Transcriptomic insights into late-onset Alzheimer's Disease: differential gene expression analysis

Silvana Yalú Cristo Martínez

2025-02-05

Índice de contenidos

Introduction

Late-onset Alzheimer's disease (LOAD) is the most prevalent form of dementia, characterized by progressive cognitive decline, memory loss, and severe neurodegeneration. Despite extensive research, the molecular mechanisms underlying LOAD remain only partially understood. While hallmark features such as beta-amyloid plaques and neurofibrillary tangles are well-documented, broader transcriptomic alterations that drive disease progression require further investigation.

High-throughput RNA sequencing (RNA-Seq) has revolutionized the study of neurodegenerative diseases by enabling the comprehensive profiling of gene expression at an unprecedented resolution. This approach not only facilitates the identification of differentially expressed genes but also provides insights into regulatory pathways and novel RNA species, such as long non-coding RNAs (lncRNAs), which are increasingly implicated in neuronal function and dysfunction.

For this analysis, publicly available RNA-Seq data from hippocampal samples of LOAD patients and age-matched controls were utilized, originally published by *(Magistri et al. 2015)*. The hippocampus, a region critical for learning and memory, is one of the first to be affected in Alzheimer's disease, making it a key target for studying disease pathology. The original study highlighted significant transcriptomic alterations, including disruptions in neurovascular integrity, beta-amyloid clearance pathways, and widespread dysregulation of lncRNAs.

Building upon these findings, this analysis aimed to identify differentially expressed genes, validate previously reported molecular signatures, and explore new potential LOAD-related mechanisms. Notably, genes such as SERPINE1 and VGLL3 were identified as significantly differentially expressed, reinforcing prior evidence on their role in LOAD pathology. The results underscore the value of transcriptomic studies in uncovering molecular drivers of disease and provide a foundation for further experimental validation and therapeutic exploration.

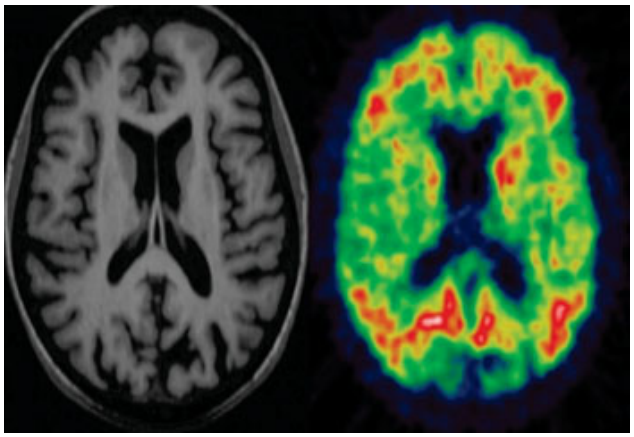

Figure 1: MRI and PET scans of a patient with Alzheimer's disease. Edison, P. Neurology Imaging Unit, Centre for Translational Imaging, Imperial College London). Rerieved from https://www.medimaging.es/medicina-nuclear/articles/294756318/combinan-pet-y-rm-para-visualizar-desarrollo-de-alzheimer.html.

Research question

This analysis seeks to determine which genes are differentially expressed in the hippocampus of late-onset Alzheimer's disease (LOAD) patients compared to age-matched controls. By identifying transcriptomic alterations, the goal is to gain insight into potential molecular mechanisms associated with the disease.

Objectives:

General objective

To identify differentially expressed genes associated with late-onset Alzheimer's disease (LOAD) using RNA-Seq data from hippocampal samples, and to explore their potential roles in the molecular mechanisms underlying the disease.

Specific objectives

- To preprocess and normalize RNA-Seq count data using appropriate bioinformatics pipelines.
- To perform differential expression analysis to identify genes significantly associated with LOAD compared to age-matched controls.
- To integrate biological interpretation of key genes, including protein-coding and long non-coding RNAs (lncRNAs), based on literature and functional annotations.
- To visualize and present the results through plots and heatmaps to provide clear insights into the transcriptomic differences between control and LOAD samples.

Methodology

1.1 Selection and collection of data

In this section, the analysis was initiated by leveraging the recount3 database, which contains preprocessed RNA-Seq data from various studies. Using the ***available_projects()*** function, all human datasets were retrieved to explore potential projects for analysis. The ***interactiveDisplayBase*** package provides an easy-to-use interface to examine metadata and select a dataset aligned with the research objectives.

```
# Load required libraries
# recount3: Provides access to recount3 data repository.
# interactiveDisplayBase: Allows an interactive visualization of available projects.
library("recount3")
library("interactiveDisplayBase")
```

```
# Retrieve available human RNA-Seq projects from recount3
# This function lists all projects available for analysis.
human_projects <- available_projects()
```

```
## 2025-02-07 00:20:55.583542 caching file sra.recount_project.MD.gz.
```

```
## 2025-02-07 00:20:56.270187 caching file gtex.recount_project.MD.gz.
```

```
## 2025-02-07 00:20:56.839284 caching file tcga.recount_project.MD.gz.
```

```
# Display the list of projects interactively
# Use this to explore the datasets and select one relevant to your research question.
#interactiveDisplayBase::display(human_projects)
```

1.2 Data proccesing

In this step, the RNA-Seq data associated with project SRP056604 from the recount3 database was extracted. The ***create_rse function*** structures the raw data into a ***RangedSummarizedExperiment*** object, which facilitates downstream analysis. Assigned read counts were calculated using the ***compute_read_counts*** function, ensuring the data is ready for normalization and differential expression analysis. Lastly, the presence of the "counts" matrix were verified, confirming successful computation.

```r
# Create a RangedSummarizedExperiment object
# This function extracts and structures the RNA-Seq data for project SRP056604.
# "data_sources" refers to the type of dataset we are analyzing.
rse_gene_SRP056604 <- create_rse(
    subset(
        human_projects,
        project == "SRP056604" & project_type == "data_sources"
    )
)
```

```
## 2025-02-07 00:21:05.360402 downloading and reading the metadata.

## 2025-02-07 00:21:06.249678 caching file sra.sra.SRP056604.MD.gz.

## 2025-02-07 00:21:06.86001 caching file sra.recount_project.SRP056604.MD.gz.

## 2025-02-07 00:21:07.47255 caching file sra.recount_qc.SRP056604.MD.gz.

## 2025-02-07 00:21:08.162025 caching file sra.recount_seq_qc.SRP056604.MD.gz.

## 2025-02-07 00:21:08.812108 caching file sra.recount_pred.SRP056604.MD.gz.

## 2025-02-07 00:21:09.080834 downloading and reading the feature information.

## 2025-02-07 00:21:09.511441 caching file human.gene_sums.G026.gtf.gz.

## 2025-02-07 00:21:10.700568 downloading and reading the counts: 8 samples across 63856 features.

## 2025-02-07 00:21:11.111317 caching file sra.gene_sums.SRP056604.G026.gz.

## 2025-02-07 00:21:11.418369 constructing the RangedSummarizedExperiment (rse) object.
```

```r
# Verify that the data has been successfully loaded
# This displays metadata about the loaded dataset.
rse_gene_SRP056604
```

```
## class: RangedSummarizedExperiment
## dim: 63856 8
## metadata(8): time_created recount3_version ... annotation recount3_url
## assays(1): raw_counts
## rownames(63856): ENSG00000278704.1 ENSG00000277400.1 ...
##   ENSG00000182484.15_PAR_Y ENSG00000227159.8_PAR_Y
## rowData names(10): source type ... havana_gene tag
## colnames(8): SRR1931812 SRR1931813 ... SRR1931818 SRR1931819
## colData names(175): rail_id external_id ...
##   recount_pred.curated.cell_line BigWigURL
```

```r
# Compute assigned read counts
# The counts represent the number of reads mapped to genes in the dataset.
assay(rse_gene_SRP056604, "counts") <- compute_read_counts(rse_gene_SRP056604)

# Confirm that the read counts have been calculated
# The "counts" matrix should now be accessible within the RangedSummarizedExperiment object.
assayNames(rse_gene_SRP056604)
```

```
## [1] "raw_counts" "counts"
```

1.3 Quality control and data cleaning

In this section, the metadata associated with the dataset was standardized. First, the initial entries of the ***sra.sample_attributes*** were inspected to identify potential issues, such as missing or inconsistent values. The next step was to replace occurrences of "Not available" in the "apoe genotype" field with NA, a standard placeholder for missing data in R. This step ensures that the dataset is clean and ready for subsequent analysis, minimizing errors caused by non-standard missing value formats.

```r
# Display the first sample attributes
# This step allows inspection of the first 8 entries in the sample attributes,
# focusing on metadata to identify missing or inconsistent information.
rse_gene_SRP056604$sra.sample_attributes[1:8]
```

```
## [1] "apoe genotype;;3/3|braak stage;;VI|gender;;female|source_name;;LOAD_hippocampi|subject group;;L
## [2] "apoe genotype;;3/3|braak stage;;V|gender;;female|source_name;;LOAD_hippocampi|subject group;;LOA
## [3] "apoe genotype;;3/3|braak stage;;VI|gender;;male|source_name;;LOAD_hippocampi|subject group;;LOAD
## [4] "apoe genotype;;3/3|braak stage;;VI|gender;;female|source_name;;LOAD_hippocampi|subject group;;L
## [5] "apoe genotype;;2/3|braak stage;;I|gender;;female|source_name;;age-matched control_hippocampi|sub
## [6] "apoe genotype;;Not available|braak stage;;II|gender;;male|source_name;;age-matched control_hippo
## [7] "apoe genotype;;3/3|braak stage;;II|gender;;male|source_name;;age-matched control_hippocampi|subj
## [8] "apoe genotype;;2/3|braak stage;;II|gender;;female|source_name;;age-matched control_hippocampi|su
```

```r
# Filter samples with missing information in the "apoe genotype" attribute
# Replace "Not available" with NA to standardize missing values for downstream processing.
rse_gene_SRP056604$sra.sample_attributes <- gsub(
    "apoe genotype;;Not available",
    "apoe genotype;;NA",
    rse_gene_SRP056604$sra.sample_attributes
)
```

```r
# Verify the updated attributes after filtering
# Inspect the first 8 attributes again to confirm the replacements were made.
rse_gene_SRP056604$sra.sample_attributes[1:8]
```

```
## [1] "apoe genotype;;3/3|braak stage;;VI|gender;;female|source_name;;LOAD_hippocampi|subject group;;L
## [2] "apoe genotype;;3/3|braak stage;;V|gender;;female|source_name;;LOAD_hippocampi|subject group;;LOA
## [3] "apoe genotype;;3/3|braak stage;;VI|gender;;male|source_name;;LOAD_hippocampi|subject group;;LOAD
## [4] "apoe genotype;;3/3|braak stage;;VI|gender;;female|source_name;;LOAD_hippocampi|subject group;;L
## [5] "apoe genotype;;2/3|braak stage;;I|gender;;female|source_name;;age-matched control_hippocampi|sub
## [6] "apoe genotype;;NA|braak stage;;II|gender;;male|source_name;;age-matched control_hippocampi|subje
## [7] "apoe genotype;;3/3|braak stage;;II|gender;;male|source_name;;age-matched control_hippocampi|subj
## [8] "apoe genotype;;2/3|braak stage;;II|gender;;female|source_name;;age-matched control_hippocampi|su
```

In this step, the metadata embedded within the SRA attributes was expanded using the ***expand_sra_attributes function***. This allowed for a clearer representation of key variables, which are crucial for accurate data interpretation. The second line of code verifies the proper expansion by identifying and displaying columns labeled with ***sra_attribute***, ensuring that all necessary metadata is accessible for subsequent analyses.

```
# Expand SRA attributes
# This function processes and expands metadata stored within the SRA attributes of
# the dataset, making it more accessible for downstream analysis.
rse_gene_SRP056604 <- expand_sra_attributes(rse_gene_SRP056604)

# Display columns containing SRA attributes
# Identify and inspect columns in the dataset metadata that contain "sra_attribute"
# to confirm proper expansion of the attributes.
colData(rse_gene_SRP056604)[
    , grep("^sra_attribute", colnames(colData(rse_gene_SRP056604)))
]
```

```
## DataFrame with 8 rows and 6 columns
##            sra_attribute.apoe_genotype sra_attribute.braak_stage
##                            <character>               <character>
## SRR1931812                         3/3                        VI
## SRR1931813                         3/3                         V
## SRR1931814                         3/3                        VI
## SRR1931815                         3/3                        VI
## SRR1931816                         2/3                         I
## SRR1931817                          NA                        II
## SRR1931818                         3/3                        II
## SRR1931819                         2/3                        II
##            sra_attribute.gender sra_attribute.source_name
##                     <character>              <character>
## SRR1931812              female          LOAD_hippocampi
## SRR1931813              female          LOAD_hippocampi
## SRR1931814                male          LOAD_hippocampi
## SRR1931815              female          LOAD_hippocampi
## SRR1931816              female    age-matched control_..
## SRR1931817                male    age-matched control_..
## SRR1931818                male    age-matched control_..
## SRR1931819              female    age-matched control_..
##            sra_attribute.subject_group sra_attribute.tissue
##                            <character>          <character>
## SRR1931812      LOAD (late onset of ..          Hippocampus
## SRR1931813      LOAD (late onset of ..          Hippocampus
## SRR1931814      LOAD (late onset of ..          Hippocampus
## SRR1931815      LOAD (late onset of ..          Hippocampus
## SRR1931816         age-matched control          Hippocampus
## SRR1931817         age-matched control          Hippocampus
## SRR1931818         age-matched control          Hippocampus
## SRR1931819         age-matched control          Hippocampus
```

This block focuses on ensuring the data types of key attributes are correctly assigned, which is essential for accurate analysis and modeling. Initially, the structure of the relevant columns was inspected to identify any necessary changes. Subsequently, specific attributes were converted to the appropriate types, such as character or factor, depending on their intended use. For instance:

- ***apoe_genotype*** was kept as character since it is not involved in the model.
- ***braak_stage was*** converted to a factor with specified levels to reflect its ordered nature.
- ***gender was converted*** to a factor with standardized lowercase values.

The final step provides a summary of the modified columns, verifying that the data is now clean and ready for further processing.

```
# Check data types of relevant columns
# Display the structure of columns containing SRA attributes to ensure the data
# types are appropriate.
str(as.data.frame(colData(rse_gene_SRP056604)[
  ,
  grepl("^sra_attribute", colnames(colData(rse_gene_SRP056604)))
]))
```

```
## 'data.frame':    8 obs. of  6 variables:
##  $ sra_attribute.apoe_genotype: chr  "3/3" "3/3" "3/3" "3/3" ...
##  $ sra_attribute.braak_stage  : chr  "VI" "V" "VI" "VI" ...
##  $ sra_attribute.gender       : chr  "female" "female" "male" "female" ...
##  $ sra_attribute.source_name  : chr  "LOAD_hippocampi" "LOAD_hippocampi" "LOAD_hippocampi" "LOAD_hipp
##  $ sra_attribute.subject_group: chr  "LOAD (late onset of Alzheimer's disease)" "LOAD (late onset of
##  $ sra_attribute.tissue       : chr  "Hippocampus" "Hippocampus" "Hippocampus" "Hippocampus" ...
```

```
# Convert specific attributes to the appropriate type
# Modify data types as required for downstream analysis.
rse_gene_SRP056604$sra_attribute.apoe_genotype <-
  as.character(rse_gene_SRP056604$sra_attribute.apoe_genotype)
# Keep as character if not used in the model

rse_gene_SRP056604$sra_attribute.braak_stage <- factor(
  rse_gene_SRP056604$sra_attribute.braak_stage,
  levels = c("I", "II", "V", "VI")
)
# Assign explicit levels for ordered data

rse_gene_SRP056604$sra_attribute.gender <- factor(
  tolower(rse_gene_SRP056604$sra_attribute.gender)
)
# Convert to factor with lowercase values

rse_gene_SRP056604$sra_attribute.source_name <-
  as.character(rse_gene_SRP056604$sra_attribute.source_name)
# Maintain as text

rse_gene_SRP056604$sra_attribute.subject_group <- factor(
  rse_gene_SRP056604$sra_attribute.subject_group
)
# Convert to factor

rse_gene_SRP056604$sra_attribute.tissue <- factor(
  rse_gene_SRP056604$sra_attribute.tissue
)
# Convert to factor
```

```
# Summarize updated attributes
# Display a summary of SRA attribute columns to confirm proper conversion and organization.
summary(as.data.frame(colData(rse_gene_SRP056604)[
  ,
  grepl("^sra_attribute", colnames(colData(rse_gene_SRP056604)))
]))
```

```
##  sra_attribute.apoe_genotype sra_attribute.braak_stage sra_attribute.gender
##  Length:8                    I :1                      female:5
##  Class :character            II:3                      male  :3
##  Mode  :character            V :1
##                              VI:3
##  sra_attribute.source_name
##  Length:8
##  Class :character
##  Mode  :character
##
##                            sra_attribute.subject_group  sra_attribute.tissue
##  age-matched control                          :4           Hippocampus:8
##  LOAD (late onset of Alzheimer's disease):4
##
##
```

In this step, the proportion of assigned gene counts was calculated for each sample. This proportion provides an indicator of data quality, as it reflects how efficiently reads were assigned to genes in the dataset. A higher proportion suggests better data quality, as it indicates a lower number of unassigned reads. This information can later be used to filter low-quality samples or outliers, ensuring that downstream analyses are reliable.

```
# Calculate the proportion of assigned gene counts
# This metric helps evaluate the quality of the data by determining the proportion
# of total reads assigned to genes for each sample.
rse_gene_SRP056604$assigned_gene_prop <-
    rse_gene_SRP056604$recount_qc.gene_fc_count_all.assigned /
    rse_gene_SRP056604$recount_qc.gene_fc_count_all.total
```

This section focuses on visualizing the quality of the samples based on the proportion of assigned genes.

- A boxplot was created to illustrate the relationship between subject groups (controls and Alzheimer's patients) and the proportion of assigned genes. This visualization helps assess whether there are significant differences between groups in terms of data quality.
- A histogram was generated to evaluate the distribution of the proportion of assigned genes across all samples, highlighting any potential outliers or poorly sequenced samples.
- Samples with a proportion of assigned genes lower than 0.3 were flagged for potential exclusion. The filtering results are summarized using a contingency table.

```
# Load ggplot2 for visualization
library(ggplot2)

# Create a boxplot to examine the relationship between subject groups and the
# proportion of assigned genes
ggplot(as.data.frame(colData(rse_gene_SRP056604)), aes(
  x = sra_attribute.subject_group, y = assigned_gene_prop
```

```
)) +
  geom_boxplot() +
  theme_bw(base_size = 10) +
  labs(
    x = "Subject group (control vs disease)",
    y = "Proportion of assigned genes",
    title = "Relationship between subject group and proportion of assigned Genes"
  )
```

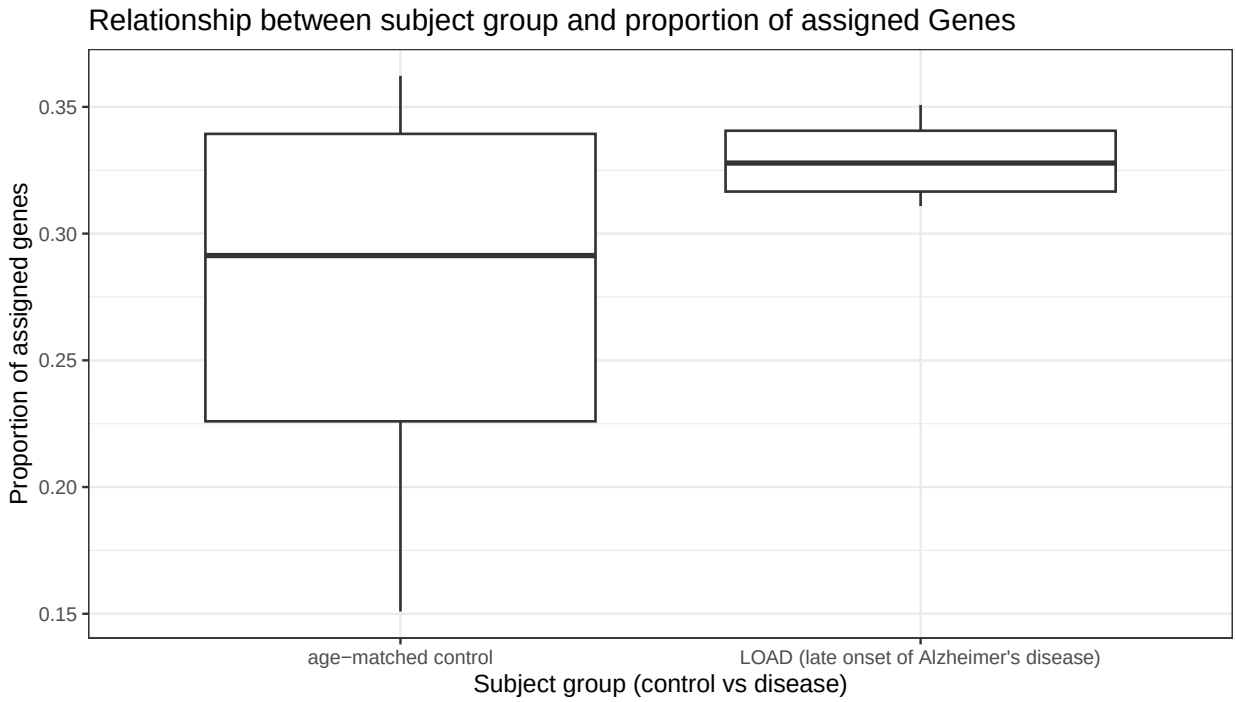

Figure 2: Boxplot of controls and LOAD patients

```
# Summarize differences in assigned gene proportion by subject group
with(
  colData(rse_gene_SRP056604),
  tapply(
    assigned_gene_prop,
    sra_attribute.subject_group,
    summary
  )
)
```

```
## $`age-matched control`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1509  0.2259  0.2913  0.2740  0.3394  0.3622
##
## $`LOAD (late onset of Alzheimer's disease)`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3109  0.3166  0.3278  0.3293  0.3406  0.3508
```

```
# Create a histogram to visualize the distribution of the proportion of assigned genes
hist(rse_gene_SRP056604$assigned_gene_prop,
     main = "Histogram of assigned gene proportion",
     xlab = "Proportion of assigned genes",
     ylab = "Frequency",
     col = "gray",
     border = "black")
```

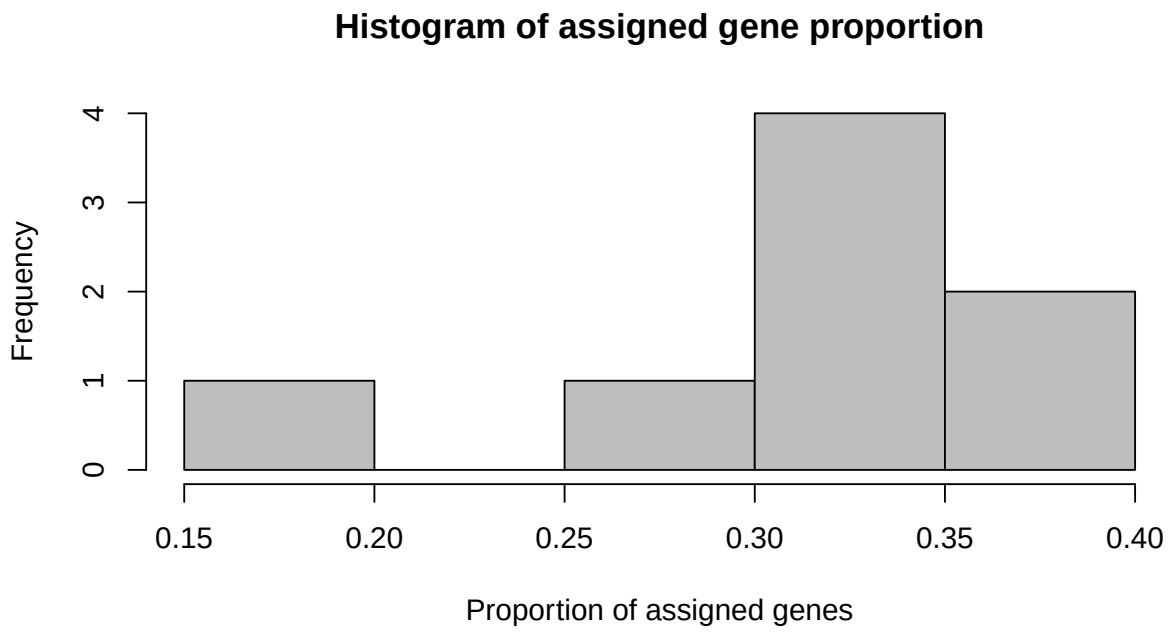

Figure 3: Histogram of the distribution of the proportion of assigned genes

In this part two samples had a proportion of assigned genes below 0.3, suggesting lower sequencing quality. Initially, the plan was to exclude these samples to maintain a consistent quality threshold. However, upon further investigation, it was determined that both samples belonged to the control group, which would have introduced a significant imbalance between the control and disease groups.

Given that our dataset consists of only 8 samples (4 controls and 4 disease cases), removing these two samples would have compromised the validity of downstream statistical analyses. To preserve the integrity of the experimental design, it was decided to retain all 8 samples.

```
# Filter samples with an assigned gene proportion < 0.3 and show the count
table(rse_gene_SRP056604$assigned_gene_prop < 0.3)
```

```
##
## FALSE  TRUE
##     6     2
```

```
# Decision: Retain all samples, including those flagged with < 0.3 proportion
```

The average expression level for each gene across all samples was calculated using the ***rowMeans()*** function. A statistical summary of the mean expression levels was obtained with the ***summary()***

function, highlighting key metrics such as the minimum, maximum, and quartiles. To ensure the ability to revert to the original dataset if needed, a copy of the unfiltered dataset was saved in the object ***rse_gene_SRP056604_unfiltered***. Genes with a mean expression level lower than 0.1 were then filtered out, as these are typically considered noise and are less likely to be biologically significant. The dimensions of the filtered dataset were checked using the ***dim()*** function to confirm the number of retained genes and samples. Additionally, the percentage of genes retained after filtering was calculated, yielding a value of 78.16%, indicating that a substantial proportion of biologically relevant genes were preserved while minimizing noise.

```
# Calculate the mean expression level for each gene across all samples
gene_means <- rowMeans(assay(rse_gene_SRP056604, "counts"))

# Summarize the distribution of mean gene expression
summary(gene_means)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0      10    1349     265 4516319
```

```
# Backup the original dataset before filtering (unfiltered dataset)
rse_gene_SRP056604_unfiltered <- rse_gene_SRP056604

# Filter genes with a mean expression level greater than 0.1
rse_gene_SRP056604 <- rse_gene_SRP056604[gene_means > 0.1, ]

# Check the dimensions of the filtered dataset (rows: genes, columns: samples)
dim(rse_gene_SRP056604)
```

```
## [1] 49909     8
```

```
# Calculate the percentage of genes retained after filtering
porcentaje_genes_retenidos <- round(nrow(rse_gene_SRP056604) / nrow(rse_gene_SRP056604_unfiltered) * 10

# Output the percentage of genes retained
porcentaje_genes_retenidos
```

```
## [1] 78.16
```

1.4 Data normalization

Normalization was performed using the edgeR package to account for differences in library sizes. A DGEList object was created to store the count data and gene metadata from ***rse_gene_SRP056604***. The normalization factors were calculated with ***calcNormFactors()***, ensuring that technical variations were minimized. The results were checked using ***dge$samples***, which showed library sizes and normalization factors.

```
# Load the edgeR library
library("edgeR")

# Create a DGEList object with count data and metadata
dge <- DGEList(
```

```
  counts = assay(rse_gene_SRP056604, "counts"),  # Add count data
  genes = rowData(rse_gene_SRP056604)           # Add gene metadata
)

# Calculate normalization factors to adjust for library size differences
dge <- calcNormFactors(dge)

# View the normalization factors and library sizes
dge$samples
```

```
##            group lib.size norm.factors
## SRR1931812     1 99676894    1.0038623
## SRR1931813     1 87220194    1.0676906
## SRR1931814     1 90867982    1.0515685
## SRR1931815     1 89077564    1.1214461
## SRR1931816     1 87375765    0.8292094
## SRR1931817     1 61887308    1.0000695
## SRR1931818     1 87486227    0.9530671
## SRR1931819     1 85714095    1.0010288
```

1.5 Differential expression analysis: model construction

The first step in the differential expression analysis involved creating a design ***matrix (mod)*** using the ***model.matrix()*** function. This matrix incorporates key variables: subject group, gender, and the proportion of assigned genes. These variables were selected to capture relevant biological and technical variability in the data. The columns of the design matrix were verified using ***colnames(mod)*** to ensure all terms were correctly specified.

```
# Construct the design matrix for differential expression analysis
mod <- model.matrix(~ sra_attribute.subject_group + sra_attribute.gender + assigned_gene_prop,
                    data = colData(rse_gene_SRP056604))

# Verify the columns of the model
colnames(mod)
```

```
## [1] "(Intercept)"
## [2] "sra_attribute.subject_groupLOAD (late onset of Alzheimer's disease)"
## [3] "sra_attribute.gendermale"
## [4] "assigned_gene_prop"
```

1.5.1 Differential expression analysis: voom normalization

The next step in the analysis utilized the voom function from the limma package to normalize the RNA-seq count data.

The red line in the plot represents the smoothed mean-variance trend, and the scatterplot shows individual genes with their variability. The trend demonstrates a reduction in variability as the expression values increase, which is expected in properly normalized RNA-seq data.

```r
# Load the limma library
library("limma")

# Apply voom normalization and generate the mean-variance trend plot
vGene <- voom(dge, mod, plot = TRUE)
```

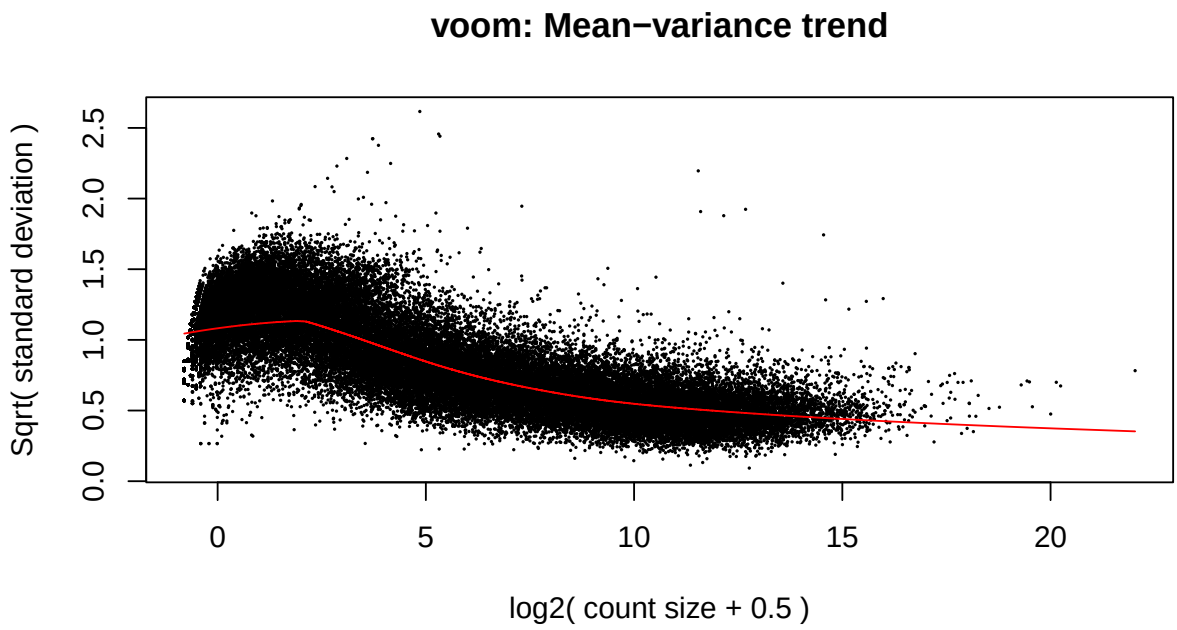

Figure 4: Mean-variance trend plot

1.5.2 Differential expression analysis: linear model and Bayesian statistics

The analysis proceeds with fitting a linear model to the normalized data, followed by applying empirical Bayes moderation using eBayes. These steps enhance the stability of the variance estimates, which is crucial given the high-dimensional nature of RNA-seq data. The moderated statistics are used to identify differentially expressed genes.

The table of results contains key metrics, including fold changes, p-values, and adjusted p-values for all genes in the dataset. The dimensions of the results confirmed that 49,909 genes were analyzed across all samples.

```r
# Fit the linear model and apply empirical Bayes moderation
eb_results <- eBayes(lmFit(vGene))

# Extract the results table for all genes without sorting
de_results <- topTable(
  eb_results,
  coef = 2,
  number = nrow(rse_gene_SRP056604),
```

```
  sort.by = "none"
)

# Confirm the dimensions of the results table
dim(de_results)
```

```
## [1] 49909    16
```

```
# Inspect the first few results
head(de_results)
```

```
##                     source type bp_length phase           gene_id
## ENSG00000223972.5  HAVANA gene      1735    NA ENSG00000223972.5
## ENSG00000278267.1 ENSEMBL gene        68    NA ENSG00000278267.1
## ENSG00000227232.5  HAVANA gene      1351    NA ENSG00000227232.5
## ENSG00000284332.1 ENSEMBL gene       138    NA ENSG00000284332.1
## ENSG00000243485.5  HAVANA gene      1021    NA ENSG00000243485.5
## ENSG00000237613.2  HAVANA gene      1219    NA ENSG00000237613.2
##                                       gene_type   gene_name level
## ENSG00000223972.5 transcribed_unprocessed_pseudogene     DDX11L1     2
## ENSG00000278267.1                            miRNA   MIR6859-1     3
## ENSG00000227232.5             unprocessed_pseudogene      WASH7P     2
## ENSG00000284332.1                            miRNA   MIR1302-2     3
## ENSG00000243485.5                          lincRNA MIR1302-2HG     2
## ENSG00000237613.2                          lincRNA     FAM138A     2
##                            havana_gene        tag       logFC    AveExpr
## ENSG00000223972.5 OTTHUMG00000000961.2       <NA> -0.25902276 -0.7172795
## ENSG00000278267.1                 <NA>       <NA>  0.66341259 -3.6100230
## ENSG00000227232.5 OTTHUMG00000000958.1       <NA>  0.07905401  3.8686061
## ENSG00000284332.1                 <NA>       <NA> -0.35856731 -6.8686660
## ENSG00000243485.5 OTTHUMG00000000959.2 ncRNA_host  1.40077705 -2.6573857
## ENSG00000237613.2 OTTHUMG00000000960.1       <NA>  0.94762034 -5.9979430
##                           t    P.Value adj.P.Val         B
## ENSG00000223972.5 -0.6041461 0.55687688 0.9660770 -5.277891
## ENSG00000278267.1  0.6794498 0.50961056 0.9619637 -4.848933
## ENSG00000227232.5  0.3490650 0.73302480 0.9795051 -6.026472
## ENSG00000284332.1 -0.3740723 0.71480714 0.9778249 -4.944031
## ENSG00000243485.5  1.9606901 0.07328563 0.8955682 -4.017234
## ENSG00000237613.2  0.8736365 0.39928116 0.9438409 -4.752762
```

```
# Count genes with a p-value < 0.05
table(de_results$P.Value < 0.05)
```

```
##
## FALSE  TRUE
## 47058  2851
```

1.5.3 Results: differential expression analysis (visualizations)

1. MA plot: visualization of differential expression

The MA plot provides a clear visualization of gene expression changes across all analyzed genes, emphasizing the log fold change (logFC) against the log-average expression (AveExpr). This graphical representation allows easy identification of genes with significant differential expression.

```
# Load required library
library(ggplot2)

# Prepare data frame for ggplot
ma_data <- data.frame(
  AveExpr = de_results$AveExpr,
  logFC = de_results$logFC,
  PValue = de_results$P.Value,
  gene = de_results$gene_name
)

# Label significant genes
ma_data$Significance <- ifelse(
  ma_data$PValue < 0.05 & abs(ma_data$logFC) > 1,
  "Significant",
  "Non-significant"
 )

# Generate MA plot
ggplot(ma_data, aes(x = AveExpr, y = logFC, color = Significance)) +
  geom_point(alpha = 0.6, size = 1.5) +
  scale_color_manual(values = c("Significant" = "cadetblue", "Non-significant" = "grey")) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "black", size = 0.5) +
  labs(
    title = "MA Plot",
    x = "Log-Average Expression",
    y = "Log Fold Change",
    color = "Significance"
  ) +
  theme_minimal(base_size = 15)
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

2. Volcano plot: highlighting differentially expressed genes

The Volcano plot effectively showcases genes with significant differential expression by plotting the log fold change (logFC) against the -log10 (p-value). This type of visualization allows to identify genes with both statistically significant and biologically meaningful changes in expression.

```
# Load required library
library(ggplot2)

# Filter significant genes for labeling
top_genes <- de_results[de_results$P.Value < 0.05 & abs(de_results$logFC) > 1, ]
top_genes <- top_genes[order(top_genes$P.Value), ]  # Order by P-value
top_genes <- head(top_genes, n = 10)  # Select top 10 significant genes
```

Figure 5: MA plot of gene expression changes

```r
# Create Volcano Plot
ggplot(de_results, aes(x = logFC, y = -log10(P.Value))) +
  geom_point(alpha = 0.5) +
  geom_point(data = top_genes, aes(x = logFC, y = -log10(P.Value)), color = "darksalmon",
             size = 2) +
  geom_text(data = top_genes, aes(x = logFC, y = -log10(P.Value), label = gene_name),
            hjust = 0, vjust = 0, size = 4, color = "cadetblue") +
  theme_minimal(base_size = 14) +
  xlab("Log Fold Change") +
  ylab("-Log10(P-value)") +
  ggtitle("Volcano plot (with gene annotations)") +
  geom_hline(yintercept = -log10(0.05), col = "firebrick", linetype = "dashed") +
  geom_vline(xintercept = c(-1, 1), col = "navy", linetype = "dotted")
```

3. Heatmap: visualizing the top 50 differentially expressed genes

The heatmap provides a comprehensive visualization of the normalized expression levels of the top 50 differentially expressed genes, clustered by both rows (genes) and columns (samples). This clustering highlights potential patterns of expression between the different groups and genders in the dataset.

```r
# Extract normalized values for the top 50 most significant genes
exprs_heatmap <- vGene$E[rank(de_results$P.Value) <= 50, ]

# Create a table with sample information
df <- as.data.frame(colData(rse_gene_SRP056604)[, c("sra_attribute.subject_group",
                                                    "sra_attribute.gender")])
colnames(df) <- c("SubjectGroup", "Gender")
```

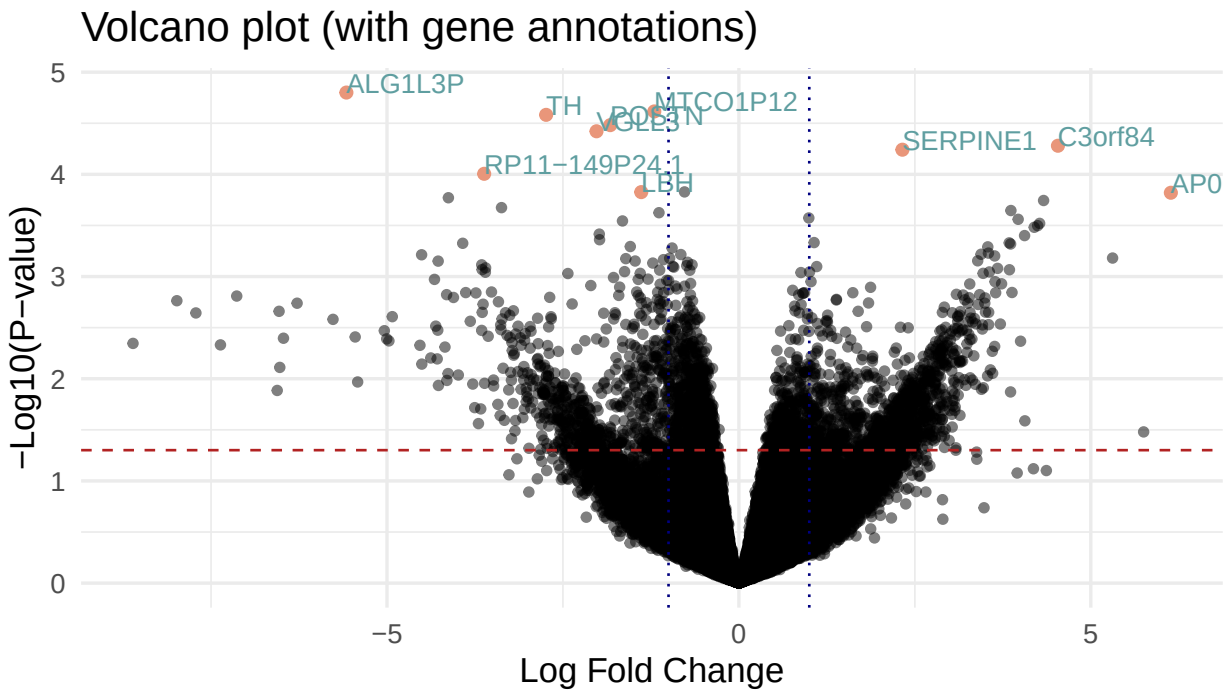

Figure 6: Volcano plot of gene expression changes

```r
# Create the heatmap
library("pheatmap")
pheatmap(
  exprs_heatmap,
  cluster_rows = TRUE,
  cluster_cols = TRUE,
  show_rownames = FALSE,  # Hide gene names
  show_colnames = FALSE,  # Hide sample names for clarity
  annotation_col = df,    # Add column annotations
  annotation_colors = list(
    SubjectGroup = c("age-matched control" = "cornsilk3",
                     "LOAD (late onset of Alzheimer's disease)" = "rosybrown"),
    Gender = c("male" = "paleturquoise3", "female" = "pink")
  ),
  color = colorRampPalette(c("navy", "white", "firebrick"))(50),  # Color scale
  fontsize = 8,  # Font size
  main = "Heatmap of the top 50 significant genes"
)
```

Gene	Description
SERPINE1	A serine protease inhibitor associated with fibrinolysis regulation, cell migration, and tissue repair.
ALG1L3P	A pseudogene of unknown function, related to asparagine-linked glycosylation processes.
RP11-149P24.1	A long intergenic non-coding RNA (lincRNA) with limited functional annotation.

Gene	Description
TH	Tyrosine hydroxylase, a rate-limiting enzyme in dopamine synthesis, implicated in catecholamine metabolism.
VGLL3	A transcriptional coactivator involved in regulating RNA polymerase II-mediated transcription.
POSTN	Periostin, an extracellular matrix protein involved in cell adhesion, tissue repair, and metastasis.
LBH	A regulator of WNT signaling pathways and a transcriptional activator involved in cellular signaling.
MTCO1P12	A mitochondrial pseudogene with limited functional information.
C3orf84	Now named CIMIP7, a ciliary microtubule inner protein, with emerging roles in cellular function.
AP000350.5	A lincRNA antisense to SLC2A11, with potential regulatory functions.

Table 1. Top 10 Differentially Expressed Genes. The table summarizes the top 10 genes identified as differentially expressed, including their known or predicted biological roles. (Retrieved from: GeneCards – the human gene database [www.genecards.org])

Biological interpretation

This findings support the hypothesis that Alzheimer's disease (AD) involves a complex interplay between neuronal and non-neuronal processes, as highlighted by the differential expression of genes related to beta-amyloid homeostasis, neurovascular defects, and transcriptional regulation. Among the top differentially expressed genes, SERPINE1 aligns with previous reports indicating its overexpression in AD, suggesting its role in modulating fibrinolysis and cell migration, which could impact neurovascular integrity ad beta-amyloid clearance. Additionally, the identification of lncRNAs such as RP11-149P24.1 and AP000350.5 emphasizes the importance of non-coding RNAs in the pathophysiology of AD, possibly affecting gene regulation at the transcriptional and post-transcriptional levels.

Furthermore, genes such as TH (tyrosine hydroxylase) and POSTN (periostin) point to disruptions in catecholamine biosynthesis and extracellular matrix remodeling, respectively, both of which are crucial for neuronal and vascular homeostasis. Interestingly, our data also highlight potential novel contributors like VGLL3, implicated in transcriptional regulation, and C3orf84, now known as CIMIP7, whose precise roles in AD remain to be elucidated.

While these results provide insights into potential molecular mechanisms underlying AD, we must acknowledge the need for further experimental validation to confirm these findings. Future studies could expand on these observations by integrating proteomic data, investigating cell-specific expression patterns, and exploring functional assays to establish causal relationships between these genes and AD progression.

Conclusions

This analysis highlights the potential of transcriptomic studies in advancing the understanding of late-onset Alzheimer's disease (LOAD). By identifying key differentially expressed genes, including lncRNAs, evidence is provided for molecular pathways and processes that may contribute to the disease. These findings, while promising, require further experimental validation to fully understand their biological significance and therapeutic potential.

This work underscores the importance of integrating bioinformatics with experimental research to unravel the complexity of neurodegenerative diseases and pave the way for future discoveries.

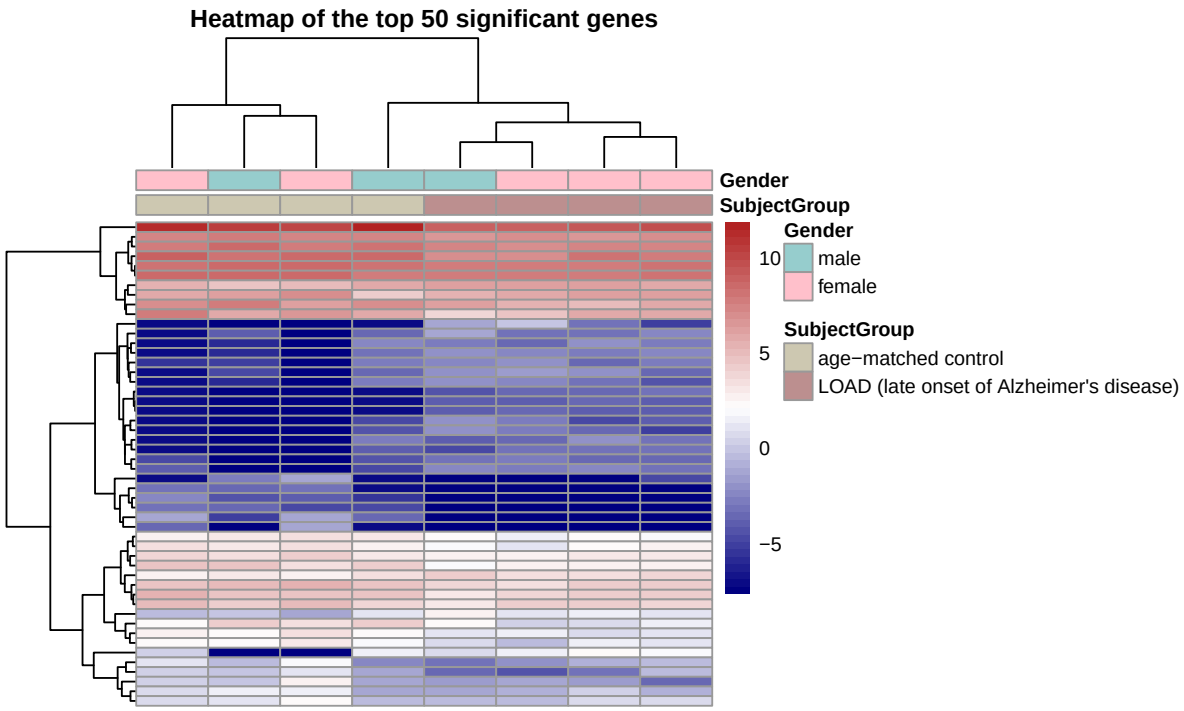

Figure 7: Heatmap of expression levels

Bibliography

Magistri, Marco, Dmitry Velmeshev, Madina Makhmutova, and Mohammad Ali Faghihi. 2015. "Transcriptomics Profiling of Alzheimer's Disease Reveal Neurovascular Defects, Altered Amyloid-Beta Homeostasis, and Deregulated Expression of Long Noncoding RNAs." *Journal of Alzheimer's Disease* 48 (3): 647–65. https://doi.org/10.3233/%7BJAD%7D-150398.