)
##################################################

##venn Diagram
create_venn<- function(result4tools)
{
  edgeR_genes <- rownames(result4tools[result4tools$edgeR == 1, ],)
  Deseq2_genes <- rownames(result4tools[result4tools$DESeq2 == 1, ])
  limma_genes <- rownames(result4tools[result4tools$limma == 1, ])
  rows_with_na <- grepl("NA", limma_genes)
  # Use the logical vector to subset and remove the row names with "NA"
  limma_genes <- limma_genes[!rows_with_na]

  Noiseq_genes <- rownames(result4tools[result4tools$NOIseq == 1, ])
  rows_with_na <- grepl("NA", Noiseq_genes)
  # Use the logical vector to subset and remove the row names with "NA"
  Noiseq_genes <- Noiseq_genes[!rows_with_na]
  X<-list(edgeR_genes,Deseq2_genes,limma_genes,Noiseq_genes)

  ggVennDiagram(X,label = "count",label_size = 2,set_size = 3,label_alpha = 0,color
= 1, lwd = 0.7,category.names = c("edgeR","DESeq2","limma", "NOISeq"))+
    theme(legend.position = "none")+
    scale_fill_gradient(low = "#F4FAFE", high = "#4981BF")+
    theme(plot.margin = unit(c(2,2,2,2), "lines") )+
    scale_x_continuous(expand = expansion(mult = .2))

}
venn_n_3 = create_venn(n_3_result_df)
venn_n_3_filtered = create_venn(n_3_result_df_filtered)
venn_n_9 = create_venn(n_9_result_df)
venn_n_9_filtered = create_venn(n_9_result_df_filtered)

combined_plots <- ggarrange(venn_n_3, venn_n_9, venn_n_3_filtered
,venn_n_9_filtered + rremove("x.text"),
                            labels = c("3 Samples, No LFC Filtering", "9 Samples,
No LFC Change Filtering", "3 Samples, Filtered Logfold 0.58","9 Samples, Filtered
Logfold 0.58"),
                            ncol = 2, nrow = 2,vjust=2.5, hjust=-0.1)
combined_plots
ggsave("figures/vennFInalD.svg", plot = combined_plots, device = "svg", dpi = 300)
```

Evaluation of the Models

The following function was used to compare results from different DGE analysis tools with the real classifications. It outputs a vector with the relevant statistics.

```
evaluate_DE <- function(calculated_data, results, method)
{
  ##takes the file with the result, the file with the calculated results, the
method used
  result_file <- file.path("Results", method, calculated_data)
  answer_file <- file.path("Meta", results)
  results <- read.table(result_file, header=T, row.names=1)
  answers <- read.table(answer_file, header=T, row.names=1)

  truePositive <- 0
  trueNegative <- 0
  falsePositive <-0
  falseNegative <- 0

  for(i in 1:nrow(answers))
  {
    row1 <- results[i,]
    row2 <- answers[i,]
    {
      if(i< nrow(results)){
      if (row1$differential.expression == 1 && row2$differential.expression == 1) {
        truePositive <- truePositive + 1
      } else if (row1$differential.expression == 0 && row2$differential.expression
== 0) {
        trueNegative <- trueNegative + 1
      } else if (row1$differential.expression == 1 && row2$differential.expression
== 0) {
        falsePositive <- falsePositive + 1
      } else if (row1$differential.expression == 0 && row2$differential.expression
== 1) {
        falseNegative <- falseNegative + 1
      }
      }
    }
  }
  recall <- truePositive/(truePositive+falseNegative)
  accuracy <-
(truePositive+trueNegative)/(trueNegative+truePositive+falseNegative+falsePositive)
  precision <- truePositive/(truePositive+falsePositive)
  falsePositiveRate <- falsePositive/(falsePositive+trueNegative)
  trueNegativeRate <- trueNegative/(trueNegative+falsePositive)
  sensitivity <- truePositive/(truePositive+falseNegative)
  specificity<-trueNegative/(trueNegative+falsePositive)



  result_vector <- c("True Positives:" = truePositive,
                     "True Negatives:"= trueNegative,
                     "False Positives:" = falsePositive,
                     "False Negatives:" = falseNegative,
                     "Recall:" = recall,
                     "Accuracy:" =accuracy,
                     "False Positive Rate:" = falsePositiveRate,
                     "True Negative Rate:"= trueNegativeRate,
                    "Sensitivity:"=sensitivity,
                    "specificity:"=specificity)
}
```

Next step was to create a df with the results for all experiments and tools.

```
#DESeq_low_log_filtered
v3_500_500DESeq =evaluate_DE("DESeq2_3_500_500_meta_low_log_filtered_Results.txt",
"3_500_500_meta.tsv","DESeq2")
v3_750_250DESeq=evaluate_DE("DESeq2_3_750_250_meta_low_log_filtered_Results.txt",
"3_750_250_meta.tsv","DESeq2")
```

```
v3_1000_0DESeq=evaluate_DE("DESeq2_3_1000_0_meta_low_log_filtered_Results.txt",
"3_1000_0_meta.tsv","DESeq2")
v6_500_500DESeq=evaluate_DE("DESeq2_6_500_500_meta_low_log_filtered_Results.txt",
"6_500_500_meta.tsv","DESeq2")
v6_750_250DESeq=evaluate_DE("DESeq2_6_750_250_meta_low_log_filtered_Results.txt",
"6_750_250_meta.tsv","DESeq2")
v6_1000_0DESeq=evaluate_DE("DESeq2_6_1000_0_meta_low_log_filtered_Results.txt",
"6_1000_0_meta.tsv","DESeq2")
v9_500_500DESeq=evaluate_DE("DESeq2_9_500_500_meta_low_log_filtered_Results.txt",
"9_500_500_meta.tsv","DESeq2")
v9_750_250DESeq=evaluate_DE("DESeq2_9_750_250_meta_low_log_filtered_Results.txt",
"9_750_250_meta.tsv","DESeq2")
v9_1000_0DESeq=evaluate_DE("DESeq2_9_1000_0_meta_low_log_filtered_Results.txt",
"9_1000_0_meta.tsv","DESeq2")
#NOISeq_low_filtered
v3_500_500noiseq =
evaluate_DE("NOISeq_3_500_500_meta_low_log_filtered_Results.txt",
"3_500_500_meta.tsv", "NOISeq")
v3_750_250noiseq =
evaluate_DE("NOISeq_3_750_250_meta_low_log_filtered_Results.txt",
"3_750_250_meta.tsv", "NOISeq")
v3_1000_0noiseq = evaluate_DE("NOISeq_3_1000_0_meta_low_log_filtered_Results.txt",
"3_1000_0_meta.tsv", "NOISeq")
v6_500_500noiseq =
evaluate_DE("NOISeq_6_500_500_meta_low_log_filtered_Results.txt",
"6_500_500_meta.tsv", "NOISeq")
v6_750_250noiseq =
evaluate_DE("NOISeq_6_750_250_meta_low_log_filtered_Results.txt",
"6_750_250_meta.tsv", "NOISeq")
v6_1000_0noiseq = evaluate_DE("NOISeq_6_1000_0_meta_low_log_filtered_Results.txt",
"6_1000_0_meta.tsv", "NOISeq")
v9_500_500noiseq =
evaluate_DE("NOISeq_9_500_500_meta_low_log_filtered_Results.txt",
"9_500_500_meta.tsv", "NOISeq")
v9_750_250noiseq =
evaluate_DE("NOISeq_9_750_250_meta_low_log_filtered_Results.txt",
"9_750_250_meta.tsv", "NOISeq")
v9_1000_0noiseq = evaluate_DE("NOISeq_9_1000_0_meta_low_log_filtered_Results.txt",
"9_1000_0_meta.tsv", "NOISeq")
#############################################################################
#
#limma log low filtered
v3_500_500limma =
evaluate_DE("limmaAG_3_500_500_meta_low_log_filtered_Results.txt",
"3_500_500_meta.tsv", "limmaAG")
v3_750_250limma =
evaluate_DE("limmaAG_3_750_250_meta_low_log_filtered_Results.txt",
"3_750_250_meta.tsv", "limmaAG")
v3_1000_0limma = evaluate_DE("limmaAG_3_1000_0_meta_low_filtered_Results.txt",
"3_1000_0_meta.tsv", "limmaAG")
v6_500_500limma = evaluate_DE("limmaAG_6_500_500_meta_low_filtered_Results.txt",
"6_500_500_meta.tsv", "limmaAG")
v6_750_250limma = evaluate_DE("limmaAG_6_750_250_meta_low_filtered_Results.txt",
"6_750_250_meta.tsv", "limmaAG")
v6_1000_0limma = evaluate_DE("limmaAG_6_1000_0_meta_low_filtered_Results.txt",
"6_1000_0_meta.tsv", "limmaAG")
v9_500_500limma = evaluate_DE("limmaAG_9_500_500_meta_low_filtered_Results.txt",
"9_500_500_meta.tsv", "limmaAG")
v9_750_250limma = evaluate_DE("limmaAG_9_750_250_meta_low_filtered_Results.txt",
"9_750_250_meta.tsv", "limmaAG")
v9_1000_0limma = evaluate_DE("limmaAG_9_1000_0_meta_low_filtered_Results.txt",
"9_1000_0_meta.tsv", "limmaAG")
#edgeR low filtered
v3_500_500edgeR = evaluate_DE("edgeR_3_500_500_meta_low_log_filtered_Results.txt",
"3_500_500_meta.tsv", "edgeR")
v3_750_250edgeR = evaluate_DE("edgeR_3_750_250_meta_low_log_filtered_Results.txt",
"3_750_250_meta.tsv", "edgeR")
```

```
v3_1000_0edgeR = evaluate_DE("edgeR_3_1000_0_meta_low_log_filtered_Results.txt",
"3_1000_0_meta.tsv", "edgeR")
v6_500_500edgeR = evaluate_DE("edgeR_6_500_500_meta_low_log_filtered_Results.txt",
"6_500_500_meta.tsv", "edgeR")
v6_750_250edgeR = evaluate_DE("edgeR_6_750_250_meta_low_log_filtered_Results.txt",
"6_750_250_meta.tsv", "edgeR")
v6_1000_0edgeR = evaluate_DE("edgeR_6_1000_0_meta_low_log_filtered_Results.txt",
"6_1000_0_meta.tsv", "edgeR")
v9_500_500edgeR = evaluate_DE("edgeR_9_500_500_meta_low_log_filtered_Results.txt",
"9_500_500_meta.tsv", "edgeR")
v9_750_250edgeR = evaluate_DE("edgeR_9_750_250_meta_low_log_filtered_Results.txt",
"9_750_250_meta.tsv", "edgeR")
v9_1000_0edgeR = evaluate_DE("edgeR_9_1000_0_meta_low_log_filtered_Results.txt",
"9_1000_0_meta.tsv", "edgeR")
################################################################################
######
#create a graph with the results
DESeq_results_list <- list(v3_500_500DESeq,
                      v3_750_250DESeq,
                      v3_1000_0DESeq,
                      v6_500_500DESeq,
                      v6_750_250DESeq,
                      v6_1000_0DESeq,
                      v9_500_500DESeq,
                      v9_750_250DESeq,
                      v9_1000_0DESeq
)
edgeR_results_list <- list(v3_500_500edgeR,
                           v3_750_250edgeR,
                           v3_1000_0edgeR,
                           v6_500_500edgeR,
                           v6_750_250edgeR,
                           v6_1000_0edgeR,
                           v9_500_500edgeR,
                           v9_750_250edgeR,
                           v9_1000_0edgeR
)
limma_results_list <- list(v3_500_500limma,
                           v3_750_250limma,
                           v3_1000_0limma,
                           v6_500_500limma,
                           v6_750_250limma,
                           v6_1000_0limma,
                           v9_500_500limma,
                           v9_750_250limma,
                           v9_1000_0limma
)
noiseq_results_list <- list(v3_500_500noiseq,
                           v3_750_250noiseq,
                           v3_1000_0noiseq,
                           v6_500_500noiseq,
                           v6_750_250noiseq,
                           v6_1000_0noiseq,
                           v9_500_500noiseq,
                           v9_750_250noiseq,
                           v9_1000_0noiseq
)
#extract the necessary values, create a dataframe
specificity_values <- c()
sensitivity_values <- c()

for (result in DESeq_results_list) {
  specificity_values <- c(specificity_values, result[["specificity:"]])
  sensitivity_values <- c(sensitivity_values, result[["Sensitivity:"]])

}
for (result in edgeR_results_list) {
  specificity_values <- c(specificity_values, result[["specificity:"]])
```

```
  sensitivity_values <- c(sensitivity_values, result[["Sensitivity:"]])
}
for (result in limma_results_list) {
  specificity_values <- c(specificity_values, result[["specificity:"]])
  sensitivity_values <- c(sensitivity_values, result[["Sensitivity:"]])
}
for (result in noiseq_results_list) {
  specificity_values <- c(specificity_values, result[["specificity:"]])
  sensitivity_values <- c(sensitivity_values, result[["Sensitivity:"]])
}
df <-data.frame(
  x= specificity_values,
  y= sensitivity_values,
  group1 = rep(c(500,750,1000), times=3),
  group2 = rep(c(3,6,9), each=3,times=1),
  group3 = rep(c("DESeq2", "edgeR", "limma", "NOIseq"),each=9,times=1)
)
print(df)
# Create a column for symbols based on group3
df$symbol <- factor(df$group3, levels = c("DESeq2", "edgeR", "limma", "NOIseq"),
labels = c("DESeq2", "edgeR", "limma", "NOIseq"))
```

Overall figure sensitivity vs specificity

Using the dataframe created by the previous function

```
#figure overall
average_data <- df %>%
  group_by(group3) %>%
  summarize(avg_specificity = mean(x)*100, avg_sensitivity = mean(y)*100)
print(average_data)
# Create the plot
custom_colors <- c("edgeR" = "#7CAE00", "DESeq2" = "#F8766D", "NOIseq" =
"#00BFC4","limma" = "#C77CFF" )
ggplot(data = average_data, aes(x = avg_specificity, y = avg_sensitivity, color =
group3)) +
  geom_point(size = 3) +
  labs(title = "Overall Performance: Filtered by  FoldChange",
       x = "Specificity(%)",
       y = "Sensitivity(%)",
       color = NULL) +
  scale_color_manual(values = custom_colors)+
  theme_minimal()
```

Ratio of Upregulated vs Downregulated genes

Using the dataframe created by the main function, plots the sensitivity and specificity.

```
result <- df %>%
  group_by(group1, group3) %>%
  summarise(mean_sensitivity = mean(y))
result
#version 2 of effect of ratio
specificity_plot <- ggplot(df, aes(x = group1, y = x, color = group3)) +
  stat_summary(fun.y = "mean", geom = "point", size = 3) +
  labs(
    title = "Effect of Ratio of Increased DEG on Specificity",
    x = "Number of Overexpressed Genes",
    y = "Specificity",
    color = "Tool"
  ) +
  theme_minimal()

# Create a scatter plot for Sensitivity
sensitivity_plot <- ggplot(df, aes(x = group1, y = y, color = group3)) +
  stat_summary(fun.y = "mean", geom = "point", size = 3) +
  labs(
    title = "Effect of Ratio of Increased DEG on Sensitivity",
    x = "Number of Overexpressed Genes",
```

```
    y = "Sensitivity",
    color = "Tool"
  ) +
  theme_minimal()

# Display the Specificity and Sensitivity plots side by side
grid.arrange(specificity_plot, sensitivity_plot, ncol = 2)
```

Time efficiency Analyses and Figure

Here, we created a new function to reproduce the DEG analyses only for all tools. Start time was as the start of the function,.

```
time_evaluation_desq2<-function(matrixcount,metaFile)
{
  #take one file and return the time to do the DEF
  start <- Sys.time()
  ##start with DESeq2
  data <- read.table(paste("Data/",matrixcount,sep = ""), header=T, row.names=1)
  meta <- read.table(paste("MetaFull/",metaFile,sep = ""), header=T, row.names=1)
  dds <- DESeqDataSetFromMatrix(countData = data, colData = meta, design =
~Condition)
  # when have to perform the median of ratio of normalization
  dds <- estimateSizeFactors(dds)
  #We then retrieve the normalize count matrix
  normalized_counts <- counts(dds, normalized=TRUE)
  dds <- DESeq(dds)
  contrast_ce <- c("Condition", "cond2", "cond1")
  res_tableCE_unshrunken <- results(dds, contrast=contrast_ce, alpha = 0.05)
  res_tableCE <- lfcShrink(dds, contrast=contrast_ce,
res=res_tableCE_unshrunken,type="normal")
  #we use filter to get only the significant
  res_tableCE_tb <- res_tableCE %>%
    data.frame() %>%
    rownames_to_column(var="gene") %>%
    as_tibble()

  padj.cutoff <- 0.05
  lfc.cutoff <-0.58
  res_tableCE_tb <- res_tableCE_tb %>%
    mutate(threshold_CE = padj <  padj.cutoff & abs(log2FoldChange) > lfc.cutoff)
  return  (Sys.time() - start)
}
##############################################################################
##########################################
#time efficiency of edgeR
time_evaluation_edgeR<-function(matrixcount,metaFile)
{
  start <- Sys.time()
  data <-read.table(paste("Data/",matrixcount,sep = ""), header=T, row.names=1)
  meta <- read.table(paste("MetaFull/",metaFile,sep = ""), header=T, row.names=1)
  #create the DGEList object, takes at leat a count matrix
  samples <- substr(matrixcount, 1, nchar(matrixcount) - 4)
  split_values <- strsplit(matrixcount, "_")[[1]]
  #extract the forst number
  num_samples <- as.integer(split_values[1])
  num_conditions <- 2
  groups <-c(rep(1, num_samples), rep(2, num_samples))
  y<- DGEList(counts = data, group=groups)
  #filter genes with low counts, remove rows a small count(the smallest group size)
  keep <- filterByExpr(y)
  y <- y[keep, , keep.lib.sizes=FALSE]
  #normalization by the size of the library
  y <- normLibSizes((y))
  #estimate dispersion
  y <- estimateDisp(y)
  et <- exactTest(y)
  padj.cutoff <- 0.05
```

```
  lfc.cutoff <-0.58
  et_table = as.data.frame(et$table)
  filtered_genes <- et_table[et_table$padj < padj.cutoff & abs(et_table$logFC) >
lfc.cutoff ]
  return (Sys.time() - start)
}
################################################################################
########################
#time efficiency limma
time_evaluation_limma<-function(matrixcount,metaFile)
{
  start <- Sys.time()
  data <-read.table(paste("Data/",matrixcount,sep = ""), header=T, row.names=1)
  meta <- read.table(paste("MetaFull/",metaFile,sep = ""), header=T, row.names=1)

  dg <- DGEList(data)

  #Creating groups
  if (substring(metaFile, 1, 1) == "3"){
    group <- as.factor(c("Condition1", "Condition1", "Condition1",
                         "Condition2", "Condition2", "Condition2"))
  }
  else if (substring(metaFile, 1, 1) == "6"){
    group <- as.factor(c("Condition1", "Condition1", "Condition1",
                         "Condition1", "Condition1", "Condition1",
                         "Condition2", "Condition2", "Condition2",
                         "Condition2", "Condition2", "Condition2"))
  }
  else if (substring(metaFile, 1, 1) == "9"){
    group <- as.factor(c("Condition1", "Condition1", "Condition1",
                         "Condition1", "Condition1", "Condition1",
                         "Condition1", "Condition1", "Condition1",
                         "Condition2", "Condition2", "Condition2",
                         "Condition2", "Condition2", "Condition2",
                         "Condition2", "Condition2", "Condition2"))
  }

  dg$samples$group <- group
  ## Data preprocessing ####

  # Removing genes that are lowly expressed
  keep.exprs <- filterByExpr(dg, group=group)
  dg <- dg[keep.exprs,, keep.lib.sizes=FALSE]
  dim(dg)

  #normalisation
  dg <- calcNormFactors(dg, method = "TMM")
  dg$samples$norm.factors

  #Design matrix
  design <- model.matrix(~0+group)
  colnames(design) <- gsub("group", "", colnames(design))
  design


  contr.matrix <- makeContrasts(
    Cnd2vsCnd1 = Condition2-Condition1,
    levels = colnames(design))
  contr.matrix

  vdg <- voom(dg, design, plot=FALSE)
  vdg

  #Fitting linear model
  vfit <- lmFit(vdg, design)
  vfit <- contrasts.fit(vfit, contrasts=contr.matrix)
  efit <- eBayes(vfit)
```

```
  #Table with pvalue and mapping differentially expressed genes
  top.table <- topTable(efit, sort.by = "P", n = Inf) %>%
    mutate(upregulation=ifelse(adj.P.Val <0.05  & logFC > 0.58, 1, 0)) %>%
    mutate(downregulation=ifelse(adj.P.Val <0.05  & logFC < 0.58, 1, 0)) %>%
    mutate(differential.expression=ifelse(upregulation ==1|
                                          downregulation ==1, 1, 0))

  top.table <- top.table[,!names(top.table) %in%
                           c("logFC","AveExpr","t","P.Value","B")]


  return (Sys.time() - start)
}
##############################################################################
########################
#time study NOISeq
#ER_boot_list_fold[[j]] <- rownames(ER_res[ER_res$log2FC > 2,])
time_evaluation_NOISeq<-function(matrixcount,metaFile)
{
  start <- Sys.time()
  data <-read.table(paste("Data/",matrixcount,sep = ""), header=T, row.names=1)
  meta <- read.table(paste("MetaFull/",metaFile,sep = ""), header=T, row.names=1)
  NoisData<-readData(data=data, factors=meta)

  mynoiseq = noiseqbio(NoisData ,norm = "uqua", factor = "Condition", lc=0)

  res_tb<-mynoiseq@results %>%
    data.frame() %>%
    rownames_to_column(var="gene") %>%
    as_tibble()
  #q : threshold for q ,set to 0.95 for Noiseqbio
  q<-0.8
  mynoiseq.deg = degenes(mynoiseq, q = q, M = NULL)


  mynoiseq.deg1 = degenes(mynoiseq, q = q, M = "up")
  mynoiseq.deg2 = degenes(mynoiseq, q = q, M = "down")
  head(mynoiseq@results[[1]])
  mynoiseq.deg<-mynoiseq.deg%>%
    data.frame() %>%
    rownames_to_column(var="gene") %>%
    as_tibble()
  print(mynoiseq)
  return (Sys.time() - start)
}

##############################################################################
########################
##

time_n3_deseq2<-0
time_n6_deseq2<-0
time_n9_deseq2<-0

time_n6_edgeR<-0
time_n3_edgeR<-0
time_n9_edgeR<-0

time_n3_limma<-0
time_n6_limma<-0
time_n9_limma<-0

time_n3_NOISeq<-0
time_n6_NOISeq<-0
time_n9_NOISeq<-0


for(i in 1:10)
```

```
{
  print(i)
  time_n3_deseq2<-time_n3_deseq2+
time_evaluation_desq2("3_500_500.tsv","3_500_500_full.txt")+time_evaluation_desq2("
3_750_250.tsv","3_750_250_full.txt")+time_evaluation_desq2("3_1000_0.tsv","3_1000_0
_full.txt")
  time_n6_deseq2<- time_n6_deseq2 +
time_evaluation_desq2("6_500_500.tsv","6_500_500_full.txt")+time_evaluation_desq2("
6_750_250.tsv","6_750_250_full.txt")+time_evaluation_desq2("6_1000_0.tsv","6_1000_0
_full.txt")
  time_n9_deseq2<-time_n9_deseq2 +
time_evaluation_desq2("9_500_500.tsv","9_500_500_full.txt")+time_evaluation_desq2("
9_750_250.tsv","9_750_250_full.txt")+time_evaluation_desq2("9_1000_0.tsv","9_1000_0
_full.txt")
  #time_n3_edgeR<-time_n3_edgeR +
time_evaluation_edgeR("3_500_500.tsv","3_500_500_full.txt")+time_evaluation_edgeR("
3_750_250.tsv","3_750_250_full.txt")+time_evaluation_edgeR("3_1000_0.tsv","3_1000_0
_full.txt")
  #time_n6_edgeR<-time_n6_edgeR +
time_evaluation_edgeR("6_500_500.tsv","6_500_500_full.txt")+time_evaluation_edgeR("
6_750_250.tsv","6_750_250_full.txt")+time_evaluation_edgeR("6_1000_0.tsv","6_1000_0
_full.txt")
  #time_n9_edgeR<-time_n9_edgeR +
time_evaluation_edgeR("9_500_500.tsv","9_500_500_full.txt")+time_evaluation_edgeR("
9_750_250.tsv","9_750_250_full.txt")+time_evaluation_edgeR("9_1000_0.tsv","9_1000_0
_full.txt")
  #time_n3_limma<-time_n3_limma +
time_evaluation_limma("3_500_500.tsv","3_500_500_full.txt")+time_evaluation_limma("
3_750_250.tsv","3_750_250_full.txt")+time_evaluation_limma("3_1000_0.tsv","3_1000_0
_full.txt")
  #time_n6_limma<-time_n6_limma +
time_evaluation_limma("6_500_500.tsv","6_500_500_full.txt")+time_evaluation_limma("
6_750_250.tsv","6_750_250_full.txt")+time_evaluation_limma("6_1000_0.tsv","6_1000_0
_full.txt")
  #time_n9_limma<-time_n9_limma +
time_evaluation_limma("9_500_500.tsv","9_500_500_full.txt")+time_evaluation_limma("
9_750_250.tsv","9_750_250_full.txt")+time_evaluation_limma("9_1000_0.tsv","9_1000_0
_full.txt")
  #time_n3_NOISeq<-time_n3_NOISeq +
time_evaluation_NOISeq("3_500_500.tsv","3_500_500_full.txt")+time_evaluation_NOISeq
("3_750_250.tsv","3_750_250_full.txt")+time_evaluation_NOISeq("3_1000_0.tsv","3_100
0_0_full.txt")
  #time_n6_NOISeq<-time_n6_NOISeq +
time_evaluation_NOISeq("6_500_500.tsv","6_500_500_full.txt")+time_evaluation_NOISeq
("6_750_250.tsv","6_750_250_full.txt")+time_evaluation_NOISeq("6_1000_0.tsv","6_100
0_0_full.txt")
  #time_n9_NOISeq<-time_n9_NOISeq +
time_evaluation_NOISeq("9_500_500.tsv","9_500_500_full.txt")+time_evaluation_NOISeq
("9_750_250.tsv","9_750_250_full.txt")+time_evaluation_NOISeq("9_1000_0.tsv","9_100
0_0_full.txt")
}

your_data_frame <- data.frame(
  Tool = rep(c("DESeq2", "edgeR", "limma", "NOISeq"), each = 3),
  NumberOfSamples = rep(c(3, 6, 9), times = 4),
  Time = c(
    time_n3_deseq2, time_n6_deseq2, time_n9_deseq2,
    time_n3_edgeR, time_n6_edgeR, time_n9_edgeR,
    time_n3_limma, time_n6_limma, time_n9_limma,
    time_n3_NOISeq, time_n6_NOISeq, time_n9_NOISeq
  )/30
)
custom_colors <- c("edgeR" = "#7CAE00", "DESeq2" = "#F8766D", "NOISeq" =
"#00BFC4","limma" = "#C77CFF" )
print(your_data_frame)
ggplot(your_data_frame, aes(x = NumberOfSamples, y = Time, color = Tool)) +
  geom_point(size = 3) +
  geom_line() +
```

```
  labs(
    x = "No of Samples",
    y = "Run time(sec)",
    color = NULL
  ) +
  scale_color_manual(values = custom_colors)+
  ggtitle("Runtime Analysis") +
  theme_minimal()+
  theme(plot.title = element_text(hjust = 0.5))
```