

R programming skills summary - Experimental Design and Data Management 2022

Sebastian Ayala-Ruano

2022-11-28T00:00:00+01:00

Table of contents

Preface	4
1 Week3 - Introduction to Gene expression analysis part 1	5
1.1 Assignment 1: Importing the data and inspecting sample information	5
1.1.1 Using the sampleInfo object, answer the following questions: What does DCM, HCM and PPCM stand for? (hint: Google)	6
1.1.2 How many co-variables are there? What do they mean? What type of value does each contain? (e.g. binary, continuous, categorical)	6
1.1.3 Are all variables measured in all individuals?	7
1.1.4 Using the sampleInfo object, create an overview of the sample sizes and characteristics in each disease category: How many individuals are healthy? How many suffer from DCM, HCM, PPCM?	7
1.1.5 What is the average age in each disease category?	8
1.1.6 How many male and female individuals are there in each disease category?	9
1.1.7 Are there other characteristics that are strikingly different between groups?	9
1.2 Assignment 2: Data exploration on the sample level	11
1.2.1 The gene expression dataset contains so-called log2-transformed CPM (counts per million) values: How are CPM values created from raw RNA-sequencing count data? And why is this needed? (HINT: Google “what the fpkm”)	11
1.2.2 Create and interpret 4 figures containing boxplots for all samples in the dataset, one for DCM, one figure for HCM, one for PPCM and one for the healthy controls.	11
1.2.3 Create and interpret 4 figures containing density plots for all samples in the dataset, one figure for DCM, one for HCM, one for PPCM and one for the healthy controls.	16
1.2.4 Assess the normality of all samples and summarize the results in a table or figure. (HINT: try ‘shapiro.test’)	18
1.2.5 Perform a principal component analysis (PCA), visualize the results and color by the disease category, sex and other variables of interest. Interpret the results: what patterns do you see?	19

2	Week4 - Introduction to Gene expression analysis part 2	22
2.1	Assignment 3: Data exploration on the gene level	23
2.1.1	Convert the CPM values to FPKM values. For some of the exercises below, we need to convert the CPM expression values to FPKM expression values.	23
2.1.2	What does FPKM stand for? How does this measure differ from CPM? (Google)	23
2.1.3	In you own words, describe what the code above does.	24
2.1.4	Can we compare the FPKM value of gene A between two samples to state in which sample gene A is more highly expressed?	24
2.1.5	Can we compare the FPKM value of gene A to gene B in a single sample to state which gene is more highly expressed?	24
2.1.6	Can we compare the CPM values of gene A between two samples to determine in which sample gene A is more highly expressed?	25
2.1.7	Can we compare the CPM value of gene A to the value of gene B in a single sample to determine which gene is more highly expressed?	25
2.1.8	Using the FPKM values, answer the following questions:	25
2.1.9	Using the CPM values, answer the following questions:	28
2.2	Assignment 4: Differential gene expression analysis.	32
2.2.1	What is differential gene expression analysis (DGEA)? What are some of the most common packages in R for DGEA?	32
2.2.2	Implement the steps noted in the limma guide for the MAGNET dataset. Start with a DGEA between DCM patients and healthy controls.	33
2.2.3	Which co-variates should be taken along for correction? (confounding; see the “alcohol causes lung cancer” example from the lecture)	34
2.2.4	Copy the top 200 differentially expressed genes to for a quick GO enrichment analysis. Which processes are changed between DCM and controls? Do these processes make biological sense? (quick literature check!)	35

Preface

This is a summary of the R programming skills sessions of the Experimental Design and Data Management (2223-MSB1005) course 2022. The course is part of the [Master of Systems Biology](#) at the Maastrich University. The course is taught by [Prof. Michiel Adriaens](#) and [Prof. Aaron Isaacs](#).

1 Week3 - Introduction to Gene expression analysis part 1

Sebastian Ayala-Ruano
today

This R script serves as a scaffold for adding the code required to fulfill the assignments. It includes the assignments as well as a few hints.

Add the necessary code and type your answers in this document for your own record.

```
library(ggplot2)
library(biomaRt)
library(pcaMethods)
library(readr)
library(dplyr)
library(tibble)
library(reshape2)
library(conflicted)
library(tidyr)
library(purrr)
```

1.1 Assignment 1: Importing the data and inspecting sample information

After unzipping the required transcriptomics (gene expression) file and sample information file, import both files as two separate objects. First set the active working directory to the folder containing the files.

```
# Load data
gxData <- read_delim("Data/MAGNET_GeneExpressionData_CPM_19112020.txt", delim = "\t")
```

```

sampleInfo <- read_csv("Data/MAGNET_SampleData_18112022.csv")

# Add gene ID as the row names
gxData <- column_to_rownames(gxData, var = "EnsemblGeneID")

# Add sample name as the row names
sampleInfo <- column_to_rownames(sampleInfo, var = "sample_name")

```

1.1.1 Using the sampleInfo object, answer the following questions: What does DCM, HCM and PPCM stand for? (hint: Google)

- **DCM:** Dilated cardiomyopathy
- **HCM:** Hypertrophic cardiomyopathy
- **PPCM:** Postpartum cardiomyopathy

1.1.2 How many co-variables are there? What do they mean? What type of value does each contain? (e.g. binary, continuous, categorical)

```

# Number of covariates
n_co_variates = ncol(sampleInfo)

# Types of variables of covariates
str(sampleInfo)

```

```

'data.frame':  366 obs. of  19 variables:
 $ tissue_source: chr  "NF" "NF" "NF" "NF" ...
 $ etiology      : chr  "NF" "NF" "NF" "NF" ...
 $ gender       : chr  "Male" "Male" "Male" "Female" ...
 $ race         : chr  "AA" "AA" "AA" "AA" ...
 $ age          : num  18 26 17 59 59 50 15 53 62 16 ...
 $ weight       : num  70 85 68 66 87 87 70 91 56 67 ...
 $ height       : num  175 183 173 157 157 175 183 165 168 163 ...
 $ hw           : num  NA NA 400 380 NA 640 NA NA NA 256 ...
 $ lv_mass      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ afib         : chr  "No" "No" "No" "No" ...
 $ VTVF        : chr  "No" "No" "No" "No" ...
 $ Diabetes     : chr  "No" "No" "No" "No" ...
 $ Hypertension : chr  "No" "No" "No" "Yes" ...

```

```

$ LVEF      : num  0.37 NA 0.27 0.6 0.55 0.65 0.15 NA NA 0.51 ...
$ RIN       : num  8.4 9.1 7.8 9.4 8.8 8.6 7.8 10 7.8 6.4 ...
$ Library.Pool : chr  "Magnet_10" "Magnet_11" "Magnet_09" "Magnet_06" ...
$ disease_race : chr  "AA_NF" "AA_NF" "AA_NF" "AA_NF" ...
$ minexpr    : num  7.47 7.47 7.47 7.47 7.47 ...
$ TIN.median. : num  72.8 74.3 77.4 69.5 73.6 ...

```

The number of co-variates is **19**.

Covariates are variables known to affect disease susceptibility and are independent of tested genotypes at the population level. They are used to control for confounding factors in the analysis of the association between a disease and a genetic variant.

1.1.3 Are all variables measured in all individuals?

No, there are NA values in some columns, which means that these values were not measured.

```

na_values <- sampleInfo %>%
  dplyr::select(everything()) %>%
  summarise_all(list(~ sum(is.na(.))))

```

```
na_values
```

```

tissue_source etiology gender race age weight height hw lv_mass afib VTVF
1             0          0      0    0    0        0    1  7    207    5    2
Diabetes Hypertension LVEF RIN Library.Pool disease_race minexpr TIN.median.
1             2          1   80    6          0          0          0          0

```

1.1.4 Using the sampleInfo object, create an overview of the sample sizes and characteristics in each disease category: How many individuals are healthy? How many suffer from DCM, HCM, PPCM?

Method 1

```

etiology1 <- sampleInfo %>%
  dplyr::select(etiology) %>%
  table()
etiology1

```

```
etiology
  DCM  HCM   NF  PPCM
166   28 166    6
```

Method 2

```
etiology2 <- sampleInfo %>%
  count(etiology)

etiology2
```

```
  etiology    n
1      DCM 166
2      HCM  28
3       NF 166
4     PPCM   6
```

Method 3

```
etiology3 <- sampleInfo %>%
  group_by(etiology) %>%
  tally()

etiology3
```

```
# A tibble: 4 x 2
  etiology    n
  <chr>    <int>
1 DCM      166
2 HCM       28
3 NF      166
4 PPCM       6
```

1.1.5 What is the average age in each disease category?

```
avg_etiology_by_age <- sampleInfo %>%
  group_by(etiology) %>%
  summarise(
    n = n(),
```



```

    age = mean(age, na.rm = TRUE)
  )

avg_etiology_by_age

```

```

# A tibble: 4 x 3
  etiology     n  age
  <chr>   <int> <dbl>
1 DCM     166  52.1
2 HCM     28  48.7
3 NF     166  55.9
4 PPCM     6  34.7

```

1.1.6 How many male and female individuals are there in each disease category?

```

count_etiology_by_gender <- sampleInfo %>%
  group_by(etiology) %>%
  count(gender)

count_etiology_by_gender

```

```

# A tibble: 7 x 3
# Groups:   etiology [4]
  etiology gender     n
  <chr>   <chr> <int>
1 DCM    Female    66
2 DCM    Male    100
3 HCM    Female    11
4 HCM    Male     17
5 NF     Female    89
6 NF     Male     77
7 PPCM   Female     6

```

1.1.7 Are there other characteristics that are strikingly different between groups?

```

summary(sampleInfo)

```

tissue_source	etiology	gender	race
Length:366	Length:366	Length:366	Length:366
Class :character	Class :character	Class :character	Class :character
Mode :character	Mode :character	Mode :character	Mode :character

age	weight	height	hw
Min. :15.00	Min. : 27.00	Min. : 55.0	Min. :151.0
1st Qu.:48.00	1st Qu.: 66.00	1st Qu.:163.0	1st Qu.:371.5
Median :55.00	Median : 78.00	Median :170.0	Median :448.0
Mean :53.28	Mean : 81.19	Mean :168.7	Mean :468.8
3rd Qu.:62.00	3rd Qu.: 91.00	3rd Qu.:178.0	3rd Qu.:556.5
Max. :83.00	Max. :267.00	Max. :196.0	Max. :923.0
		NA's :1	NA's :7

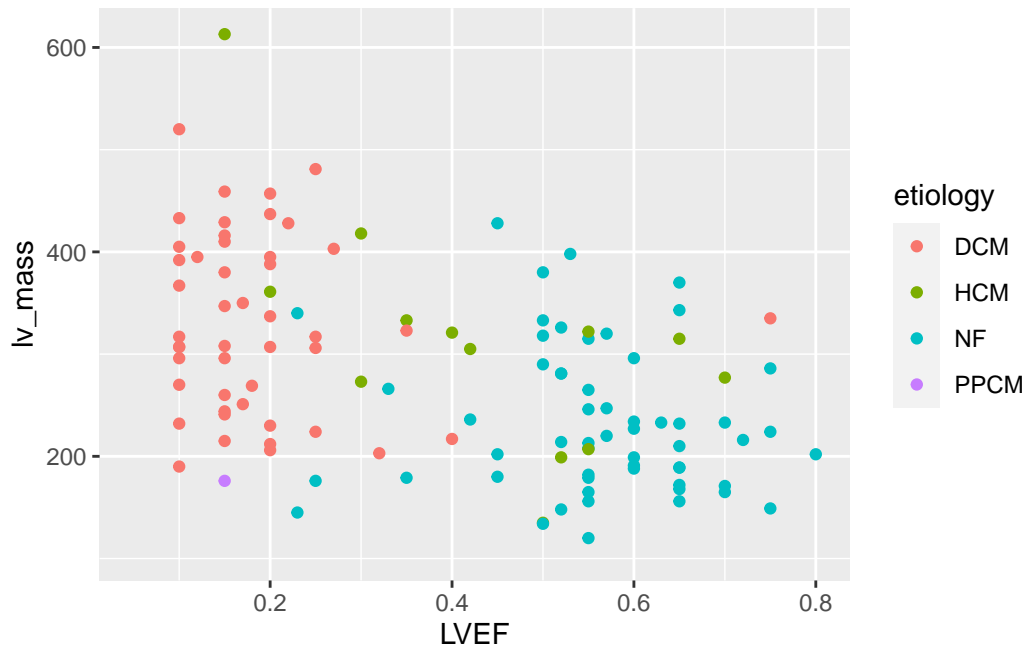
lv_mass	afib	VTVF	Diabetes
Min. :104.0	Length:366	Length:366	Length:366
1st Qu.:195.0	Class :character	Class :character	Class :character
Median :247.0	Mode :character	Mode :character	Mode :character
Mean :266.0			
3rd Qu.:321.5			
Max. :613.0			
NA's :207			

Hypertension	LVEF	RIN	Library.Pool
Length:366	Min. :0.0500	Min. : 5.800	Length:366
Class :character	1st Qu.:0.1500	1st Qu.: 8.100	Class :character
Mode :character	Median :0.2000	Median : 8.500	Mode :character
	Mean :0.3068	Mean : 8.451	
	3rd Qu.:0.5200	3rd Qu.: 8.900	
	Max. :0.8000	Max. :10.000	
	NA's :80	NA's :6	

disease_race	minexpr	TIN.median.
Length:366	Min. :7.469	Min. :23.38
Class :character	1st Qu.:7.469	1st Qu.:61.26
Mode :character	Median :7.469	Median :70.12
	Mean :7.469	Mean :65.03
	3rd Qu.:7.469	3rd Qu.:73.32
	Max. :7.469	Max. :81.52

An example of two features that are different between groups:

```
ggplot(sampleInfo, aes(x= LVEF, y = lv_mass, color = etiology)) +  
  geom_point()
```



1.2 Assignment 2: Data exploration on the sample level

1.2.1 The gene expression dataset contains so-called log2-transformed CPM (counts per million) values: How are CPM values created from raw RNA-sequencing count data? And why is this needed? (HINT: Google “what the fpkm”)

FPKM stands for fragments per kilobase of exon per million mapped fragments. It is used specifically in paired-end RNA-seq experiments. The interpretation of FPKM is that if you sequence your RNA sample again, you expect to see for gene i , $FPKM_i$ reads divided by gene i length over a thousand and divided by the total number of reads mapped over a million.

1.2.2 Create and interpret 4 figures containing boxplots for all samples in the dataset, one for DCM, one figure for HCM, one for PPCM and one for the healthy controls.

Create dataframes with information of the samples in each disease category.

```

# Create a vector of the names of etiologies
names_et <- etiology2$etiology

# Get the names of the samples per etiology
for (i in names_et){
  # Define the name of the dataframes with the list of samples per etiology
  var_name <- paste(i,"columns",sep = "_")
  # Get the dataframes with the list of sample names per etiology
  df <- rownames_to_column(sampleInfo, var = "sample") %>%
    dplyr::filter(etiology == i) %>%
    dplyr::select(sample) %>%
    pull(sample)
  # Assign the names to the dataframes
  assign(var_name, df)
}

```

1.2.2.1 Boxplot of NF patients

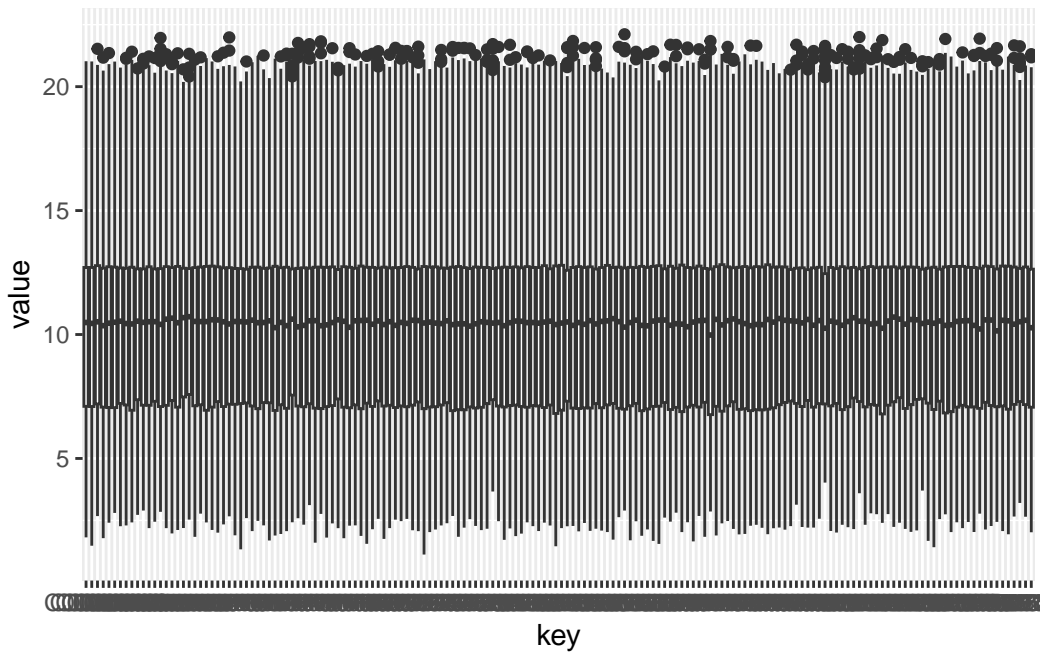
```

NF_data <- gxData %>%
  dplyr::select(NF_columns)

NF_data_group <- gather(NF_data) # melt also works, but it is deprecated

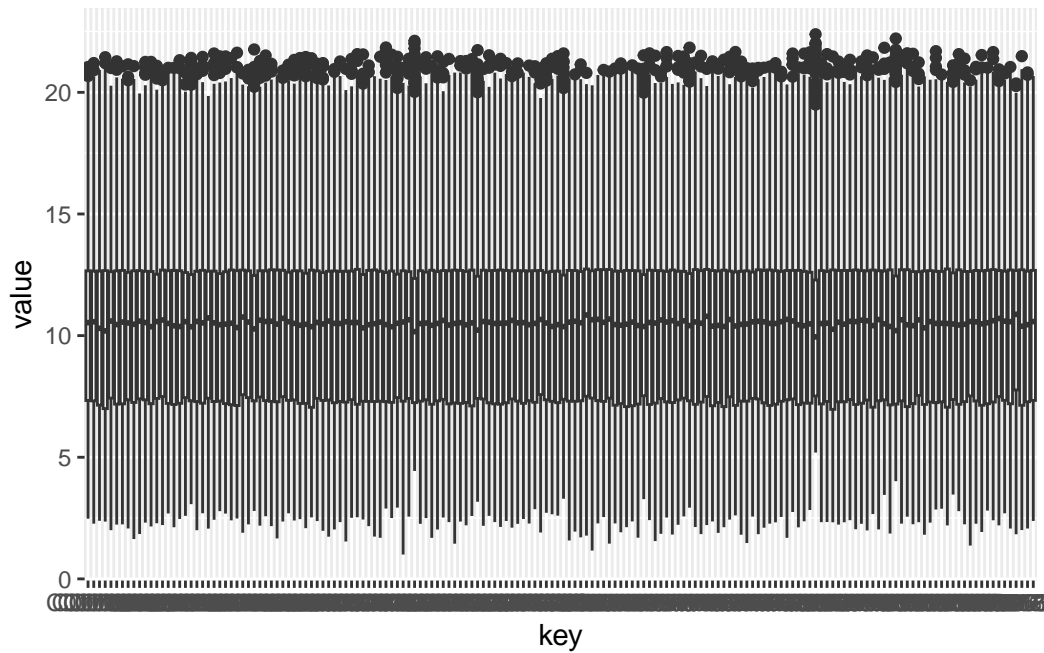
ggplot(NF_data_group, aes(x = key, y = value)) +
  geom_boxplot()

```



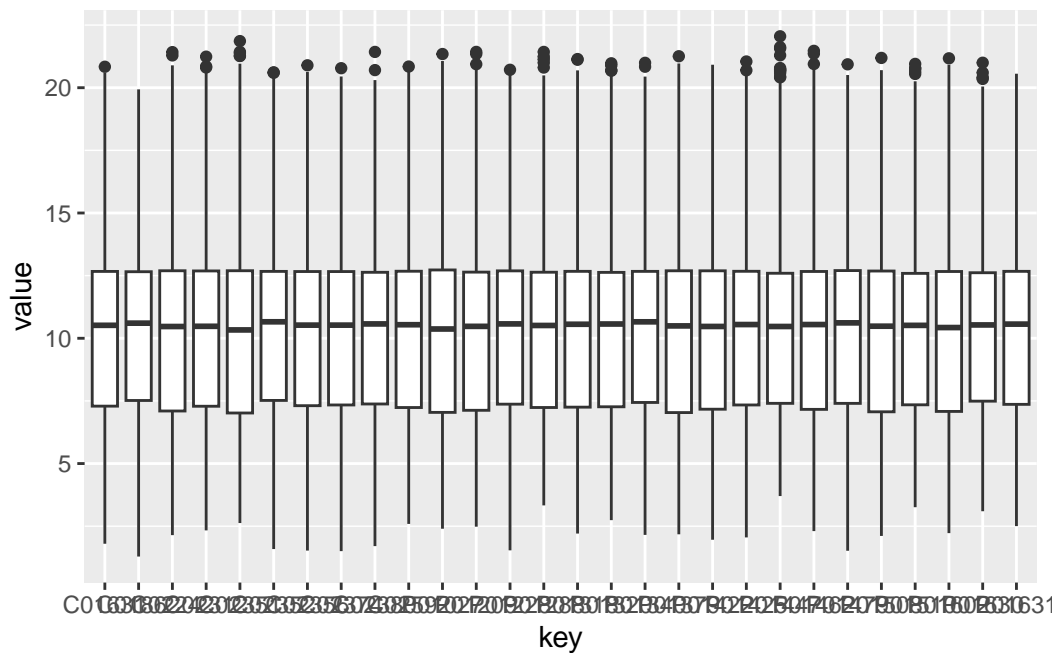
1.2.2.2 Boxplot of DCM patients

```
DCM_data <- gxDat %>%  
  dplyr::select(DCM_columns)  
  
DCM_data_group <- gather(DCM_data) # melt also works, but it is deprecated  
  
ggplot(DCM_data_group, aes(x = key, y = value)) +  
  geom_boxplot()
```



1.2.2.3 Boxplot of HCM patients

```
HCM_data <- gxDat %>%  
  dplyr::select(HCM_columns)  
  
HCM_data_group <- gather(HCM_data) # melt also works, but it is deprecated  
  
ggplot(HCM_data_group, aes(x = key, y = value)) +  
  geom_boxplot()
```

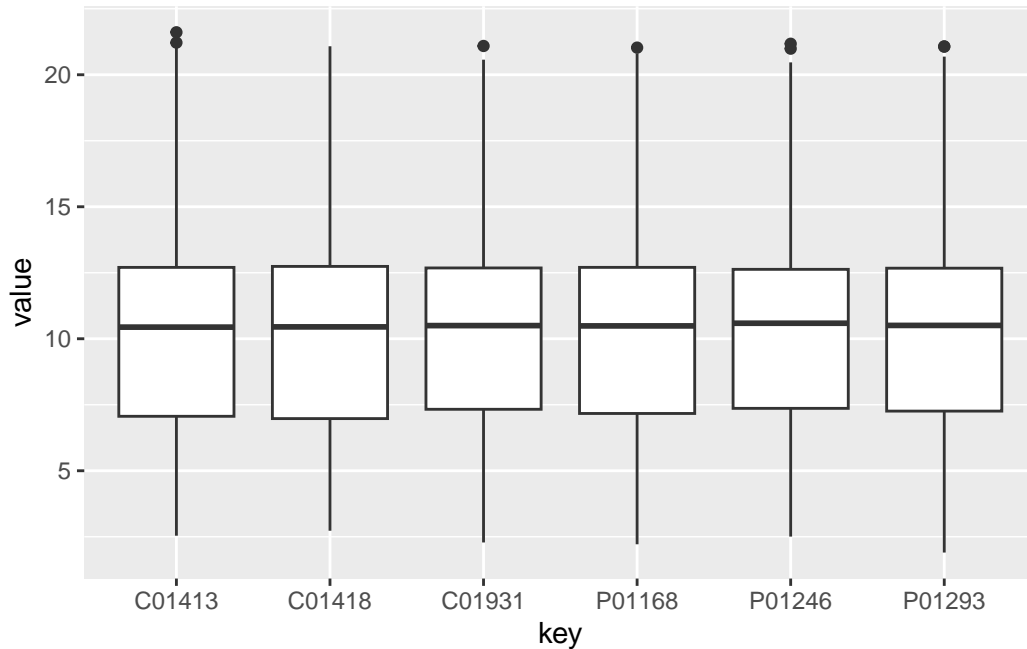


1.2.2.4 Boxplot of PPCM patients

```
PPCM_data <- gxData %>%
  dplyr::select(PPCM_columns)

PPCM_data_group <- gather(PPCM_data) # melt also works, but it is deprecated

ggplot(PPCM_data_group, aes(x = key, y = value)) +
  geom_boxplot()
```



1.2.3 Create and interpret 4 figures containing density plots for all samples in the dataset, one figure for DCM, one for HCM, one for PPCM and one for the healthy controls.

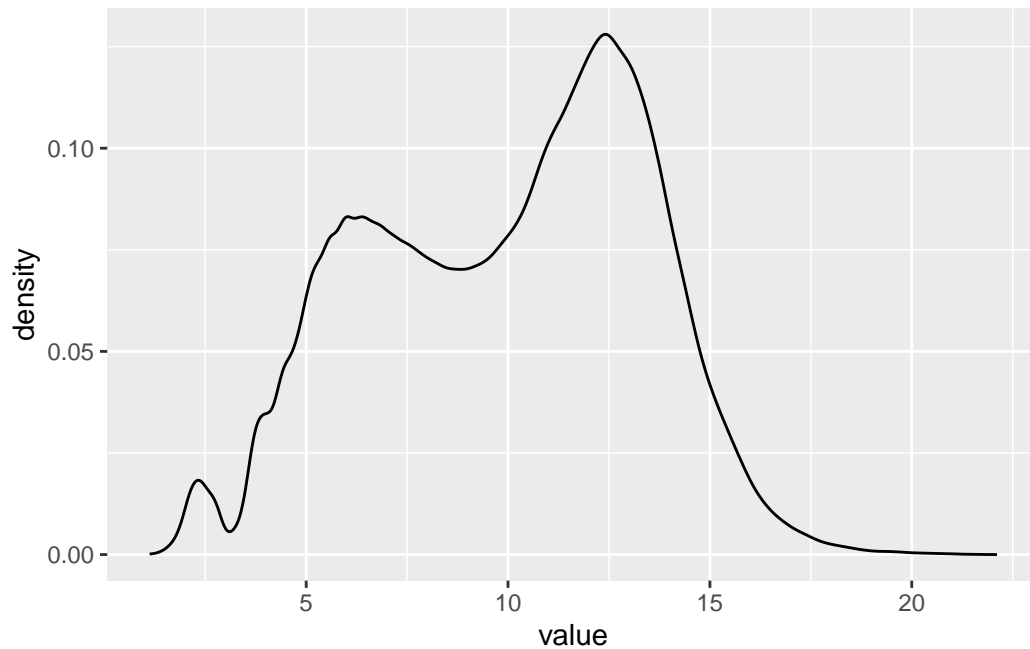
1.2.3.1 Density plot of all the values from the NF group colored by sample

```
ggplot(NF_data_group, aes(x = value, color = key)) +  
  geom_density(alpha = 0.2)
```


74		P01032		P01163		P01220		P01337		P01383		P01463	
185		P01067		P01167		P01233		P01339		P01387		P01465	
05		P01070		P01171		P01238		P01347		P01388		P01469	
20		P01076		P01173		P01241		P01348		P01396		P01471	
32		P01100		P01183		P01245		P01354		P01397		P01473	
58		P01101		P01184		P01248		P01356		P01400		P01481	
68		P01106		P01185		P01252		P01357		P01404		P01485	
86		P01116		P01197		P01259		P01359		P01406		P01488	
88		P01117		P01199		P01279		P01360		P01412		P01490	
90		P01119		P01202		P01287		P01362		P01413		P01492	
92		P01121		P01205		P01294		P01365		P01414		P01497	
93		P01122		P01206		P01296		P01367		P01420		P01498	
95		P01124		P01207		P01307		P01369		P01421		P01500	
96		P01134		P01208		P01310		P01374		P01436		P01503	
97		P01137		P01211		P01323		P01375		P01440		P01507	

1.2.3.2 Density plot of all the values from the NF group

```
ggplot(NF_data_group, aes(x = value)) +  
  geom_density()
```



The same code applies for the pther groups

1.2.4 Assess the normality of all samples and summarize the results in a table or figure. (HINT: try 'shapiro.test')

```
# Get random sample for all the samples
norm_df <- sample_n(gxData, 5000) %>%
  #slice(gxData, 5000:8000) %>%
  sapply(., shapiro.test) %>%
  as_tibble() %>%
  slice(2) %>%
  gather(sample, p_value) %>%
  mutate(normality = p_value < 0.05)

norm_df %>%
  count(normality)
```

```
# A tibble: 1 x 2
  normality     n
  <lgl>       <int>
1 TRUE       366
```

1.2.5 Perform a principal component analysis (PCA), visualize the results and color by the disease category, sex and other variables of interest. Interpret the results: what patterns do you see?

HINT: use the functions 'pca' and 'plotPcs' from the package 'pcaMethods'

1.2.5.1 Calculate the PCAs

```
# Transpose dataframe to merge with the metadata
t_gxData <- t(gxData)

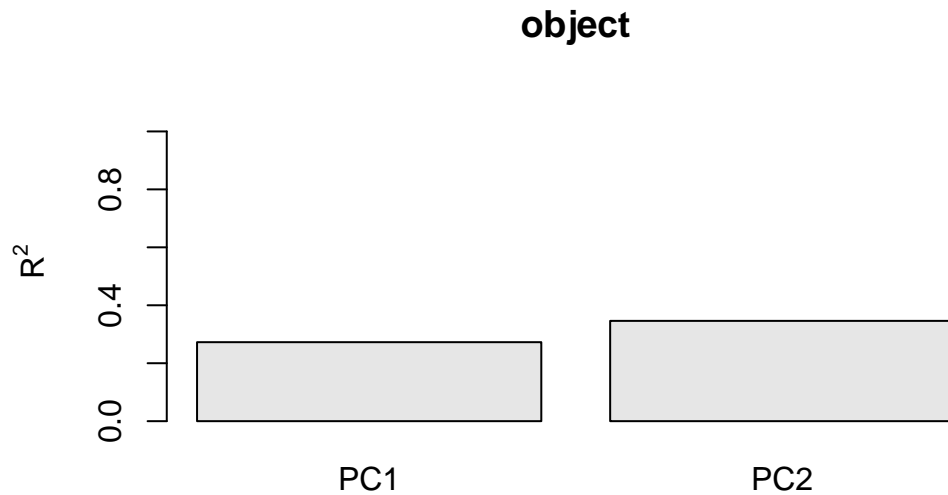
# Calculate pca
pca_hf <- pca(t_gxData, method = "svd")
```

1.2.5.2 Summary and barplot of the number of instances in the PCAs

```
# Get a short summary on the calculated model
summary(pca_hf)
```

```
svd calculated PCA
Importance of component(s):
              PC1    PC2
R2              0.2726 0.07351
Cumulative R2 0.2726 0.34612
```

```
plot(pca_hf)
```

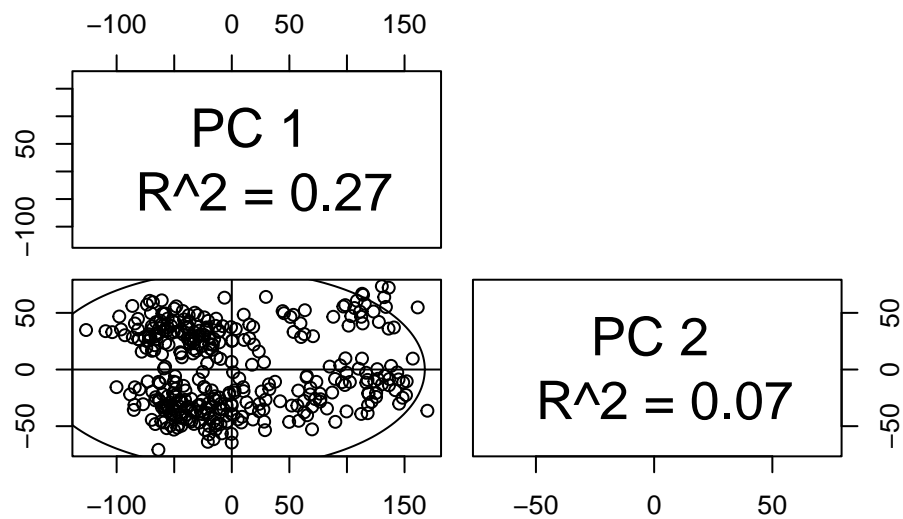


1.2.5.3 Merge pca scores with metadata for plotting

```
# Merge pca scores with metadata
df_hf <- merge(scores(pca_hf), sampleInfo, by = 0)
```

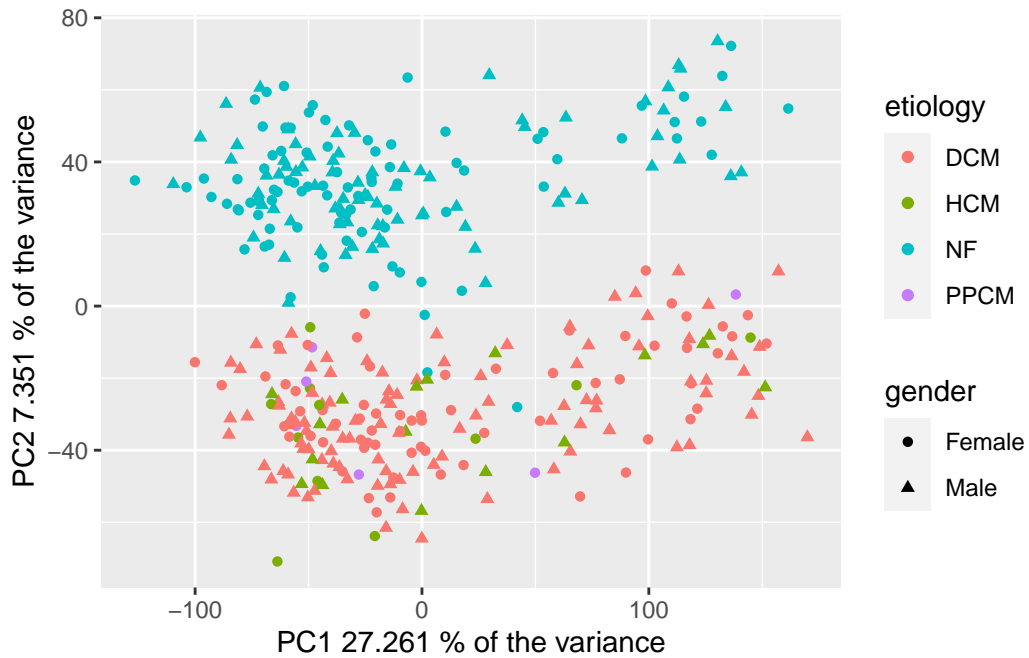
1.2.5.4 Plot the PCAs with the default them of plotsPcs

```
## Create scatterplot of the PCA with ggplot version 2
plotPcs(pca_hf)
```



1.2.5.5 Plot the PCAs with ggplot

```
## Create scatterplot of the PCA with ggplot version 1
ggplot(df_hf, aes(PC1, PC2, shape=gender, color=etiology)) +
  geom_point() +
  xlab(paste("PC1", pca_hf@R2[1] * 100, "% of the variance")) +
  ylab(paste("PC2", pca_hf@R2[2] * 100, "% of the variance"))
```



2 Week4 - Introduction to Gene expression analysis part 2

Sebastian Ayala Ruano
today

This R script serves as a scaffold for adding the code required to fulfill the assignments. It includes the assignments as well as a few hints.

Add the necessary code and type your answers in this document for your own record.

```
library(ggplot2)
library(readr)
library(dplyr)
library(tibble)
library(reshape2)
library(conflicted)
library(tidyr)
library(purrr)
library(limma)
library(edgeR)
library(corrmorant)
```

2.1 Assignment 3: Data exploration on the gene level

2.1.1 Convert the CPM values to FPKM values. For some of the exercises below, we need to convert the CPM expression values to FPKM expression values.

```
# Load data
geneTotExonLengths <- read_delim("Data/MAGNET_exonLengths.txt", delim = "\t")
gxData <- read_delim("Data/MAGNET_GeneExpressionData_CPM_19112020.txt", delim = "\t")
sampleInfo <- read_csv("Data/MAGNET_SampleData_18112022.csv")

# Add gene ID as the row names
gxData <- column_to_rownames(gxData, var = "EnsemblGeneID")
geneTotExonLengths <- column_to_rownames(geneTotExonLengths, var = "EnsemblGeneID")

# Check that row names are the same
all(rownames(geneTotExonLengths) == rownames(gxData)) # TRUE (just a check)
```

```
[1] TRUE
```

```
# Add sample name as the row names
sampleInfo <- column_to_rownames(sampleInfo, var = "sample_name")

# Convert CPM expression values to FPKM
cpm2fpkm <- function(x) {
  t <- 2^(x) * 1E3 / geneTotExonLengths[, 1] # . before variable makes it a hidden variable
}
gxData_fpkm <- cpm2fpkm(gxData)
```

2.1.2 What does FPKM stand for? How does this measure differ from CPM? (Google)

These metrics attempt to normalize for sequencing depth and gene length. Normalized expression units are necessary to remove technical biases in sequenced data such as depth of sequencing and gene length, and make gene expressions directly comparable within and across samples. More sequencing depth produces more read count for a gene expressed at the same level and differences in gene length generate unequal reads count for genes expressed at the same level.

CPM is a basic gene expression unit that normalizes only for sequencing depth (depth-normalized counts). It is biased in some applications where the gene length influences gene expression, such as RNA-seq.

$$CPM = \frac{N \text{ reads mapped to gene} \times 10^6}{Total \text{ } N \text{ of mapped reads}} \quad (2.1)$$

RPKM (reads per kilobase of transcript per million reads mapped) is a gene expression unit that measures the expression levels (mRNA abundance) of genes or transcripts. RPKM is a gene length normalized expression unit that is used for identifying the differentially expressed genes by comparing the RPKM values between different experimental conditions. Generally, the higher the RPKM of a gene, the higher the expression of that gene.

$$RPKM = \frac{N \text{ reads mapped to gene} \times 10^3 \times 10^6}{Total \text{ } N \text{ of mapped reads} \times \text{gene length in bp}} \quad (2.2)$$

Here, 10^3 normalizes for gene length and 10^6 for sequencing depth factor.

FPKM (fragments per kilobase of exon per million mapped fragments) is a gene expression unit which is analogous to RPKM. FPKM is used especially for normalizing counts for paired-end RNA-seq data in which two (left and right) reads are sequenced from the same DNA fragment. Generally, the higher the FPKM of a gene, the higher the expression of that gene.

When we map paired-end data, both reads or only one read with high quality from a fragment can map to reference sequence. To avoid confusion or multiple counting, the fragments to which both or single read mapped are counted and represented for FPKM calculation.

2.1.3 In your own words, describe what the code above does.

The code takes the gene expression values in CPM and normalizes them by the length of the genes, according to the formula expressed before.

2.1.4 Can we compare the FPKM value of gene A between two samples to state in which sample gene A is more highly expressed?

No, because FPKM values are normalized by the length of genes, which means that we cannot compare the values across different samples.

2.1.5 Can we compare the FPKM value of gene A to gene B in a single sample to state which gene is more highly expressed?

Yes, because FPKM values are normalized by the length of the genes.

2.1.6 Can we compare the CPM values of gene A between two samples to determine in which sample gene A is more highly expressed?

Yes, because CPM values are normalized by the sequencing depth (depth-normalized counts) and they do not take into account the length of genes for the normalization process.

2.1.7 Can we compare the CPM value of gene A to the value of gene B in a single sample to determine which gene is more highly expressed?

No, because CPM values are not normalized by the length of the genes.

2.1.8 Using the FPKM values, answer the following questions:

2.1.8.1 What are the IDs of the 5 highest expressed genes? What is their function according to the GeneCards website?

```
# Create a column with the mean of expression values of all samples
exp_mean_df <- gxData_fpkm %>%
  rownames_to_column(var = "geneID") %>%
  rowwise() %>%
  mutate(exp_mean = mean(c_across(C00039:P01640))) %>%
  column_to_rownames(var = "geneID") %>%
  select(exp_mean)

# Select the 5 most expressed genes
max5_genes_mean <- exp_mean_df %>%
  slice_max(n= 5, exp_mean)
```

Gene ID	Name	Function
ENSG00000198804	MT-CO1	Contributes to cytochrome-c oxidase activity
ENSG00000198899	MT-ATP6	Contributes to proton-transporting ATP synthase activity
ENSG00000198938	MT-CO3	Involved in respiratory chain complex IV assembly
ENSG00000198712	MT-CO2	Contributes to cytochrome-c oxidase activity

Gene ID	Name	Function
ENSG00000198886	MT-ND4	Enables NADH dehydrogenase (ubiquinone) activity

All of the genes are involved in processes related to mitochondria activity, which make sense because the dataset has muscle samples.

2.1.8.2 What are the IDs of the 5 lowest expressed genes? What is their function according to the GeneCards website?

```
# Select the 5 lowest expressed genes
min5_genes_mean <- exp_mean_df %>%
  slice_min(n = 5, exp_mean)
```

Gene ID	Name	Function
ENSG00000015568	RGPD5	RAN is a small GTP-binding protein of the RAS superfamily that is associated with the nuclear membrane
ENSG00000162105	SHANK2	This gene encodes a protein that is a member of the Shank family of synaptic proteins that may function as molecular scaffolds in the postsynaptic density of excitatory synapses
ENSG00000267586	LINC00907	RNA Gene, and is affiliated with the lncRNA class
ENSG00000215126	ZNG1F	Predicted to enable ATP binding activity
ENSG00000183914	DNAH2	Dyneins are microtubule-associated motor protein complexes

There are pseudogenes, RNA genes, and others related to different processes (i.e. synaptic genes).

2.1.8.3 What are the IDs of the 5 most variable genes? What is their function according to the GeneCards website?

```
# Create a column with the mean of expression values of all samples
exp_var_df <- gxData %>%
  rownames_to_column(var = "geneID") %>%
  rowwise() %>%
  mutate(exp_var = var(c_across(C00039:P01640))) %>%
  column_to_rownames(var = "geneID") %>%
  select(exp_var)

# Select the 5 most variable genes
max5_genes_var <- exp_var_df %>%
  slice_max(n= 5, exp_var)
```

Gene ID	Name	Function
ENSG00000198692	EIF1AY	Eukaryotic Translation Initiation Factor 1A Y-Linked
ENSG00000129824	RPS4Y1	Ribosomal Protein S4 Y-Linked 1
ENSG00000114374	USP9Y	Ubiquitin Specific Peptidase 9 Y-Linked
ENSG00000067048	DDX3Y	DEAD-Box Helicase 3 Y-Linked
ENSG00000012817	KDM5D	Lysine Demethylase 5D - encodes a protein containing zinc finger domains

4 of these genes are related to Y chromosome, which are absent in the female samples.

Note: By using the FPKM dataset, we got the same genes as the 5 most highly expressed.

2.1.8.3.1 What are the IDs of the 5 least variable (= stable!) genes? What is their function according to the GeneCards website?

```
# Select the 5 least variable genes
min5_genes_var <- exp_var_df %>%
  slice_min(n= 5, exp_var)
```

Gene ID	Name	Function
ENSG00000136709	WD Repeat Domain 33	WD repeats are conserved regions, which may facilitate formation of heterotrimeric or multiprotein complexes
ENSG00000089053	ANAPC5	Anaphase Promoting Complex Subunit 5
ENSG00000111361	EIF2B1	Eukaryotic Translation Initiation Factor 2B Subunit Alpha
ENSG00000086475	SEPHS1	Selenophosphate Synthetase 1
ENSG00000106609	TMEM248	Transmembrane Protein 248

All of the genes are related to conserved cellular functions - house keeping genes.

2.1.9 Using the CPM values, answer the following questions:

2.1.9.1 Which 5 genes show the strongest correlation to age in the control group?

```
# Get the dataframe with the list of sample names with NF
NF_columns <- rownames_to_column(sampleInfo, var = "sample") %>%
  dplyr::filter(etiology == "NF") %>%
  dplyr::select(sample) %>%
  pull(sample)

# Get the gene expression data from the NF patients
NF_data <- gxDat %>%
  dplyr::select(NF_columns)

# Transpose the dataframe to have genes as columns
NF_data <- as.data.frame(t(NF_data))

# Get the metadata from NF patients
t_sampleInfo <- as.data.frame(t(sampleInfo))

NF_metadata <- t_sampleInfo %>%
  rownames_to_column(var = "covariate") %>%
  dplyr::select(c(NF_columns, covariate)) %>%
  column_to_rownames(var = "covariate")
```

```

# Transpose the dataframe to have age as column
NF_metadata <- as.data.frame(t(NF_metadata))

# Add age column into the gene expression dataframe
NF_data <- NF_data %>%
  mutate(age = NF_metadata$age)

# Calculate correlation values
data_cor <- cor(NF_data[, colnames(NF_data) != "age"],
  as.numeric(NF_data$age))

data_cor <- as.data.frame(data_cor)

# Select the 5 most correlated genes with age in the control group
corr5_genes_age <- data_cor %>%
  arrange(desc(abs(V1))) %>%
  slice_head(n= 5)

# Get the gene expression data
corr5_gene_data <- NF_data %>%
  dplyr::select(row.names(corr5_genes_age))

# Calculate significance (p value) of the 5 most correlated genes with age
corr5_genes_age[1,2] <- cor.test(corr5_gene_data$ENSG00000244681, as.numeric(NF_data$age))
corr5_genes_age[2,2] <- cor.test(corr5_gene_data$ENSG00000244694, as.numeric(NF_data$age))
corr5_genes_age[3,2] <- cor.test(corr5_gene_data$ENSG00000182264, as.numeric(NF_data$age))
corr5_genes_age[4,2] <- cor.test(corr5_gene_data$ENSG00000154080, as.numeric(NF_data$age))
corr5_genes_age[5,2] <- cor.test(corr5_gene_data$ENSG00000250337, as.numeric(NF_data$age))

colnames(corr5_genes_age) <- c("estimate", "p_value")

```

- Is the correlation positive or negative?

4 of the values are positive and one is negative

- Is the correlation significant?

Yes, all of the correlations were significant

2.1.9.2 What is their function according to the GeneCards website? Are they genes of which the expression is known to change with age (use Pubmed)?

Gene ID	Name	Function
ENSG00000244681	MTHFD2P1	Pseudogene
ENSG00000244694	PTCHD4	Predicted to be integral component of membrane
ENSG00000182264	IZUMO1	The sperm-specific protein Izumo is essential for sperm-egg plasma membrane binding and fusion
ENSG00000154080	CHST9	Catalyzes the transfer of sulfate to position 4 of non-reducing N-acetylgalactosamine (GalNAc) residues in both N-glycans and O-glycans
ENSG00000250337	PURPL	RNA Gene, and is affiliated with the lncRNA class. Diseases associated with colorectal cancer and myasthenic syndrome

2.1.9.3 Visualize the result for at least 1 gene (HINT: CPM values on the y-axis, age in years on the x-axis)

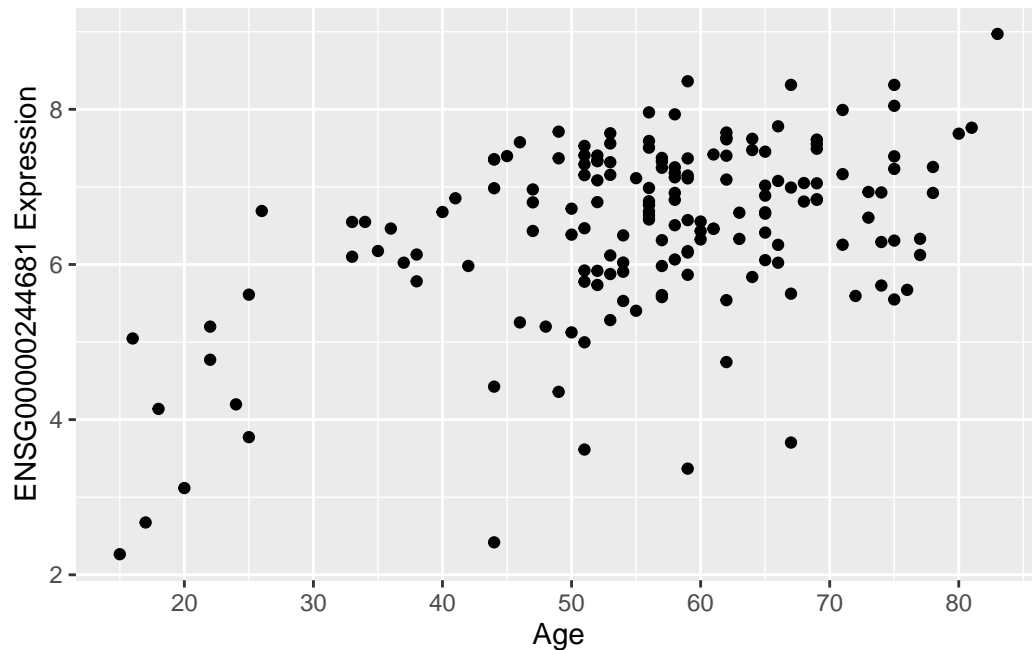
Scatter plot of one gene vs age

```
# Add age column into the 5 most correlated genes dataframe
corr5_gene_data <- corr5_gene_data %>%
  mutate(Age = as.numeric(NF_metadata$age), .before = 1)

corr1gene_age <- corr5_gene_data %>%
  select(Age, ENSG00000244681)

corr1gene_age_plot <- ggplot(corr1gene_age, aes(x = Age, y = ENSG00000244681)) +
  geom_point() +
  labs(x = "Age", y = "ENSG00000244681 Expression") +
  scale_x_continuous(n.breaks = 10.0)
```

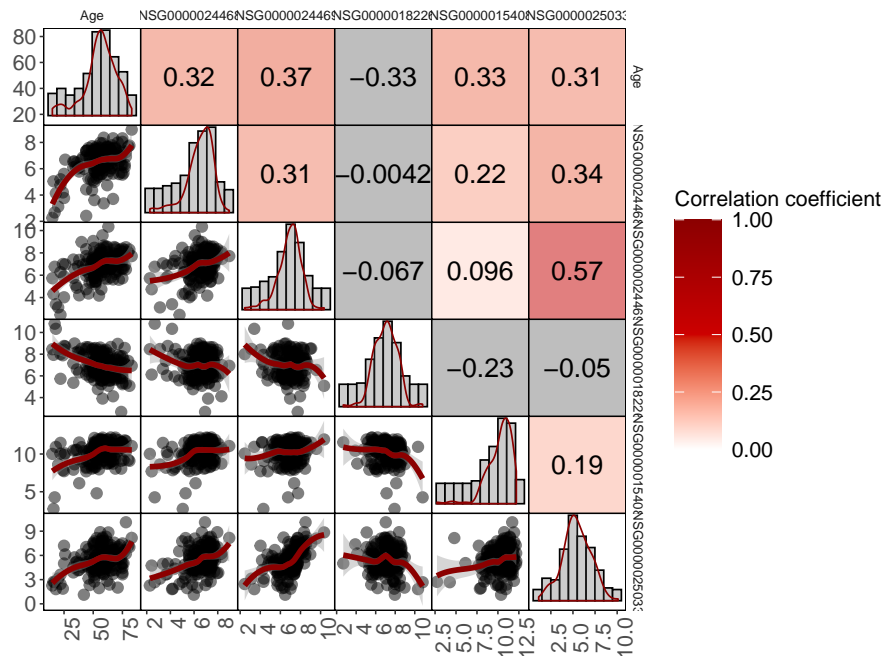
```
corr1gene_age_plot
```



Scatter plots, correlation values, and distributions of all genes and age

```
corr_plot_allgenes <- ggcorrmm(data = corr5_gene_data) +
  theme_corrmm(base_size = 6) +
  theme(axis.text.x = element_text(angle = 90, size = 8),
        axis.text.y = element_text(size = 8),
        strip.text.x = element_text(size = 5),
        strip.text.y = element_text(size = 5),
        legend.text = element_text(size = 8),
        legend.title = element_text(size = 8)) +
  lotri(geom_point(alpha = 0.5)) +
  lotri(geom_smooth(colour = "red4")) +
  utri_heatmap(alpha = 0.5, corr_method = "spearman") +
  utri_corrtext(corr_method = "spearman", size = 3.5) +
  dia_histogram(lower = 0.1, fill = "grey80", color = 1) +
  dia_density(lower = 0.1, alpha = .1, colour = "red4") +
  scale_fill_gradient2(low = "white", mid = "red3", high = "red4",
                      midpoint = 0.5, space = "rgb",
                      guide = guide_colorbar(title = "Correlation coefficient"),
                      limits = c(0, 1))
```

corr_plot_allgenes



2.2 Assignment 4: Differential gene expression analysis.

Now that we have explored the gene expression data, it is time to perform a differential gene expression analysis.

2.2.1 What is differential gene expression analysis (DGEA)? What are some of the most common packages in R for DGEA?

Differential expression analysis means taking the normalised read count data and performing statistical analysis to discover quantitative changes in expression levels between experimental groups.

R packages:

- [DESeq2](#)
- [limma](#)

We are going to use the limma package to perform a DGEA. We need to use the CPM normalized values. Have a look at the limma guide section 15.4: (<https://www.bioconductor.org/packages/devel/bioc/v>

2.2.2 Implement the steps noted in the limma guide for the MAGNET dataset. Start with a DGEA between DCM patients and healthy controls.

2.2.2.1 Limma-trend

```
# Convert counts to logCPM values
logCPM <- cpm(gxData, log = TRUE, prior.count = 3)

# Create design matrix
design = model.matrix(~0 + sampleInfo$etiology)

# Apply limma pipeline
fit <- lmFit(logCPM, design)
fit <- eBayes(fit, trend = TRUE)
topTable(fit, coef = ncol(design))
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
ENSG00000089053	6.381604	6.384943	961.3848	0	0	1415.207
ENSG00000129351	6.429434	6.416465	955.8661	0	0	1413.270
ENSG00000105323	6.380228	6.365844	944.4249	0	0	1409.213
ENSG00000136709	6.329745	6.324768	937.9541	0	0	1406.892
ENSG00000106609	6.348231	6.340574	921.1237	0	0	1400.769
ENSG00000176915	6.281051	6.282988	907.5412	0	0	1395.732
ENSG00000075785	6.481443	6.478184	898.0510	0	0	1392.161
ENSG00000100711	6.284988	6.278731	891.0022	0	0	1389.480
ENSG00000182944	6.459927	6.446393	890.8330	0	0	1389.416
ENSG00000113648	6.404155	6.400461	890.3165	0	0	1389.218

```
# Give more weight to fold-changes in the gene ranking
fit <- lmFit(logCPM, design)
fit <- treat(fit, lfc = log2(1.2), trend = TRUE)
topTreat(fit, coef = ncol(design))
```

	logFC	AveExpr	t	P.Value	adj.P.Val
ENSG000000000003	6.117085	6.111910	388.5360	0	0
ENSG000000000419	6.294569	6.292874	480.5645	0	0
ENSG000000000457	6.100006	6.111824	433.3020	0	0
ENSG000000000460	5.886348	5.887347	308.1765	0	0
ENSG000000000938	5.992674	6.059126	206.5199	0	0

ENSG00000000971	6.391421	6.392179	255.8019	0	0
ENSG00000001036	6.299640	6.308049	581.1499	0	0
ENSG00000001084	6.273295	6.255830	284.7585	0	0
ENSG00000001167	6.134147	6.122626	201.0493	0	0
ENSG00000001460	5.943226	5.954836	370.8629	0	0

Results show the p value of 0 in some genes, which means that something is wrong.

2.2.3 Which co-variates should be taken along for correction? (confounding; see the “alcohol causes lung cancer” example from the lecture)

```
# Convert counts to logCPM values
logCPM <- cpm(gxData, log = TRUE, prior.count = 3)

# Create design matrix considering confounding variables
design = model.matrix(~0 + etiology + gender + age, data = sampleInfo)

# Apply limma pipeline with confounding variables
fit <- lmFit(logCPM, design)

cont.matrix <- makeContrasts(DCMvsControl = etiologyDCM - etiologyNF,
                             levels = design)

fit <- contrasts.fit(fit, cont.matrix)

efit <- eBayes(fit, trend = TRUE)

dgeRes <- topTable(efit, coef = 'DCMvsControl', number = nrow(gxData))

glimpse(dgeRes)
```

Rows: 20,781

Columns: 6

```
$ logFC      <dbl> 0.15831103, 0.33966098, 0.21263013, -0.44141913, 0.36628640, ~
$ AveExpr    <dbl> 6.286287, 5.952977, 6.101884, 6.022681, 6.049712, 6.371697, ~
$ t          <dbl> 29.01545, 27.11832, 25.18996, -25.06301, 24.77563, -24.26241~
$ P.Value     <dbl> 7.851116e-97, 1.689807e-89, 7.264246e-82, 2.342191e-81, 3.33~
$ adj.P.Val   <dbl> 1.631540e-92, 1.755794e-85, 5.031943e-78, 1.216827e-77, 1.38~
$ B          <dbl> 210.1073, 193.3613, 175.9173, 174.7550, 172.1180, 167.3887, ~
```

After including the cofounding variables, the p values are not 0, which means that the cofounding variables are important for the analysis.

2.2.4 Copy the top 200 differentially expressed genes to for a quick **GO enrichment analysis**. Which processes are changed between DCM and controls? Do these processes make biological sense? (quick literature check!)

```
# Select the 200 most correlated genes with age in the control group
to200_corr_genes <- dgeRes %>%
  slice_head(n = 200) %>%
  rownames_to_column(var = "Gene_ID") %>%
  select(Gene_ID)

# Select the names of all genes in the control group
all_corr_genes <- dgeRes %>%
  rownames_to_column(var = "Gene_ID") %>%
  select(Gene_ID)

# Export target list to csv file
write_csv(to200_corr_genes, "Outputs/to200_corr_genes.csv", col_names = FALSE)

# Export background list to csv file
write_csv(all_corr_genes, "Outputs/all_corr_genes.csv", col_names = FALSE)
```

In the [GORilla server](#), the inputs are the target and background tables exported in the previous step. The results are shown in the following figure:

The results show that the most enriched GO terms are related to inflammation and structural processed, and the immune system, which is consistent with the literature.

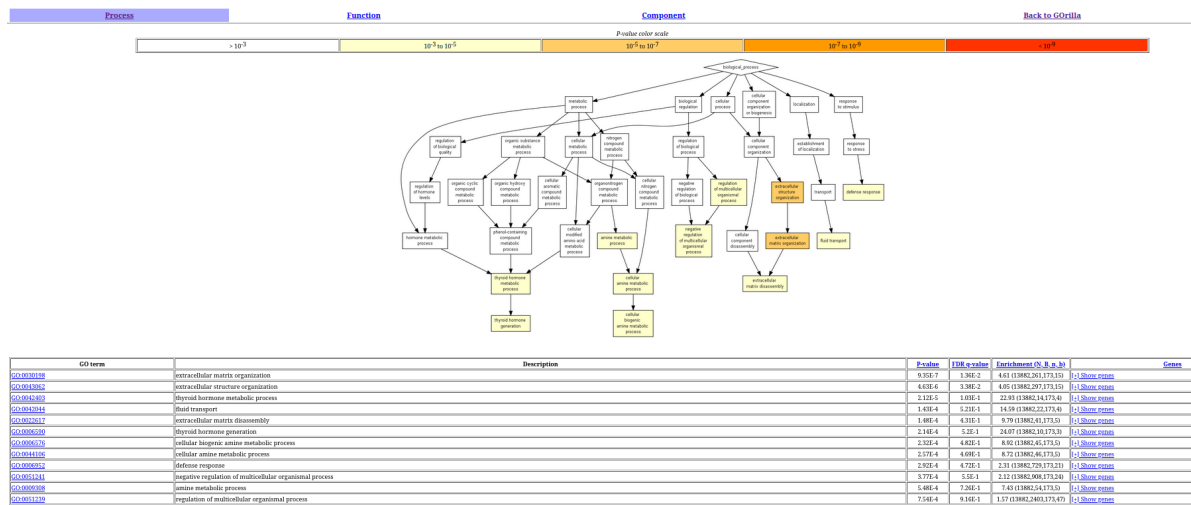


Figure 2.1: GOrilla results