Comparative genomics using R

Stefano Berto, PhD

2020-03-01

Contents

Introduction **5**

What will you get out of this? 5

Structure of the course 5

Packages needed to run the course code 6

1 Introduction to RNA-seq pipes **9**

1.1 Sequencing Machines 9

1.2 FASTQ 9

1.3 Quality Check 9

1.4 Quality Trimming 10

1.5 Alignment 10

1.6 Statistics and Filtering 11

1.7 Lift over 12

1.8 Quantification 13

2 Data Exploration **15**

3 Differential Expression Analysis **37**

3.1 Data loading 37

3.2 DGE based on linear model 39

3.3 Species Specific DGE 41

3.4 Surrogate Variables 43

3.5 Balance the gene expression for covariates 47

3.6 DGE visualizations 47

3.7 Functional Enrichment 53
3.8 DGE based on DESeq2 54
3.9 Exercise for Differential Expression Chapter 56

Introduction

The aim of this course is to provide the fundamentals for data analysis for comparative genomics. This course is a starting point for computational genomics students interested on comparative genomics and a guide for further data analysis in more specific topics in genomics.

What will you get out of this?

This resource describes the skills and provides how-tos that will help readers analyze their own comparative genomic data.

Working together:

- You will get exposed to current tools for transcriptomics.
- You will advance with the basics of R and dive right in to specialized uses of R for comparative genomics.
- You will apply simple data processing and analysis approached on the data
- You will be able to use R and the library-verse to do some visualizations and in deep analysis.
- You will (hopefully :-)) develop a critical mindset on data analysis, gaps, weakness and how to solve some problems.
- You will develop ideas, questions, hypothesis to how solve the big puzzle called brain

Structure of the course

The course is designed with insights into practical data analysis for comparative genomics. The course will focus on methods, data visualizations, and some

biostats that can be applied not only to comparative data but also to more diverse data. The course will always show the code and explain the code for a particular data analysis task. In addition, the course will provide also links and additional information such as websites, papers, websites for readers who desire to gain a bit more knowledge on the comparative genomics.

Here the chapters with some exercises:

- **"Introduction to RNA-seq pipes" chapter**

Basic concepts of high-throughput sequencing pipelines, tools, and how to apply these to comparative genomics. This will go through just from step A (fastq.gz) to step Z (count table). Importantly, how-tos for quality checks, processing, alignments of high-throughput sequencing.

- **"Data Exploration" chapter**

It provides basic R skills to explore, analyze, visualize data. The skills introduced in this chapter are important because cna be applied, with some modifications, to other type of data.

- **"Differential Expression" chapter**

It provides basic R skills to explore, analyze, visualize differential expression between species.

Packages needed to run the course code

This course is primarily about using R packages. Therefore if you want to reproduce the analysis in this course you need to install the relevant packages in each chapter using `install.packages` or `BiocManager::install` functions.

So here we go!!

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install(c('qvalue','clusterProfiler','DESeq2',
                       'limma','sva','edgeR','doParallel',
                       'EnsDb.Hsapiens.v86','AnnotationDbi',
                       'GOstats')

install.packages("future.apply")
install.packages("biomaRt")
```

```r
install.packages("tidymodels")
install.packages("broom")
install.packages("here")
install.packages("pheatmap")
install.packages("VennDiagram")
install.packages("GGally")

install.packages("devtools")
devtools::install_github("kassambara/ggpubr")
devtools::install_github("tidyverse/tidyverse")
devtools::install_github("ycphs/openxlsx")
devtools::install_github("tidyverse/ggplot2")
devtools::install_github("hms-dbmi/UpSetR")
```

Chapter 1

Introduction to RNA-seq pipes

1.1 Sequencing Machines

Sequencing machines vary based on number of reads they sequence and running costs. Nowadays, the most used are NovaSeq and NextSeq 500, both illumina.

1.2 FASTQ

This is the files sequencing facilities usually provide. This output is the same for multiple data, from RNA-seq to ChIP-seq, ATAC-seq, and so on. The FASTQ file is a modified version of a FASTA file with some additional information.

1.3 Quality Check

This is a step to understand the quality of the million of reads the FASTQ contains. It can be done using fastqc (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/) or in R with `library(fastqcr)` or `library(Rqc)`

Here a simple R example:

```
BiocManager::install("Rqc")
library(Rqc)
# load the fastq.gz files in your specific directory
quality_checks <- rqc(path = "path_to_your_directory", pattern = ".fastq.gz",
```

```
  openBrowser = FALSE)

rqcCycleQualityBoxPlot(quality_checks)
rqcCycleBaseCallsLinePlot(quality_checks)
```

This will output similar plots and data quality check as fastqc.

1.4 Quality Trimming

After checking the quality you can decide to trim the reads or not. Quality trimming is necessary to remove potential portion of the reads and/or base pairs with low quality. Low quality bases or fragments can affect mappability to the genome. Trimming can be done using trimmomatic (http://www.usadellab.org/cms/?page=trimmomatic), fastx (http://hannonlab.cshl.edu/fastx_toolkit/), or Trim Galore (https://www.bioinformatics.babraham.ac.uk/projects/trim_galore/).

1.5 Alignment

After quality check and trimming, the reads can be aligned to the reference genome. The process map the reads into the genome using a specific set of coordinates for genes. Alignment algorithms tolerate mismatches between reads and genome increasing overall mappability quality. Reads can be multimapped (map to multiple places on the genome) or with overall low quality. Alignment algorithms provide a quality score (MAPQ) for each reads and this can be easily used for filtering bad reads and retain only reads with high quality score which is often linked to uniquely mapped reads.

There are different aligner for different type of NGS data.

Here some example for ChIP-seq/ATAC-seq:

- BWA (http://bio-bwa.sourceforge.net/)
- Bowtie2 (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml)

For RNA-seq data instead there are splice aware aligners. They require a gene annotation usually stored in a gtf file. GTF can be found in UCSC/Ensembl for different species. For model species such as human and mouse you can also find them in Genecode (https://www.gencodegenes.org/). These aligners can splice the reads that belongs to different portions of the genomes (e.g. two different exons separated by an intron):

- STAR (https://github.com/alexdobin/STAR)
- HISAT (https://ccb.jhu.edu/software/hisat2/index.shtml)

First you will need to create a genome index. Here an example for STAR:

```
STAR --runMode genomeGenerate \
--genomeDir Human_Genome_Directory/ \
--genomeFastaFiles Human_Genome_Directory/*.fa \
--runThreadN 13 \
--sjdbGTFfile gene_annotation.gtf \
--sjdbOverhang 75
```

Then you can start the alignment

```
for file in `ls *.Trim.fastq.gz`
do
outputname=`basename $file | sed -e "s/.Trim.fastq.gz//"`
STAR --runThreadN 14 \
--genomeDir Human_Genome_Directory/ \
--readFilesIn $file \
--readFilesCommand zcat \
--sjdbGTFfile Human_Genome_Directory/gene_annotation.gtf \
--outFilterType BySJout  \
--outFilterMismatchNoverReadLmax 0.04 \
--outFilterMultimapNmax 10 \
--alignSJoverhangMin 10 \
--alignSJDBoverhangMin 1 \
--outSAMtype BAM SortedByCoordinate \
--outSAMunmapped Within \
--outFilterMismatchNmax 3 \
--twopassMode Basic \
--outFileNamePrefix $outputname \
--chimSegmentMin 15 \
--chimScoreMin 15 \
--chimScoreSeparation 10 \
--chimJunctionOverhangMin 15 \
--quantMode TranscriptomeSAM
echo $outputname
done
```

1.6 Statistics and Filtering

Alignment will provide a SAM/BAM file that can be easily handle with samtools (https://github.com/samtools/). This format contains all the reads and the

statistics from the alignment (mapped, unmapped, MAPQ, ...) with specific FLAG id. Ideally you want to work with uniquely mapped reads (e.g. reads that mapped into a single region). Some aligners can provide directly a bam file with only uniquely mapped reads. On the other hand, some aligner does not provide a flag for uniquely mapped (NH:i:1). Therefore you need to use MAPQ and filter for a specific threshold. As I descibed above, MAPQ is a -log10(P-value). Therefore, we can apply a threshold for such quality score.

Here an example:

```
ls *.bam | parallel --progress --eta -j 15 'samtools view -bq 10 {} > {.}_mapQ10.bam'
```

Here the example for fetching uniquely mapped reads (from STAR):

```
for file in `ls *.ex.bam`
do
  outputname=`basename $file | sed -e "s/.bam/.unique.bam/"`
    (samtools view -H $file; samtools view -F 2308 $file | grep -w 'NH:i:1') | \
    samtools view -bS - > "$outputname"
  echo $file
  echo $outputname
done
```

Before and after filtering you can collect alignment statistics on the BAM file. This step can help with data handling and see whether there are biases on the sequencing you are analyzing. One of the most used tool is picard (https://broadinstitute.github.io/picard/). Picard can be wrapped using picardmetrics (https://github.com/slowkow/picardmetrics), a tool that will collect directly multiple statictis.

```
find . -name "*.bam" | \
  xargs -n 1 -P 12 -iFILES sh -c 'picardmetrics run -k -r -o PICARDMETRICS_OUT/ FILES;
```

1.7 Lift over

When analyzing different species you need a good quality gene annotation for the downstream quantification. Unfortunately, the best annotations are always based on model species as **Human** and **Mouse**.

Therefore, it is necessary to translate coordinates of non-model species into model species for a better annotation. You can use these translated files with the primary annotation for the model species (in this case Human).

There are tools that can help you with that: **liftOver** (https://genome.ucsc.edu/cgi-bin/hgLiftOver) and **CrossMap** (http://crossmap.sourceforge.net/)

These methods are based on cross-species annotations that you can easily find at the **UCSC Genome Brower** website (e.g. https://hgdownload.soe.ucsc.edu/goldenPath/hg38/liftOver/).

Here an example for CrossMap:

```
# Lifting a chimpanzee bam to human coordinates
python CrossMap.py bam panTro4ToHg38.over.chain.gz Chimp_Input.bam Chimp_Output
```

1.8 Quantification

BAM files contain reads mapped to the genome with specific genomic coordinates (e.g. chromosome, start, end, strand). These information are used for the quantification per gene counting the reads that overlap with the genomic location of a specific gene. Nevertheless there are some issue the quantifications tool are aware of (e.g. different genes found in overlapping genomic locations but opposite strand). These confounding factors are taken into account during quantifications.

There are several typo of quantification methods.

Alignment free (or pseudoalignment):

- Salmon (https://salmon.readthedocs.io/en/latest/salmon.html)
- Kallisto (https://pachterlab.github.io/kallisto/)
- Sailfish (https://www.cs.cmu.edu/~ckingsf/software/sailfish/)

Alignment based (they need a GTF and aligned reads):

- HTSeq-Count (https://htseq.readthedocs.io/en/release_0.11.1/index.html)
- featureCounts (http://bioinf.wehi.edu.au/featureCounts/)
- RSEM (https://github.com/deweylab/RSEM)

Here one example for HTSeq:

```
parallel -j 14 'samtools view {} | htseq-count -m intersection-strict \
-t exon \
-i gene_name \
-s reverse \
- Human_Genome_Directory/gene_annotation.gtf > {.}.txt' ::: *.unique.bam
```

The result of the quantification is a value per each gene per each sample. This is the raw count that determine how much this gene is expressed.

Chapter 2

Data Exploration

The starting data consist in RNA-seq count matrix after quantification (exp), a demographic table (demo) and a gene length table (width). The count table is a integer matrix without any normalizations. Each rows correspond to a gene, each column represent a different sample from three species (human, chimpanzee and rhesus macaque). The genes are protein-coding and orthologous between the three species.

2.0.1 Data loading

So let's start!

```
# First load the libraries we will use in this section
suppressPackageStartupMessages(library(sva))
suppressPackageStartupMessages(library(DESeq2))
suppressPackageStartupMessages(library(limma))
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(openxlsx))
suppressPackageStartupMessages(library(ggplot2))
suppressPackageStartupMessages(library(ggpubr))
suppressPackageStartupMessages(library(pheatmap))
suppressPackageStartupMessages(library(here))
suppressPackageStartupMessages(library(future.apply))
suppressPackageStartupMessages(library(broom))

# Clean the env
rm(list = ls())

# the data is stored under the subdirectory 'data'
```

```
list.files()
##  [1] "_book"                    "_bookdown_files"
##  [3] "_bookdown.yml"            "_build.sh"
##  [5] "_deploy.sh"               "_output.yml"
##  [7] "01-introPipes.Rmd"        "02-DataExploration.Rmd"
##  [9] "03-DiffExpAnalysis.Rmd" "addson"
## [11] "book.bib"                 "bookdown-demo_cache"
## [13] "bookdown-demo_files"      "bookdown-demo.Rmd"
## [15] "bookdown-demo.Rproj"      "DESCRIPTION"
## [17] "Dockerfile"               "images"
## [19] "index.Rmd"                "LICENSE"
## [21] "now.json"                 "packages.bib"
## [23] "peb_data"                 "preamble.tex"
## [25] "README.md"                "style.css"
## [27] "toc.css"

# load the data you need
load(here("peb_data", "PEB_2020.RData"))

# Check what data you loaded
ls()
## [1] "demo"  "exp"   "width"
```

The **exp** is the data frame containing raw counts.

The **demo** is the demographic data with some biological and technical covariates

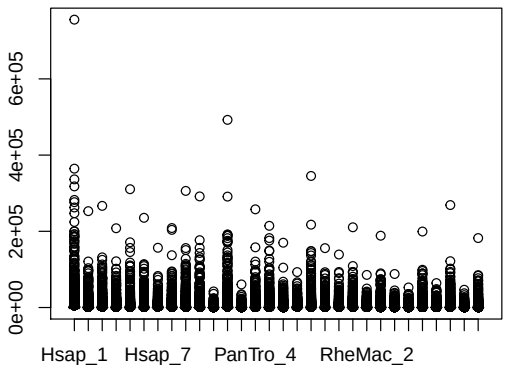
The **width** is the length of the the genes for data normalizations.

```
# Check the dimension of exp
dim(exp)
## [1] 13291    30

# Let's have a look to the data distribution of the
# expression
summary(exp)
##      Hsap_1           Hsap_2           Hsap_3
##  Min.   :     0   Min.   :     0   Min.   :     0
##  1st Qu.:    28   1st Qu.:    13   1st Qu.:    14
##  Median :   464   Median :   156   Median :   195
##  Mean   :  3415   Mean   :  1209   Mean   :  1427
##  3rd Qu.:  2256   3rd Qu.:   787   3rd Qu.:   974
##  Max.   :755442   Max.   :252898   Max.   :266769
##      Hsap_4           Hsap_5           Hsap_6
##  Min.   :     0   Min.   :     0   Min.   :     0
##  1st Qu.:     9   1st Qu.:    13   1st Qu.:    12
```

```
##  Median :   110   Median :   163   Median :   176
##  Mean   :   977   Mean   :  1419   Mean   :  1217
##  3rd Qu.:   578   3rd Qu.:   882   3rd Qu.:   823
##  Max.   :208039   Max.   :310497   Max.   :235453
##      Hsap_7           Hsap_8           Hsap_9
##  Min.   :     0   Min.   :     0   Min.   :     0
##  1st Qu.:    19   1st Qu.:    14   1st Qu.:     9
##  Median :   238   Median :   142   Median :   132
##  Mean   :  1150   Mean   :  1080   Mean   :  1212
##  3rd Qu.:   990   3rd Qu.:   642   3rd Qu.:   704
##  Max.   :156780   Max.   :208555   Max.   :306040
##     Hsap_10          PanTro_1         PanTro_2
##  Min.   :     0   Min.   :    0   Min.   :     0
##  1st Qu.:     9   1st Qu.:    8   1st Qu.:    13
##  Median :   146   Median :   72   Median :   188
##  Mean   :  1312   Mean   :  363   Mean   :  1942
##  3rd Qu.:   774   3rd Qu.:  321   3rd Qu.:  1048
##  Max.   :291565   Max.   :42193   Max.   :492501
##     PanTro_3         PanTro_4         PanTro_5
##  Min.   :    0   Min.   :     0   Min.   :     0
##  1st Qu.:    0   1st Qu.:     9   1st Qu.:     7
##  Median :   50   Median :   139   Median :   113
##  Mean   :  312   Mean   :  1125   Mean   :  1218
##  3rd Qu.:  235   3rd Qu.:   697   3rd Qu.:   670
##  Max.   :60608   Max.   :257678   Max.   :214389
##     PanTro_6         PanTro_7         PanTro_8
##  Min.   :     0   Min.   :    0   Min.   :     0
##  1st Qu.:     6   1st Qu.:   24   1st Qu.:    12
##  Median :    69   Median :  268   Median :   173
##  Mean   :   575   Mean   : 1067   Mean   :  1447
##  3rd Qu.:   345   3rd Qu.: 1000   3rd Qu.:   910
##  Max.   :170033   Max.   :92973   Max.   :345260
##     PanTro_9         PanTro_10        RheMac_1
##  Min.   :     0   Min.   :     0   Min.   :     0
##  1st Qu.:    12   1st Qu.:    15   1st Qu.:     9
##  Median :   165   Median :   169   Median :   136
##  Mean   :   995   Mean   :   936   Mean   :  1053
##  3rd Qu.:   695   3rd Qu.:   727   3rd Qu.:   678
##  Max.   :155820   Max.   :139103   Max.   :210549
##     RheMac_2         RheMac_3         RheMac_4
##  Min.   :    0   Min.   :     0   Min.   :    0
##  1st Qu.:   13   1st Qu.:     5   1st Qu.:    6
##  Median :  156   Median :    76   Median :   98
##  Mean   :  713   Mean   :   803   Mean   :  538
##  3rd Qu.:  610   3rd Qu.:   432   3rd Qu.:  419
```

```
##  Max.   :85555   Max.   :188456   Max.   :87970
##     RheMac_5        RheMac_6         RheMac_7
##  Min.   :    0   Min.   :     0   Min.   :    0
##  1st Qu.:    9   1st Qu.:     7   1st Qu.:   12
##  Median :   98   Median :   103   Median :  148
##  Mean   :  492   Mean   :   919   Mean   :  706
##  3rd Qu.:  399   3rd Qu.:   522   3rd Qu.:  590
##  Max.   :52623   Max.   :199397   Max.   :64114
##     RheMac_8        RheMac_9        RheMac_10
##  Min.   :    0   Min.   :    0   Min.   :     0
##  1st Qu.:   14   1st Qu.:    8   1st Qu.:     4
##  Median :  177   Median :   78   Median :    65
##  Mean   : 1137   Mean   :  429   Mean   :   697
##  3rd Qu.:  750   3rd Qu.:  334   3rd Qu.:   365
##  Max.   :268860   Max.   :46577   Max.   :182266
```

```
boxplot(exp)
```



2.0.2 Demographic screening

Demographics are important to understand what type of data you are dealing with. There are several factors that can influence expression: some are categorical variables (e.g. Sex, Hemisphere), others are continous variables (e.g. Age). These typically are important covariates to take into account into the analysis. In principle, RNA-seq is a snapshot of the gene expression in a specific moment of time from a specific tissue (bulk RNA-seq) or cell-type (single cell RNA-seq).

Different factors can play a role in explaining partially the variance of the gene expression you see.

For instance, gender can slightly differentiate gene expression therefore we presume it will have a minimal impact on gene expression variance. On the other hand variables like PMI (Post-mortem interval), RIN (rna integrity number), Brain Bank (the institute where you collect the data), Batch (sequencing days and hands), and Age (the age of the individual who donated the tissue) might have a sever impact on the gene expression variance.

These variables must be taken into account when RNA-seq (but also other genomics) is analyzed.

Now let's see what we have in the demographic

```
head(demo)
##         Species Sex Age RIN Hemisphere PMI
## Hsap_1    Hsap   M  42 9.8          L  13
## Hsap_2    Hsap   M  39 6.4          L  11
## Hsap_3    Hsap   F  25 8.8          R  18
## Hsap_4    Hsap   F  21 7.9          L   6
## Hsap_5    Hsap   M  45 6.6          L  16
## Hsap_6    Hsap   M  25 5.8          R   5

str(demo)
## 'data.frame':    30 obs. of  6 variables:
##  $ Species   : Factor w/ 3 levels "Hsap","PanTro",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Sex       : Factor w/ 2 levels "F","M": 2 2 1 1 2 2 1 2 2 2 ...
##  $ Age       : int  42 39 25 21 45 25 26 39 31 39 ...
##  $ RIN       : num  9.8 6.4 8.8 7.9 6.6 5.8 8.7 6.6 8.1 2.7 ...
##  $ Hemisphere: Factor w/ 2 levels "L","R": 1 1 2 1 1 2 2 2 2 2 ...
##  $ PMI       : int  13 11 18 6 16 5 19 18 10 18 ...
```

2.0.3 Data normalization

Do you see something weird in the boxplot?

Pretty sure you realized that the distribution is skewed toward the bottom.

There are several factors that can influence this:

- Library size: sequencing depth.
- Gene lenght: long gens = more reads
- Library composition: some biological factors can influence the transcriptome between samples.

- GC content: GC can influence mappability, therefore gene count.

Several methods have been developped to normalize the values prior downstream analysis. Here some examples:

- **CPM** (*Counts per million*): no length considered
- **RPKM** (*Reads per kilobase per million*): length considered
- **TPM** (*Transcript per million*): length considered
- **CQN** (*Conditional quantile normalization*): length and GC content considered

2.0.3.1 Calculate CPM

```
# Let's calculate the CPM
plan(multiprocess)

# future.apply is an alternative of the standard apply that
# goes in parallel.  here we apply a function to the
# expression matrix dividing each column by the total reads *
# one million.
cpm <- future_apply(exp, 2, function(x) x/sum(as.numeric(x)) *
  10^6)

# Let's have a look!
head(cpm)
##          Hsap_1  Hsap_2 Hsap_3  Hsap_4 Hsap_5  Hsap_6
## A1BG    3.65692 12.6344 7.4863 31.5862 9.3343 14.0957
## A1CF    0.15421  0.3734 0.1582  0.0000 0.5834  0.1236
## A2M     1.49801  4.1077 0.4218  0.0000 2.5457  1.5456
## A2ML1   0.41856  0.9958 0.4218  0.2311 1.0607  0.6182
## A3GALT2 1.93861  2.3651 1.9507  0.6934 0.5304  1.6692
## A4GALT  0.02203  0.2490 0.1054  0.1541 0.0000  0.1236
##         Hsap_7    Hsap_8  Hsap_9 Hsap_10 PanTro_1 PanTro_2
## A1BG    5.6898 194.65722 14.0305  2.4087  11.1888  0.23248
## A1CF    0.1962   0.06969  0.0000  0.0000   0.0000  0.00000
## A2M     1.1118   0.62725  0.6829  0.4588   1.2432  4.92086
## A2ML1   0.9810   3.13626  0.0000  0.4015   5.5944  0.11624
## A3GALT2 3.2046  11.70870  1.3037  0.4588   3.9368  0.19373
## A4GALT  0.1962   0.62725  0.2483  0.0000   0.2072  0.03875
##         PanTro_3 PanTro_4 PanTro_5 PanTro_6 PanTro_7
## A1BG      23.145  1.33764  0.74119   3.4020   7.3308
```

```
## A1CF       0.000  0.06688  0.06177   0.0000   0.2115
## A2M        5.545  0.13376  1.05002   1.8318   0.6344
## A2ML1      0.000  0.53506  0.24706   0.3925   1.4098
## A3GALT2    6.751  1.20388  0.43236   1.0468   4.1588
## A4GALT     0.000  0.00000  0.00000   0.2617   0.4229
##          PanTro_8 PanTro_9 PanTro_10 RheMac_1 RheMac_2
## A1BG       0.7277   5.2930    4.3415   0.5717   1.1613
## A1CF       0.3639   0.8318    0.0804   0.2144   0.1056
## A2M        1.7153   1.1342    0.4020   0.6431   0.7390
## A2ML1      0.6238   1.5879    1.2864   0.2858   0.5279
## A3GALT2    0.6238   2.3440    4.8239   1.0004   0.9502
## A4GALT     0.1559   0.3781    0.0000   0.2858   0.2112
##          RheMac_3 RheMac_4 RheMac_5 RheMac_6 RheMac_7
## A1BG       0.1875   0.1398   1.5296   0.0000   3.0919
## A1CF       0.1875   0.0000   0.0000   0.0000   0.1066
## A2M        2.2501   0.0000   1.9885   2.3751   0.7463
## A2ML1      0.1875   0.6988   0.7648   0.1638   0.6397
## A3GALT2    0.1875   0.0000   1.5296   0.4914   1.0662
## A4GALT     0.0000   0.1398   0.0000   0.0819   0.3198
##          RheMac_8 RheMac_9 RheMac_10
## A1BG      0.06617   0.7008    0.6475
## A1CF      0.06617   0.1752    0.0000
## A2M       7.27894   2.2776    1.0792
## A2ML1     0.52938   0.3504    0.0000
## A3GALT2   0.99258   3.6793    0.7555
## A4GALT    0.13234   0.0000    0.0000

# The sum of each column is = 10^6.
colSums(cpm)
##    Hsap_1    Hsap_2    Hsap_3    Hsap_4    Hsap_5    Hsap_6
##     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06
##    Hsap_7    Hsap_8    Hsap_9   Hsap_10  PanTro_1  PanTro_2
##     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06
##  PanTro_3  PanTro_4  PanTro_5  PanTro_6  PanTro_7  PanTro_8
##     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06
##  PanTro_9 PanTro_10  RheMac_1  RheMac_2  RheMac_3  RheMac_4
##     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06
##  RheMac_5  RheMac_6  RheMac_7  RheMac_8  RheMac_9 RheMac_10
##     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06
```

2.0.3.2 Calculate RPKM

```
# For RPKM you need the length of the genes (width)
head(width)
##      Gene Length GCperc
## 1    A1BG   4006 0.5580
## 2    A1CF   9603 0.3624
## 3     A2M   6384 0.3718
## 4   A2ML1   7303 0.4423
## 5 A3GALT2   1023 0.5419
## 6  A4GALT   2943 0.5239

# let's create a vector with the gene length
l <- as.vector(width$Length)

# Calculate the RPKM
rpkm <- future_apply(exp, 2, function(x) 10^9 * x/l/sum(as.numeric(x)))

# Let's have a look!
head(rpkm)
##           Hsap_1  Hsap_2  Hsap_3  Hsap_4  Hsap_5  Hsap_6
## A1BG    0.912860 3.15386 1.86878 7.88473 2.33008 3.51865
## A1CF    0.016058 0.03889 0.01647 0.00000 0.06075 0.01288
## A2M     0.234651 0.64344 0.06607 0.00000 0.39877 0.24210
## A2ML1   0.057314 0.13636 0.05775 0.03165 0.14524 0.08465
## A3GALT2 1.895021 2.31188 1.90681 0.67777 0.51843 1.63170
## A4GALT  0.007485 0.08459 0.03583 0.05235 0.00000 0.04201
##           Hsap_7    Hsap_8  Hsap_9 Hsap_10 PanTro_1 PanTro_2
## A1BG    1.42033 48.591418 3.50236 0.60128   2.7930  0.05803
## A1CF    0.02043  0.007258 0.00000 0.00000   0.0000  0.00000
## A2M     0.17416  0.098254 0.10697 0.07187   0.1947  0.77081
## A2ML1   0.13433  0.429448 0.00000 0.05497   0.7660  0.01592
## A3GALT2 3.13257 11.445459 1.27441 0.44849   3.8483  0.18938
## A4GALT  0.06667  0.213134 0.08438 0.00000   0.0704  0.01317
##         PanTro_3 PanTro_4 PanTro_5 PanTro_6 PanTro_7
## A1BG      5.7777 0.333909 0.185019  0.84922  1.82995
## A1CF      0.0000 0.006965 0.006432  0.00000  0.02202
## A2M       0.8686 0.020953 0.164476  0.28694  0.09937
## A2ML1     0.0000 0.073265 0.033830  0.05375  0.19304
## A3GALT2   6.5989 1.176810 0.422639  1.02322  4.06531
## A4GALT    0.0000 0.000000 0.000000  0.08892  0.14371
##         PanTro_8 PanTro_9 PanTro_10 RheMac_1 RheMac_2
## A1BG     0.18166  1.32126  1.083746  0.14270  0.28990
## A1CF     0.03789  0.08661  0.008372  0.02232  0.01099
## A2M      0.26869  0.17766  0.062968  0.10074  0.11576
## A2ML1    0.08541  0.21743  0.176142  0.03914  0.07228
## A3GALT2  0.60974  2.29133  4.715420  0.97791  0.92883
```

```
## A4GALT   0.05299  0.12846  0.000000  0.09712  0.07175
##          RheMac_3 RheMac_4 RheMac_5 RheMac_6 RheMac_7
## A1BG     0.04681  0.03489   0.3818  0.00000  0.77181
## A1CF     0.01953  0.00000   0.0000  0.00000  0.01110
## A2M      0.35246  0.00000   0.3115  0.37203  0.11690
## A2ML1    0.02568  0.09569   0.1047  0.02243  0.08759
## A3GALT2  0.18329  0.00000   1.4952  0.48034  1.04219
## A4GALT   0.00000  0.04749   0.0000  0.02783  0.10868
##          RheMac_8 RheMac_9 RheMac_10
## A1BG     0.016518  0.17494    0.1616
## A1CF     0.006891  0.01824    0.0000
## A2M      1.140185  0.35677    0.1691
## A2ML1    0.072488  0.04798    0.0000
## A3GALT2 0.970267  3.59655    0.7385
## A4GALT   0.044969  0.00000    0.0000

# Now the sum are all different!.
colSums(rpkm)
##    Hsap_1    Hsap_2    Hsap_3    Hsap_4    Hsap_5    Hsap_6
##    135992    137546    136729    136757    135507    137081
##    Hsap_7    Hsap_8    Hsap_9   Hsap_10  PanTro_1  PanTro_2
##    140837    140161    135588    134455    149958    133939
##  PanTro_3  PanTro_4  PanTro_5  PanTro_6  PanTro_7  PanTro_8
##    145104    135347    132978    138628    141954    135596
##  PanTro_9 PanTro_10  RheMac_1  RheMac_2  RheMac_3  RheMac_4
##    138648    139438    134781    142715    132472    139544
##  RheMac_5  RheMac_6  RheMac_7  RheMac_8  RheMac_9 RheMac_10
##    141869    133605    140033    136176    142478    132871
```

2.0.3.3 Calculate TPM

```
# For TPM you need the length of the genes (l)

# First step: Calculate the reads per kilobase
rpk <- future_apply(exp, 2, function(x) x/(l/1000))

# Now the TPM
tpm <- future_apply(rpk, 2, function(x) x/sum(as.numeric(x)) *
  10^6)

# Let's have a look!
head(tpm)
##           Hsap_1  Hsap_2  Hsap_3  Hsap_4  Hsap_5   Hsap_6
```

```
## A1BG     6.71259 22.9295 13.6678 57.6550 17.1953 25.66842
## A1CF     0.11808  0.2827  0.1205  0.0000  0.4483  0.09393
## A2M      1.72548  4.6780  0.4832  0.0000  2.9428  1.76613
## A2ML1    0.42145  0.9914  0.4224  0.2314  1.0719  0.61755
## A3GALT2 13.93477 16.8080 13.9459  4.9560  3.8259 11.90319
## A4GALT   0.05504  0.6150  0.2620  0.3828  0.0000  0.30649
##          Hsap_7    Hsap_8  Hsap_9 Hsap_10 PanTro_1 PanTro_2
## A1BG    10.0849 346.68275 25.8309  4.4720  18.6252   0.4333
## A1CF     0.1451   0.05178  0.0000  0.0000   0.0000   0.0000
## A2M      1.2366   0.70101  0.7889  0.5345   1.2986   5.7550
## A2ML1    0.9538   3.06396  0.0000  0.4088   5.1083   0.1188
## A3GALT2 22.2425  81.65935  9.3991  3.3356  25.6623   1.4139
## A4GALT   0.4734   1.52063  0.6223  0.0000   0.4695   0.0983
##         PanTro_3 PanTro_4 PanTro_5 PanTro_6 PanTro_7
## A1BG      39.817  2.46706  1.39135   6.1259  12.8912
## A1CF       0.000  0.05146  0.04837   0.0000   0.1551
## A2M        5.986  0.15481  1.23687   2.0699   0.7000
## A2ML1      0.000  0.54131  0.25441   0.3877   1.3599
## A3GALT2   45.477  8.69476  3.17826   7.3811  28.6383
## A4GALT     0.000  0.00000  0.00000   0.6414   1.0124
##         PanTro_8 PanTro_9 PanTro_10 RheMac_1 RheMac_2
## A1BG      1.3397   9.5296   7.77226   1.0588  2.03133
## A1CF      0.2794   0.6247   0.06004   0.1656  0.07704
## A2M       1.9816   1.2814   0.45159   0.7474  0.81116
## A2ML1     0.6299   1.5682   1.26323   0.2904  0.50649
## A3GALT2   4.4967  16.5262  33.81741   7.2556  6.50828
## A4GALT    0.3908   0.9265   0.00000   0.7206  0.50274
##         RheMac_3 RheMac_4 RheMac_5 RheMac_6 RheMac_7
## A1BG      0.3533   0.2500   2.6915   0.0000  5.51160
## A1CF      0.1474   0.0000   0.0000   0.0000  0.07928
## A2M       2.6606   0.0000   2.1956   2.7846  0.83483
## A2ML1     0.1938   0.6857   0.7382   0.1679  0.62552
## A3GALT2   1.3836   0.0000  10.5396   3.5953  7.44243
## A4GALT    0.0000   0.3403   0.0000   0.2083  0.77611
##         RheMac_8 RheMac_9 RheMac_10
## A1BG      0.1213   1.2278     1.217
## A1CF      0.0506   0.1281     0.000
## A2M       8.3729   2.5041     1.272
## A2ML1     0.5323   0.3368     0.000
## A3GALT2   7.1251  25.2428     5.558
## A4GALT    0.3302   0.0000     0.000

# The sum of each column is = 10^6!.
colSums(tpm)
##    Hsap_1    Hsap_2    Hsap_3    Hsap_4    Hsap_5    Hsap_6
```

```
##     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06
##   Hsap_7    Hsap_8    Hsap_9   Hsap_10  PanTro_1  PanTro_2
##     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06
##  PanTro_3  PanTro_4  PanTro_5  PanTro_6  PanTro_7  PanTro_8
##     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06
##  PanTro_9 PanTro_10  RheMac_1  RheMac_2  RheMac_3  RheMac_4
##     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06
##  RheMac_5  RheMac_6  RheMac_7  RheMac_8  RheMac_9 RheMac_10
##     1e+06     1e+06     1e+06     1e+06     1e+06     1e+06
```

2.0.3.4 Calculate CQN

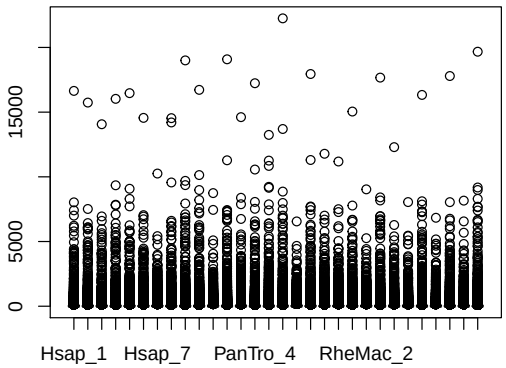
```
suppressPackageStartupMessages(library(cqn))

# let's create a vector with the gene length and gc content
length <- as.vector(width$Length)
gc_content <- as.vector(width$GCperc)

# Calculate CQN and normalized values
temp <- cqn(exp, lengths = length, x = gc_content, sizeFactors = colSums(exp),
  lengthMethod = "fixed", sqn = FALSE)

# get log2(normalized RPKM) values
quantGC <- 2^(temp$y + temp$offset)

# Let's have a look!
head(quantGC)
##          Hsap_1   Hsap_2  Hsap_3    Hsap_4  Hsap_5  Hsap_6
## A1BG    2.56927 10.55845 5.08924 30.841415 9.74631 8.55624
## A1CF    0.01205  0.02649 0.01316  0.004706 0.03552 0.01240
## A2M     0.15803  0.39396 0.04539  0.007449 0.23461 0.16546
## A2ML1   0.06277  0.15799 0.06846  0.054177 0.18876 0.09853
## A3GALT2 4.70302  6.63554 4.55666  2.526362 1.98350 3.65302
## A4GALT  0.03149  0.24275 0.10567  0.220901 0.05107 0.11910
##          Hsap_7   Hsap_8    Hsap_9  Hsap_10 PanTro_1
## A1BG    1.52687 156.7761 15.730795 2.678769  1.68308
## A1CF    0.01873   0.0109  0.004176 0.003558  0.03280
## A2M     0.12593   0.0843  0.080090 0.050280  0.30675
## A2ML1   0.12394   0.5023  0.011012 0.078499  0.64153
## A3GALT2 3.36678  30.5082  4.944737 1.853574  2.52037
## A4GALT  0.09204   0.5077  0.314778 0.058764  0.09251
##         PanTro_2 PanTro_3 PanTro_4 PanTro_5 PanTro_6
## A1BG    0.323282  7.57013 1.203895 1.307150  2.22728
```

```
## A1CF    0.002187  0.02701 0.008085 0.006478  0.01398
## A2M     0.456473  0.90128 0.018918 0.093457  0.29765
## A2ML1   0.032910  0.02960 0.100878 0.059772  0.08046
## A3GALT2 0.968989  8.16268 3.743524 2.591185  2.57671
## A4GALT  0.097419  0.08978 0.058774 0.089540  0.25963
##         PanTro_7 PanTro_8 PanTro_9 PanTro_10 RheMac_1
## A1BG     1.36851  0.80972  1.83918   1.15951  0.47642
## A1CF     0.02369  0.02137  0.08902   0.01406  0.01753
## A2M      0.08660  0.14582  0.17202   0.06197  0.06846
## A2ML1    0.14986  0.11238  0.22794   0.17717  0.05387
## A3GALT2  3.05732  2.36518  3.10524   5.06858  2.79150
## A4GALT   0.12378  0.21146  0.19201   0.02893  0.28296
##         RheMac_2 RheMac_3 RheMac_4 RheMac_5 RheMac_6
## A1BG     0.24532  0.38822  0.09098  0.35784 0.073764
## A1CF     0.02100  0.01422  0.01176  0.01670 0.005465
## A2M      0.11888  0.19411  0.01709  0.32997 0.255428
## A2ML1    0.06967  0.05314  0.10073  0.10292 0.042981
## A3GALT2  0.80543  1.29033  0.16979  1.39610 1.816089
## A4GALT   0.08459  0.12283  0.11129  0.04386 0.157588
##         RheMac_7 RheMac_8 RheMac_9 RheMac_10
## A1BG     0.66299  0.05289  0.17867  1.095789
## A1CF     0.02118  0.01095  0.04853  0.005946
## A2M      0.12278  0.92096  0.46590  0.106475
## A2ML1    0.08463  0.09071  0.06470  0.021158
## A3GALT2  0.95892  1.61560  3.16347  4.244491
## A4GALT   0.12191  0.10148  0.05126  0.153846

# The sum of each column is = 10^6!.
colSums(quantGC)
##    Hsap_1    Hsap_2    Hsap_3    Hsap_4    Hsap_5    Hsap_6
##    136938    135675    132641    148347    140195    136839
##    Hsap_7    Hsap_8    Hsap_9   Hsap_10  PanTro_1  PanTro_2
##    117928    159501    155370    143235    133363    149459
##  PanTro_3  PanTro_4  PanTro_5  PanTro_6  PanTro_7  PanTro_8
##    147426    134711    138760    165924    108437    128301
##  PanTro_9 PanTro_10  RheMac_1  RheMac_2  RheMac_3  RheMac_4
##    143486    132448    132904    120266    134040    128488
##  RheMac_5  RheMac_6  RheMac_7  RheMac_8  RheMac_9 RheMac_10
##    127934    145847    122743    142587    140086    137997
```

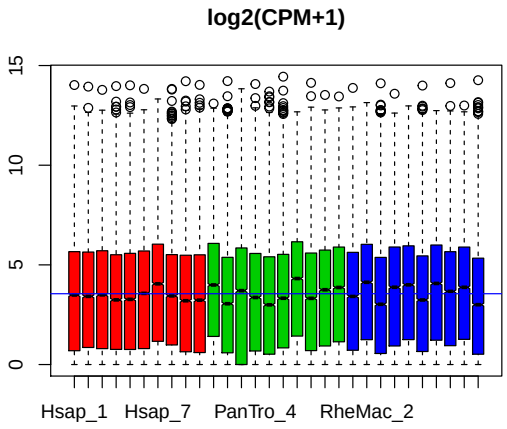
2.0.3.5 Check the normalized data

Now we can have a look to the distribution of the normalized data

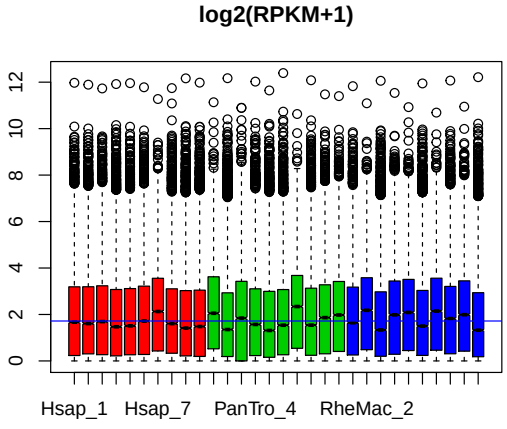
```
# Let's color based on the species
colors <- as.numeric(factor(demo$Species)) + 1

# Check the boxplot for CPM

boxplot(cpm)
```



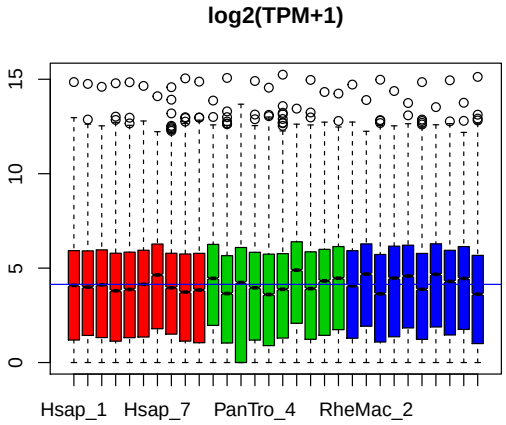

```
# Still skewed? Let's make every normally distributed the +1
# is a offset reads to keep the 0 as 0!
boxplot(log2(cpm + 1), notch = TRUE, main = "log2(CPM+1)", col = colors)
abline(h = median(log2(cpm + 1)), col = "blue")
```



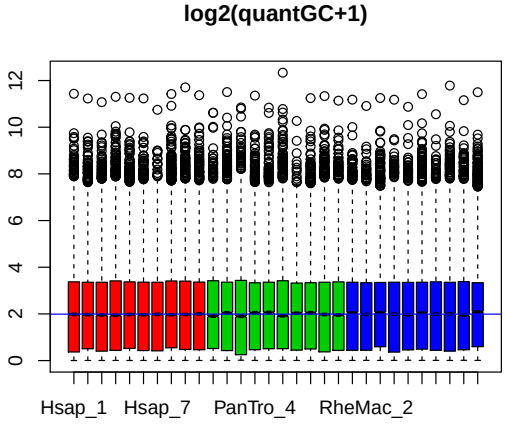

```
boxplot(log2(rpkm + 1), notch = TRUE, main = "log2(RPKM+1)",
  col = colors)
abline(h = median(log2(rpkm + 1)), col = "blue")
```




```
boxplot(log2(tpm + 1), notch = TRUE, main = "log2(TPM+1)", col = colors)
abline(h = median(log2(tpm + 1)), col = "blue")
```




```
boxplot(log2(quantGC + 1), notch = TRUE, main = "log2(quantGC+1)",
  col = colors)
abline(h = median(log2(quantGC + 1)), col = "blue")
```



2.0.4 Initial Data Exploration

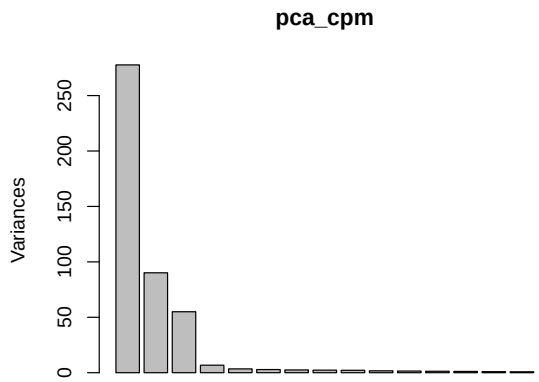
Now the counts are normalized and we know what we are dealing with, we can start to understand how the samples are similar/dissimilar based on the

quantified gene expression profiles of the genes. We would expect that biological/technical replicates will cluster together. This can be computed with different unsupervised clustering methods (e.g. hierarchical clustering) or with dimensionality reductions methods (e.g. Principal Component Analysis)

Let's start:

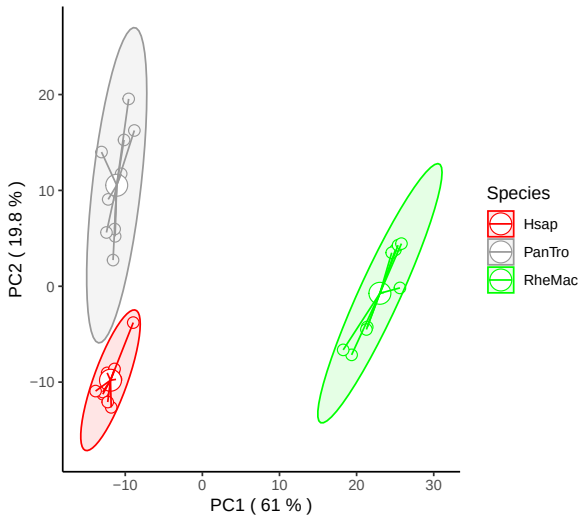
```
# First let's make a matric with log2 scaled data.
mat <- log2(cpm + 1)

# Calculate the variance for each gene This will detect the
# most variables genes
variance <- apply(mat, 1, var)

# Let's sort them and select the top 100 genes.
selgene <- names(variance[order(variance, decreasing = T)][1:100])

# Let's now calculate the Principal Components for the top
# 100 genes
pca_cpm <- prcomp(t(mat[selgene, ]))

# Print the summary of the pcas
summary(pca_cpm)
## Importance of components:
##                           PC1   PC2   PC3    PC4     PC5
## Standard deviation     16.66 9.494 7.416 2.6041 1.85325
## Proportion of Variance  0.61 0.198 0.121 0.0149 0.00754
## Cumulative Proportion   0.61 0.808 0.929 0.9434 0.95096
##                            PC6     PC7     PC8     PC9
## Standard deviation     1.69094 1.57609 1.52093 1.48110
## Proportion of Variance 0.00628 0.00546 0.00508 0.00482
## Cumulative Proportion  0.95724 0.96270 0.96778 0.97259
##                           PC10   PC11    PC12    PC13
## Standard deviation     1.30788 1.2250 1.14155 1.07289
## Proportion of Variance 0.00376 0.0033 0.00286 0.00253
## Cumulative Proportion  0.97635 0.9796 0.98251 0.98503
##                           PC14    PC15    PC16    PC17
## Standard deviation     0.96525 0.91623 0.86372 0.80614
## Proportion of Variance 0.00205 0.00184 0.00164 0.00143
## Cumulative Proportion  0.98708 0.98892 0.99056 0.99199
##                           PC18   PC19    PC20    PC21
## Standard deviation     0.7399 0.7074 0.63831 0.63085
## Proportion of Variance 0.0012 0.0011 0.00089 0.00087
## Cumulative Proportion  0.9932 0.9943 0.99519 0.99606
##                           PC22    PC23   PC24   PC25
## Standard deviation     0.58808 0.54664 0.5226 0.4790
```

```
## Proportion of Variance 0.00076 0.00066 0.0006 0.0005
## Cumulative Proportion  0.99682 0.99747 0.9981 0.9986
##                           PC26    PC27    PC28    PC29
## Standard deviation     0.46119 0.43126 0.37833 0.32475
## Proportion of Variance 0.00047 0.00041 0.00031 0.00023
## Cumulative Proportion  0.99905 0.99945 0.99977 1.00000
##                           PC30
## Standard deviation     7.4e-15
## Proportion of Variance 0.0e+00
## Cumulative Proportion  1.0e+00

# Let's have a look which PCA explain more variance
screeplot(pca_cpm, npcs = 15, type = "barplot")
```

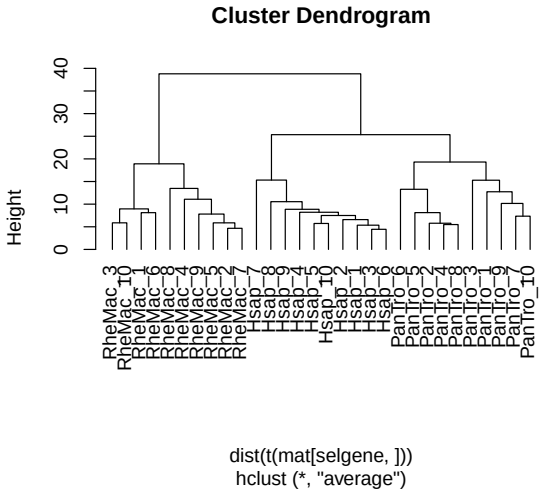



```
# Now let's create a data frame with the PCAs and the values
# for coloring
PCi <- data.frame(pca_cpm$x, Species = demo$Species)

# Let's calcualte the variance explained by the each
# components
eig <- (pca_cpm$sdev)^2
variance <- eig * 100/sum(eig)

# Now plot the PCA
ggscatter(PCi, x = "PC1", y = "PC2", color = "Species", palette = c("red",
  "grey60", "green"), shape = 21, size = 3, ellipse = TRUE,
  mean.point = TRUE, star.plot = TRUE) + xlab(paste("PC1 (",
```

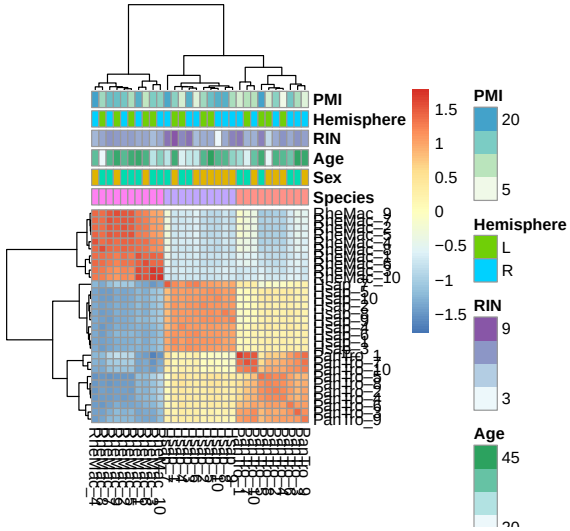
```
  round(variance[1], 1), "% )")) + ylab(paste("PC2 (", round(variance[2],
  1), "% )")) + theme_classic()
```




```
# Now let's make a dendrogram based on hirachical clustering
hc <- hclust(dist(t(mat[selgene, ])), "ave")
plot(hc, hang = -1)
```



```
# Another way with correlation
correlation_mat <- cor(mat[selgene, ], method = "pearson")
pheatmap(correlation_mat, annotation_col = demo, scale = "row")
```

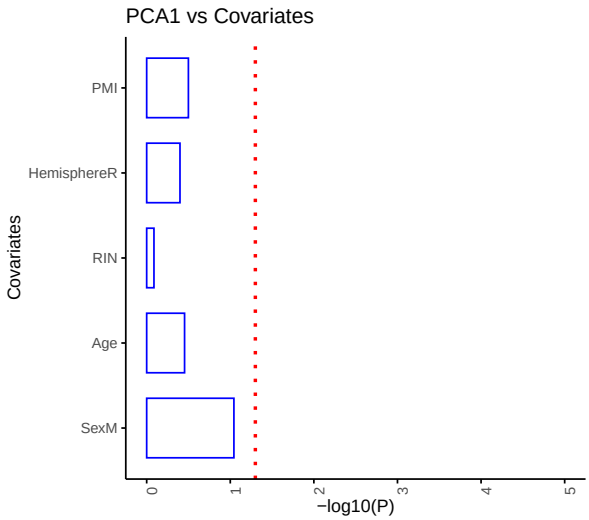


2.0.5 Variance explained by covariates

Now the big question: how much of the gene expression is explained by the biological/technical covaraites? The covaraites can introduces systematic shifts in the downstream analysis and a good quality check is indeed to evaluate the variance that is explained by each of these covariates.

```
# Re-check the demographic data
head(demo)
##         Species Sex Age RIN Hemisphere PMI
## Hsap_1    Hsap   M  42 9.8          L  13
## Hsap_2    Hsap   M  39 6.4          L  11
## Hsap_3    Hsap   F  25 8.8          R  18
## Hsap_4    Hsap   F  21 7.9          L   6
## Hsap_5    Hsap   M  45 6.6          L  16
## Hsap_6    Hsap   M  25 5.8          R   5

# Now let's calcualte the PCA for all the data (not just the
# top variant genes)
pca_all <- prcomp(t(log2(cpm + 1)))

# Let's make a temporary demographic without the species
```

```
tmp_demo <- demo[c(-1)]

# Let's include the first PC into the demo file.  This is the
# one that explain most of the variance in the data.
tmp_demo$pca1 <- pca_all$x[, 1]

# Make a model matrix for the model
mm <- as.data.frame(model.matrix(~., tmp_demo))

# Now we are going to fit a moodel between the first
# component and the other covaraites
fit1 = lm(pca1 ~ ., data = mm)

# Let's transform the fit into a dataframe with
# library(broom)
df <- tidy(fit1)[-1, ]

# Let's check the info reported
head(df)
## # A tibble: 5 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 SexM          -42.8      24.3     -1.76   0.0904
## 2 Age             1.38      1.45     0.949  0.352
## 3 RIN             1.95      8.40     0.233  0.818
## 4 HemisphereR    18.3      21.3      0.858  0.400
## 5 PMI            -2.40      2.35    -1.02   0.317

# The Pvalue reflect how much the covariates are associated
# with the first component.  Transform the p-value into a
# log10 scale.
df$log10 = -log10(df$p.value)

# Let's visualize the association based on this modeling
ggbarplot(df, x = "term", y = "log10", fill = "white", color = "blue",
  x.text.angle = 90, ylab = "-log10(P)", xlab = "Covariates",
  rotate = TRUE, ggtheme = theme_classic()) + geom_hline(yintercept = 1.3,
  linetype = "dotted", color = "red", size = 1) + ylim(0, 5) +
  ggtitle("PCA1 vs Covariates")
```



2.0.6 Save the data

Now it's time to save all the data we generated.

```
save(exp, cpm, rpkm, tpm, quantGC, demo, file = "peb_data/Normalized_data.RData")
```

2.0.7 Exercise for Data Exploration chapter

- Do data exploration for RPKM.
- Do data exploration for TPM
- Do data exploration for quantGC.
- Covaraite association with different normalized values.

Chapter 3

Differential Expression Analysis

Differential expression analysis allows to test tens of thousands of hypotheses (one test for each gene) against the null hypothesis that the gene expression is the same between two or multiple species. Some limiting factors: sample size, non-normally distribution of counts, high/low expressed genes. Therefore, to apply any statistics, it is necessary to check the data first and apply the right modeling. However, some tools such as DESeq2 address these limitations using statistical models in order to maximize the amount of knowledge that can be extracted from such noisy datasets.

In this chapter we will apply:

- **Linear Model**
- **DESeq2**

We will also use **Surrogates Variables (sva)**, a method used to detect sources of unwanted variation in high throughput sequencing.

We will then identify species specific differentially expressed genes which will be input for the functional/visualization chapter.

3.1 Data loading

So let's start!

```r
# First load the libraries we will use in this section
suppressPackageStartupMessages(library(sva))
suppressPackageStartupMessages(library(DESeq2))
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(openxlsx))
suppressPackageStartupMessages(library(ggplot2))
suppressPackageStartupMessages(library(ggpubr))
suppressPackageStartupMessages(library(pheatmap))
suppressPackageStartupMessages(library(here))
suppressPackageStartupMessages(library(future.apply))
suppressPackageStartupMessages(library(broom))
suppressPackageStartupMessages(library(ggrepel))
suppressPackageStartupMessages(library(DT))
suppressPackageStartupMessages(library(clusterProfiler))
suppressPackageStartupMessages(library(org.Hs.eg.db))

# Multicore activated
plan(multiprocess)

# the data is stored under the subdirectory 'data'
list.files()
##  [1] "_book"                    "_bookdown_files"
##  [3] "_bookdown.yml"            "_build.sh"
##  [5] "_deploy.sh"               "_output.yml"
##  [7] "01-introPipes.Rmd"        "02-DataExploration.Rmd"
##  [9] "03-DiffExpAnalysis.Rmd" "addson"
## [11] "book.bib"                 "bookdown-demo_cache"
## [13] "bookdown-demo_files"      "bookdown-demo.Rmd"
## [15] "bookdown-demo.Rproj"      "DESCRIPTION"
## [17] "Dockerfile"               "images"
## [19] "index.Rmd"                "LICENSE"
## [21] "now.json"                 "packages.bib"
## [23] "peb_data"                 "preamble.tex"
## [25] "README.md"                "style.css"
## [27] "toc.css"

# Remove previous loads
rm(list = ls())

# Create a directory where to save the data
dir.create("peb_data/output/")

# load the data you need
load(here("peb_data", "Normalized_data.RData"))
```

```
# Check what data you loaded
ls()
## [1] "cpm"     "demo"    "exp"     "quantGC" "rpkm"
## [6] "tpm"
```

3.2 DGE based on linear model

We are going to define changes in gene expression between all the three species. We will then apply a parsimony approach to define species-specific changes. Here a representation:

Let's start!

```
# Filter the demographic for human-chimpanzee
demoHC <- demo %>% rownames_to_column("ID") %>% filter(Species %in%
  c("Hsap", "PanTro")) %>% column_to_rownames("ID") %>% droplevels()  # Remove unwanted factors

# First remove the genes with 0s.
logCPM <- log2(cpm + 1)

# we need to filter the low expressed genes.  Here we are
# going to use a fitler where all the samples express the
# gene > 0.5
perc <- 100
vec = round((ncol(logCPM) * perc)/100)
notAllZero = (rowSums(logCPM > 0) >= vec)
logCPM_filtered = logCPM[notAllZero, ]

# you can also use a different percentage and/or a
# conditional filtering Not to run filter=apply(logCPM, 1,
# function(x) (all(x[grep('Hsap',names(x))] > 0) |
# all(x[grep('PanTro',names(x))] > 0)) |
# all(x[grep('RheMac',names(x))] > 0)) logCPM_filtered <-
# logCPM[filter,]

# Now for the cpm and log2 transform to make the data
# normally distributed fetching Human and Chimp sample

logCPM_HC <- logCPM_filtered[, grep("Hsap|PanTro", colnames(logCPM_filtered))]

# Now we can start to plan the linear modeling for the
# analysis!  The model you wanna fit with all the covariates.
model <- "GeneExpr ~ Species + Age + Sex + Hemisphere + PMI + RIN"
```

```r
# Create a temporary metadata
tmpDemo <- demoHC

# Function for fitting the model.
fit_lm <- function(vectorizedExpression) {
  tmpMetaData <- cbind(tmpDemo, data.frame(GeneExpr = unname(vectorizedExpression)))
  residuals <- lm(model, data = tmpMetaData)
  pval <- as.numeric(tidy(residuals)$p.value[2])
  effect_size <- as.numeric(tidy(residuals)$estimate[2])
  c(EffSize_LM = effect_size, Pval_LM = pval)
}

fit_the_mod <- function(vectorizedExpression) {
  tryCatch(fit_lm(vectorizedExpression), error = function(e) c(EffSize_LM = NA,
    Pval_LM = NA))
}

# Run the analysis and get the stats
hc_stat <- apply(logCPM_HC, 1, fit_the_mod) %>% t() %>% as.data.frame() %>%
  rownames_to_column("Gene") %>% # Adjust the p-value by FDR
mutate(FDR_LM = p.adjust(Pval_LM, method = "BH")) %>% # Relabel
dplyr::rename(EffSize_HC = EffSize_LM, Pval_HC = Pval_LM, FDR_HC = FDR_LM) %>%
  # Switch sign
mutate(EffSize_HC = -1 * EffSize_HC)

# Have a look to the data! head(hc_stat)
DT::datatable(hc_stat, options = list(pageLength = 10))
```

```r
# Now let's apply the same model for the other two
# comparisons (H vs R, C vs R)

# Demo for HR
demoHR <- demo %>% rownames_to_column("ID") %>% filter(Species %in%
  c("Hsap", "RheMac")) %>% column_to_rownames("ID") %>% droplevels()

# Demo for CR
demoCR <- demo %>% rownames_to_column("ID") %>% filter(Species %in%
  c("PanTro", "RheMac")) %>% column_to_rownames("ID") %>% droplevels()

# Get the gene expression for HR and CR
logCPM_HR <- logCPM_filtered[, grep("Hsap|RheMac", colnames(logCPM_filtered))]
logCPM_CR <- logCPM_filtered[, grep("PanTro|RheMac", colnames(logCPM_filtered))]

# Run the modeling HR
```

```
tmpDemo <- demoHR

hr_stat <- apply(logCPM_HR, 1, fit_the_mod) %>% t() %>% as.data.frame() %>%
  rownames_to_column("Gene") %>% mutate(FDR_LM = p.adjust(Pval_LM,
  method = "BH")) %>% dplyr::rename(EffSize_HR = EffSize_LM,
  Pval_HR = Pval_LM, FDR_HR = FDR_LM) %>% mutate(EffSize_HR = -1 *
  EffSize_HR)  # Switch sign

# have a look to the data just created
DT::datatable(hr_stat, options = list(pageLength = 10))
```

```
# Run the modeling PR
tmpDemo <- demoCR

cr_stat <- apply(logCPM_CR, 1, fit_the_mod) %>% t() %>% as.data.frame() %>%
  rownames_to_column("Gene") %>% mutate(FDR_LM = p.adjust(Pval_LM,
  method = "BH")) %>% dplyr::rename(EffSize_CR = EffSize_LM,
  Pval_CR = Pval_LM, FDR_CR = FDR_LM) %>% mutate(EffSize_CR = -1 *
  EffSize_CR)  # Switch sign

# have a look to the data just created
DT::datatable(cr_stat, options = list(pageLength = 10))
```

```
# Now it's time to combine all the data together and save the
# files
HCR_stat <- Reduce(dplyr::full_join, list(hc_stat, hr_stat, cr_stat))

openxlsx::write.xlsx(HCR_stat, file = "peb_data/output/HCR_stat.xlsx",
  colNames = TRUE, borders = "columns", sheetName = "Full Table")

save(hc_stat, hr_stat, cr_stat, HCR_stat, file = "peb_data/output/Linear_Modeling_Statistics.RDat

# have a look to the data just created
DT::datatable(HCR_stat, options = list(pageLength = 10))
```

3.3 Species Specific DGE

Now it's time to apply the parsimony to define the species-specific differentially expressed genes. This is based on adjusted p.value and also direction of the gene. For instance, As in the picture, the genes should be differentially expressed in H vs C, H vs C but not in C vs R. We expect that also the fold change (effect size) will have the same direction in H vs C and H vs R.

Let's start!

```
# Let's have another look to the input data
head(HCR_stat)
##    Gene EffSize_HC Pval_HC FDR_HC EffSize_HR   Pval_HR
## 1  AAAS    0.08334  0.8923 0.9618   -0.09687 8.146e-01
## 2  AACS   -0.18288  0.6488 0.8625   -0.55786 1.746e-01
## 3 AADAT   -0.45706  0.2017 0.5443   -1.03622 1.086e-03
## 4  AAK1    0.02924  0.7759 0.9152   -0.37570 3.593e-05
## 5  AAMP   -0.32675  0.6377 0.8577   -0.66929 2.743e-01
## 6 AANAT    0.34394  0.2938 0.6386   -0.64980 1.374e-01
##      FDR_HR EffSize_CR   Pval_CR   FDR_CR
## 1 0.8713298    -0.2380 6.944e-01 0.827769
## 2 0.3004664    -0.4988 3.094e-01 0.535762
## 3 0.0083901    -1.1605 1.813e-02 0.090267
## 4 0.0007026    -0.3745 6.045e-05 0.001625
## 5 0.4118502    -0.7806 2.000e-01 0.418384
## 6 0.2534909    -0.2207 6.361e-01 0.788821

# We are going to define the species specific genes now.  We
# are going to filter for FDRs and same direction of the
# effect sizes.

# Human first!
Hsap_Spec <- HCR_stat %>% filter(FDR_HC < 0.05, FDR_HR < 0.05,
  FDR_CR > 0.1, sign(EffSize_HC) == sign(EffSize_HR))

# Let's check how many genes survived
dim(Hsap_Spec)
## [1] 129  10

DT::datatable(Hsap_Spec, options = list(pageLength = 10))
```

```
# Now chimp specific
PanTro_Spec <- HCR_stat %>% filter(FDR_HC < 0.05, FDR_HR > 0.1,
  FDR_CR < 0.05, sign(-1 * EffSize_HC) == sign(EffSize_CR))

dim(PanTro_Spec)
## [1] 81 10

DT::datatable(PanTro_Spec, options = list(pageLength = 10))
```

```
# Now macaque specific
RheMac_Spec <- HCR_stat %>% filter(FDR_HC > 0.1, FDR_HR < 0.05,
  FDR_CR < 0.05, sign(-1 * EffSize_HR) == sign(-1 * EffSize_CR))

dim(RheMac_Spec)
## [1] 830  10

DT::datatable(RheMac_Spec, options = list(pageLength = 10))
```

```
# Save the data
save(Hsap_Spec, PanTro_Spec, RheMac_Spec, file = "peb_data/output/Species_Specific_DGE.RData")
```

3.4 Surrogate Variables

Now we ara going to add surrogates variable in the analysis.

```
# We need the expressed genes
head(logCPM_filtered)
##       Hsap_1 Hsap_2 Hsap_3 Hsap_4 Hsap_5 Hsap_6 Hsap_7
## AAAS  3.6938 3.1745 3.5225 3.7854 2.5814  3.540  3.829
## AACS  6.8571 6.6629 7.0101 6.3269 6.4212  6.977  8.100
## AADAT 4.6202 3.8461 4.5841 4.0090 4.1591  4.312  4.841
## AAK1  8.8296 8.7640 8.8367 8.9483 8.7082  8.854  9.082
## AAMP  2.6524 1.3886 1.6845 0.4701 1.8261  1.966  3.494
## AANAT 0.7127 0.7524 0.1446 1.5489 0.9272  1.036  1.501
##       Hsap_8 Hsap_9 Hsap_10 PanTro_1 PanTro_2 PanTro_3
## AAAS  3.7221 2.4839  2.4524    4.887   2.9599    5.058
## AACS  6.6292 6.9805  6.2730    8.362   6.7094    7.165
## AADAT 4.2883 4.2854  4.6223    3.725   5.0173    2.417
## AAK1  9.0357 8.7888  8.7831    8.498   8.9204    9.087
## AAMP  2.8204 0.5206  1.6157    3.558   0.7301    1.869
## AANAT 0.7629 0.5206  0.4266    1.518   0.1077    2.954
##       PanTro_4 PanTro_5 PanTro_6 PanTro_7 PanTro_8 PanTro_9
## AAAS    3.7121   2.1567    1.433    5.364   2.9164    3.365
## AACS    6.4997   5.1855    7.389    8.036   6.4443    7.547
## AADAT   4.4150   5.0815    2.546    4.650   4.6655    4.207
## AAK1    8.9209   8.9417    8.660    8.692   8.7896    8.970
## AAMP    1.1400   1.3472    1.746    3.896   1.1000    2.449
## AANAT   0.7955   0.5184    1.033    1.572   0.5017    1.239
##       PanTro_10 RheMac_1 RheMac_2 RheMac_3 RheMac_4
## AAAS      4.627   3.0880    4.037   2.5314    3.119
## AACS      7.816   6.6871    8.123   6.3658    7.759
```

```
## AADAT     3.952   5.7265    4.968   6.1274    5.685
## AAK1      8.799   9.3706    9.280   9.2022    9.175
## AAMP      2.894   2.1704    4.136   1.3751    3.023
## AANAT     1.665   0.6523    1.905   0.5546    2.780
##       RheMac_5 RheMac_6 RheMac_7 RheMac_8 RheMac_9
## AAAS     3.792    2.395    3.964   4.3729    4.083
## AACS     7.893    6.730    7.690   7.2772    7.685
## AADAT    4.446    5.261    5.527   5.3187    4.402
## AAK1     9.192    9.246    9.252   9.2576    9.306
## AAMP     3.281    1.399    3.517   3.3026    2.519
## AANAT    1.424    1.102    2.106   0.6129    1.550
##       RheMac_10
## AAAS     2.2562
## AACS     5.8028
## AADAT    5.9103
## AAK1     9.0968
## AAMP     1.1290
## AANAT    0.4046

# Calculate the surrogate variables using SVA First we need
# to create two model matrix, one for the fitting and one as
# null The null model contains all the covaraites except the
# predictor (Species)

mod <- model.matrix(~Species + Age + Sex + Hemisphere + PMI +
  RIN, data = demo)
mod0 <- model.matrix(~Age + Sex + Hemisphere + PMI + RIN, data = demo)

# Create SVA object (input the gene expression) with 100
# permuation.
svaobj <- sva(as.matrix(logCPM_filtered), mod, mod0, n.sv = NULL,
  B = 100, method = "two-step")
## Number of significant surrogate variables is:  3

# Get the surrogates variables
svaobj$sv = data.frame(svaobj$sv)
colnames(svaobj$sv) = c(paste0("SV", seq(svaobj$n.sv)))
metadata_sv <- cbind(demo, svaobj$sv)

# Recreate demographics
demoHC_sv <- metadata_sv %>% rownames_to_column("ID") %>% filter(Species %in%
  c("Hsap", "PanTro")) %>% column_to_rownames("ID") %>% droplevels()

demoHR_sv <- metadata_sv %>% rownames_to_column("ID") %>% filter(Species %in%
  c("Hsap", "RheMac")) %>% column_to_rownames("ID") %>% droplevels()
```

```
demoCR_sv <- metadata_sv %>% rownames_to_column("ID") %>% filter(Species %in%
  c("PanTro", "RheMac")) %>% column_to_rownames("ID") %>% droplevels()

# Let's add the SV into the model and run the analysis!
model <- as.formula(paste("GeneExpr ~", paste(c(colnames(demoHC_sv)),
  collapse = "+")))

tmpDemo <- demoHC_sv

# Run the analysis and get the stats
hc_stat_sva <- apply(logCPM_HC, 1, fit_the_mod) %>% t() %>% as.data.frame() %>%
  rownames_to_column("Gene") %>% mutate(FDR_LM = p.adjust(Pval_LM,
  method = "BH")) %>% dplyr::rename(EffSize_HC = EffSize_LM,
  Pval_HC = Pval_LM, FDR_HC = FDR_LM) %>% mutate(EffSize_HC = -1 *
  EffSize_HC)  # Switch sign

# Now we are going to apply the same method to HvsR Now
# create the new model including the surrogates variables.
model <- as.formula(paste("GeneExpr ~", paste(c(colnames(demoHR_sv)),
  collapse = "+")))

tmpDemo <- demoHR_sv

hr_stat_sva <- apply(logCPM_HR, 1, fit_the_mod) %>% t() %>% as.data.frame() %>%
  rownames_to_column("Gene") %>% mutate(FDR_LM = p.adjust(Pval_LM,
  method = "BH")) %>% dplyr::rename(EffSize_HR = EffSize_LM,
  Pval_HR = Pval_LM, FDR_HR = FDR_LM) %>% mutate(EffSize_HR = -1 *
  EffSize_HR)  # Switch sign

# Now we are going to apply the same method to CvsR Now
# create the new model including the surrogates variables.
model <- as.formula(paste("GeneExpr ~", paste(c(colnames(demoCR_sv)),
  collapse = "+")))

tmpDemo <- demoCR_sv

cr_stat_sva <- apply(logCPM_CR, 1, fit_the_mod) %>% t() %>% as.data.frame() %>%
  rownames_to_column("Gene") %>% mutate(FDR_LM = p.adjust(Pval_LM,
  method = "BH")) %>% dplyr::rename(EffSize_CR = EffSize_LM,
  Pval_CR = Pval_LM, FDR_CR = FDR_LM) %>% mutate(EffSize_CR = -1 *
  EffSize_CR)  # Switch sign

# Combine the data
HCR_stat_sva <- Reduce(dplyr::full_join, list(hc_stat_sva, hr_stat_sva,
  cr_stat_sva))
```

```
openxlsx::write.xlsx(HCR_stat_sva, file = "peb_data/output/HCR_stat_sva.xlsx",
  colNames = TRUE, borders = "columns", sheetName = "Full Table")

save(hc_stat_sva, hr_stat_sva, cr_stat_sva, HCR_stat_sva, file = "peb_data/output/Linea

save(svaobj, file = "peb_data/output/Surrogate_Variables.RData")

# Now calculate species specific genes with SVs

# Human first!
Hsap_Spec_sva <- HCR_stat_sva %>% filter(FDR_HC < 0.05, FDR_HR <
  0.05, FDR_CR > 0.1, sign(EffSize_HC) == sign(EffSize_HR))

# Let's check how many genes survived
dim(Hsap_Spec_sva)
## [1] 324  10

DT::datatable(Hsap_Spec_sva, options = list(pageLength = 10))
```

```
# Now chimp specific
PanTro_Spec_sva <- HCR_stat_sva %>% filter(FDR_HC < 0.05, FDR_HR >
  0.1, FDR_CR < 0.05, sign(-1 * EffSize_HC) == sign(EffSize_CR))

dim(PanTro_Spec_sva)
## [1] 199  10

DT::datatable(PanTro_Spec_sva, options = list(pageLength = 10))
```

```
# Now macaque specific
RheMac_Spec_sva <- HCR_stat_sva %>% filter(FDR_HC > 0.1, FDR_HR <
  0.05, FDR_CR < 0.05, sign(-1 * EffSize_HR) == sign(-1 * EffSize_CR))

dim(RheMac_Spec_sva)
## [1] 1933   10

DT::datatable(RheMac_Spec_sva, options = list(pageLength = 10))
```

```
# Save the data that we will explore later
save(Hsap_Spec_sva, PanTro_Spec_sva, RheMac_Spec_sva, file = "peb_data/output/Species_S
```

3.5 Balance the gene expression for covariates

We calculated the DGE with/without SVs based on linear modeling. However, the normalized data is not actually adjusted for any of these covariates. The covariates are taken into account in the modeling but the input data is not changed.

Why do we care about this?

Gene expression data should be balanced by these covariates because you have some variance that is explained by each covariate, minimal or large.

Applying residualization procedure, we will remove all the variance explained by covariates and eventually unwanted sources of variations.

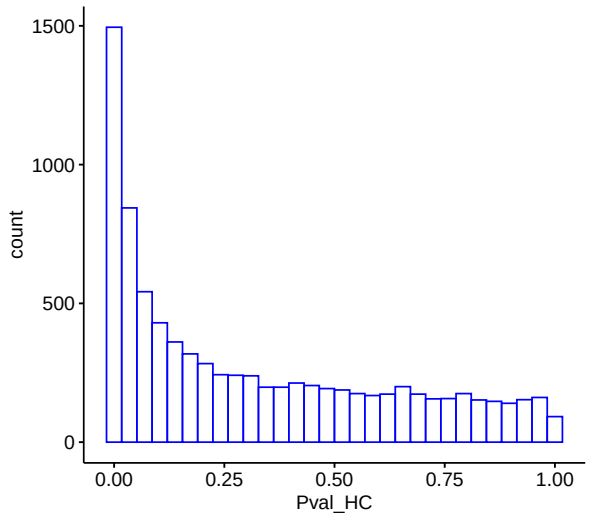
This step is important for any visualizations or additional analysis (e.g. coexpression based on correlation) you want to apply: the data must be adjusted before.

```
# For this part we will need the log2 scaled cpm filtered and
# the total demographic with the SVs.  We will regress out
# everything except SPECIES.  You don't want to remove the
# variance explained by species right? :-)

# Residualisation procedure
avebeta.lm <- lapply(1:nrow(logCPM_filtered), function(x) {
  # Remove Species!
  lm(unlist(logCPM_filtered[x, ]) ~ ., data = metadata_sv[c(-1)])
})
# Get the residuals
residuals <- lapply(avebeta.lm, function(x) residuals(summary(x)))
residuals <- do.call(rbind, residuals)

logCPM_adjusted <- residuals + matrix(apply(logCPM_filtered,
  1, mean), nrow = nrow(residuals), ncol = ncol(residuals))

rownames(logCPM_adjusted) <- rownames(logCPM_filtered)

# Save the data
save(logCPM_adjusted, file = "peb_data/output/logCPM_adjusted.RData")
```

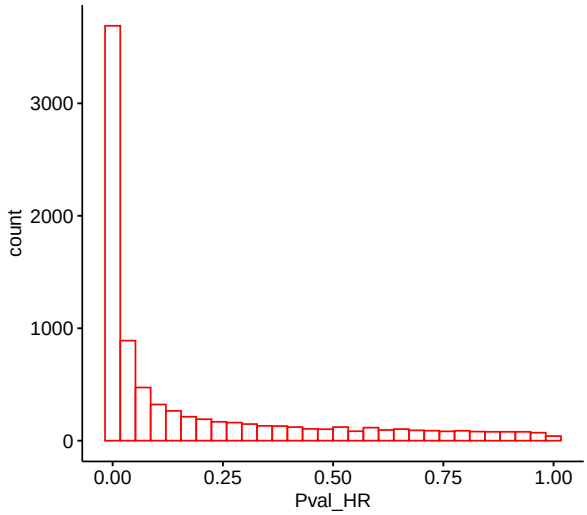
3.6 DGE visualizations

Now we calculated the species-specific DGE (with/without SVs). Now it's time to check the genes we idenfied. We will work on the SV adjusted data

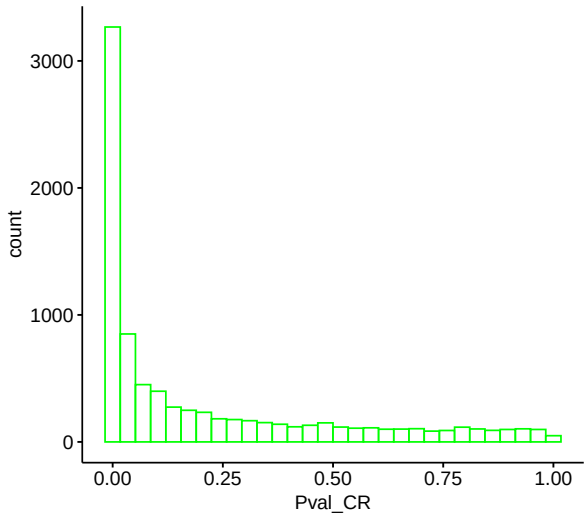
```
# Let's make some diagnostic plots!
# Let's have a look to the p-value distribution of H vs C
gghistogram(HCR_stat_sva,
            x = "Pval_HC",
            color = "blue"
            )
```




```
# Let's have a look to the p-value distribution of H vs R
gghistogram(HCR_stat_sva,
            x = "Pval_HR",
            color = "red"
            )
```

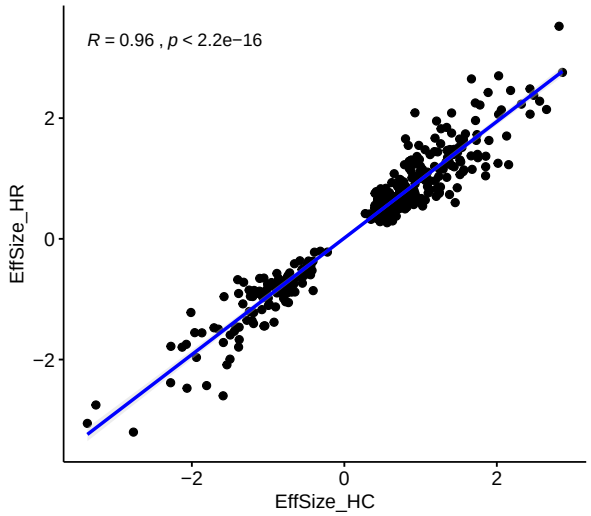



```
# Let's have a look to the p-value distribution of C vs R
gghistogram(HCR_stat_sva,
            x = "Pval_CR",
            color = "green"
            )
```

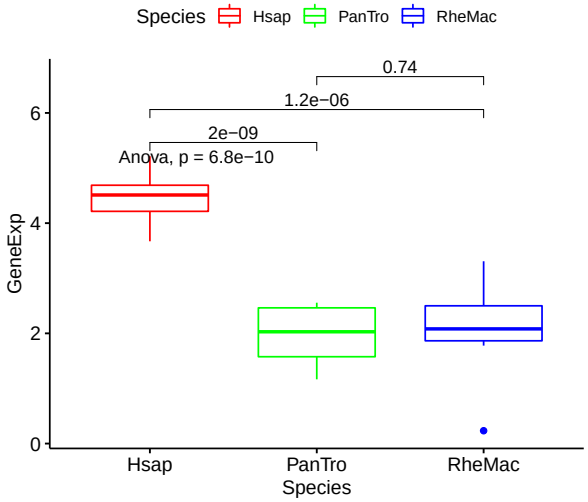



```
# Now let's have a look to the concordance between effect sizes of Hspc Genes
ggscatter(Hsap_Spec_sva,
          x = "EffSize_HC",
```

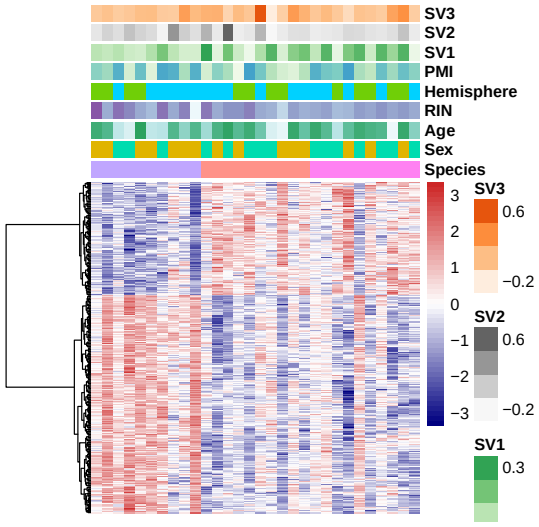
```
          y = "EffSize_HR",
          add = "reg.line",
          conf.int = TRUE,
          add.params = list(color = "blue",
                            fill = "lightgray")
          ) +
  stat_cor(method = "pearson")
```




```
# Boxplot for a gene
tmp <- data.frame(Species = demo$Species, GeneExp = as.numeric(logCPM_filtered["MET",])

comp <- list( c("Hsap", "PanTro"),
              c("Hsap", "RheMac"),
              c("PanTro", "RheMac"))

ggboxplot(tmp,
          x = "Species",
          y = "GeneExp",
          color = "Species",
          palette = c("red","green","blue"))+
  stat_compare_means(comparisons = comp, method = "t.test") +
  stat_compare_means(method = "anova")
```

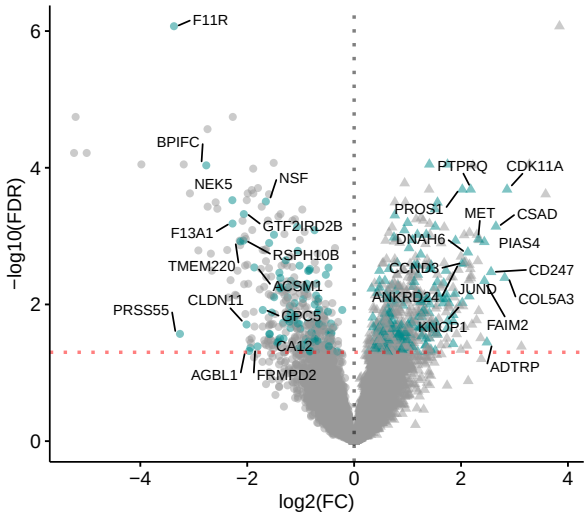



```
# let's make a heatmap of the human specifc genes!
# We will use the expression balanced by covariates and the SVs
# Let's make a matrix for the data
genes <- Hsap_Spec_sva$Gene
mat <- logCPM_adjusted[rownames(logCPM_adjusted) %in% genes,]

# Let's make a heatmap!
pheatmap(mat,
            cluster_cols=F,
            scale="row",
            color = colorRampPalette(c("navy", "white", "firebrick3"))(50),
            annotation=metadata_sv, # add the annotation
            show_rownames = F,
            show_colnames=F,
            cutree_cols = 3,
            clustering_method = "ward.D2"
            )
```




```
# Now let's make a vulcano plot with the top genes annotated
# First let's create a temporary file with some coloring information
tmp <- HCR_stat_sva %>%
        mutate(LOG = -log10(FDR_HC),
  Threshold = case_when(Gene %in% Hsap_Spec_sva$Gene ~ "DGE", !(Gene %in% Hsap_Spec_sva
  Direction = case_when(EffSize_HC > 0 ~ "UP",EffSize_HC < 0 ~ "DOWN"))

# First let's create a temporary file with the top genes you wanna visualize
top_labelled <- tbl_df(tmp) %>%
                    filter(Threshold == "DGE") %>%
                    group_by(Direction) %>%
                    top_n(n = 15, wt = abs(EffSize_HC)) # Top by fold change

ggscatter(tmp,
      x = "EffSize_HC",
      y = "LOG",
      color = "Threshold",
      shape = "Direction",
      size=2,
      alpha=0.5,
      palette = c("cyan4", "grey60"))+
    geom_hline(yintercept = 1.3, colour = "red",linetype="dotted",size=1,alpha=0.5) +
    geom_vline(xintercept = 0, colour = "black",linetype="dotted",size=1,alpha=0.5) +
    geom_text_repel(data = top_labelled,
                          mapping = aes(label = Gene),
                          size = 3,
                          box.padding = unit(0.4, "lines"),
```

```
                        point.padding = unit(0.4, "lines")) +
    theme(legend.position="none")+
    xlab("log2(FC)")+
    ylab("-log10(FDR)")
```



3.7 Functional Enrichment

DGE analysis provides genes that are found differentially expressed between two or more groups of samples.

How can we interpret so many genes?

Functional enrichment might help with that! Instead of going through each individual gene to have some clues about what kind of biological function the gene has, we can apply enrichment analyses of functional terms that appear associated to the given set of differentially expressed genes more often than expected by chance. The functional terms usually are associated to multiple genes (e.g. synaptic transmission). So genes can be clustered together and gene ontology analysis helps to quickly find out systematic changes that can describe differences between groups of samples.

We can use R for that using some libraries such as **gProfileR** or **GOStats** or **clusterProfiler** or going online with tools such as **ToppGene** (https://toppgene.cchmc.org/).

Today we are going to use quickly clusterProfiler.

Let's staaaart!

```
# We are going to analyze the human specific genes!
head(Hsap_Spec_sva)
##    Gene EffSize_HC   Pval_HC   FDR_HC EffSize_HR   Pval_HR
## 1  ACP2    -0.7700 0.0009982 0.018184    -1.0425 0.0001799
## 2 ACSM1    -1.8683 0.0000475 0.002875    -1.5574 0.0000949
## 3 ADCY6     0.7374 0.0030412 0.034226     0.9758 0.0001342
## 4 ADTRP     2.4842 0.0033051 0.035828     2.3779 0.0031442
## 5 AGBL1    -1.9622 0.0052972 0.047965    -1.5544 0.0014893
## 6 AGFG2     0.6328 0.0001778 0.006311     0.5425 0.0148278
##      FDR_HR EffSize_CR Pval_CR FDR_CR
## 1 0.0012257   -0.14770 0.33414 0.4476
## 2 0.0007821    0.41314 0.09177 0.1664
## 3 0.0010035    0.24530 0.06662 0.1293
## 4 0.0103883   -0.06809 0.91202 0.9405
## 5 0.0058599   -0.04552 0.85263 0.8972
## 6 0.0347538   -0.17914 0.31631 0.4286

# Create a table with Gene Symbol translated
GenesOfInterest <- bitr(as.character(Hsap_Spec_sva$Gene), fromType = "SYMBOL",
  toType = c("ENSEMBL", "ENTREZID"), OrgDb = org.Hs.eg.db)

# Time for enrichment!  You can try biological processes
# (BP), cellular compontent (CC), and/or molecular function
# (MF)
Hsap_GO <- enrichGO(gene = unique(GenesOfInterest$ENTREZID),
  keyType = "ENTREZID", OrgDb = org.Hs.eg.db, ont = "BP", pAdjustMethod = "none",
  pvalueCutoff = 0.05, qvalueCutoff = 1, readable = TRUE)

DT::datatable(as.data.frame(Hsap_GO), options = list(pageLength = 10))
```

```
# Kegg Pathways!
Hsap_KEGG <- enrichKEGG(gene = unique(GenesOfInterest$ENTREZID),
  organism = "hsa", pAdjustMethod = "none", pvalueCutoff = 0.05,
  qvalueCutoff = 1)

DT::datatable(as.data.frame(Hsap_KEGG), options = list(pageLength = 10))
```

3.8 DGE based on DESeq2

Now we are going to touch base with **DESeq2** (https://bioconductor.org/packages/release/bioc/html/DESeq2.html). This method apply a different nor-

malization called Variance Stabilizing Transformation (VST). This method is based on the Negative Binomial distributed counts. It is a variant of the Arc-hyperbolic-sine transformation (asinh). Therefore, the input for DESeq2 is the count matrix and the modeling design.

Let's start then!

```
# We will perform the DESeq analysisn on H vs C with SVA We
# will need the demoHC, the logCPM_HC, and the row counts.

# First we need to filter the expressed genes for H and C
# from the raw counts!  This are the genes considered
# expressed!
genes <- rownames(logCPM_filtered)

# Subset the counts
counts_HC <- exp[rownames(exp) %in% genes, grep("Hsap|PanTro",
  colnames(exp))]

# Now let's run DESeq2
ddsHC <- DESeqDataSetFromMatrix(countData = counts_HC, colData = demoHC,
  design = as.formula(paste("~", paste(c(colnames(demoHC)),
    collapse = "+"))))

ddsHC <- estimateSizeFactors(ddsHC)  # Estimate Size factors for VST
ddsHC <- DESeq(ddsHC, full = design(ddsHC), parallel = TRUE)  # Run DESeq2
deseq_HC <- as.data.frame(results(ddsHC, contrast = c("Species",
  "Hsap", "PanTro"), cooksCutoff = FALSE))

# Have a look to the info!
DT::datatable(deseq_HC, options = list(pageLength = 10))
```

```
# Here for Human vs Macque
counts_HR <- exp[rownames(exp) %in% genes, grep("Hsap|RheMac",
  colnames(exp))]

ddsHR <- DESeqDataSetFromMatrix(countData = counts_HR, colData = demoHR,
  design = as.formula(paste("~", paste(c(colnames(demoHR)),
    collapse = "+"))))

ddsHR <- estimateSizeFactors(ddsHR)  # Estimate Size factors for VST
ddsHR <- DESeq(ddsHR, full = design(ddsHR), parallel = TRUE)  # Run DESeq2
deseq_HR <- as.data.frame(results(ddsHR, contrast = c("Species",
  "Hsap", "RheMac"), cooksCutoff = FALSE))
```

```
DT::datatable(deseq_HR, options = list(pageLength = 10))
```

```
# And Chimp vs Macaque
counts_CR <- exp[rownames(exp) %in% genes, grep("PanTro|RheMac",
  colnames(exp))]

ddsCR <- DESeqDataSetFromMatrix(countData = counts_CR, colData = demoCR,
  design = as.formula(paste("~", paste(c(colnames(demoCR)),
    collapse = "+"))))

ddsCR <- estimateSizeFactors(ddsCR)  # Estimate Size factors for VST
ddsCR <- DESeq(ddsCR, full = design(ddsCR), parallel = TRUE)  # Run DESeq2
deseq_CR <- as.data.frame(results(ddsCR, contrast = c("Species",
  "PanTro", "RheMac"), cooksCutoff = FALSE))

DT::datatable(deseq_CR, options = list(pageLength = 10))
```

3.9 Exercise for Differential Expression Chapter

- Calculate DGE with DESeq with SVs
- Define Species-Specific DGE based on DESeq2 analysis
- Visualize Species-Specific DGE based on DESeq2
- Diagnostic analysis and functional interpretation of DESeq2 results