MGI DNBSeq-G400
RNASeq sensitivity test

Stéphane Plaisance [VIB - Nucleomics Core, nucleomics@vib.be]

March 4th, 2020 - version 1.0

Contents

last edits: Thu Mar 05, 2020

Introduction

Evaluate two RNA library samples and their in-vitro mix after sequencing on the MGI device and RNASeq mapping to the human genome hg38 build using STAR *(v2.7.3a)*

REM: the various in-silico mix produced in this report are made from the STAR gene counts obtained from the full read-data and not from mapping read-subsets in different ratios. the later method would be more strict but also more resource demanding and working from the full counts was preferred here.

REM: In this work, we do not assess differential expression of genes and only want to see if in-vitro count mixtures from the two RNA samples leads to a correct assessment of gene expression as compared to the counts obtained from either sample alone.

We chose to use TPM counts rather than RPKM or RNASeq normalized counts used in the regular Nucleomics Core analysis pipeline. TPM are equalized for all samples to report a sum of all counts of 1 within each sample and are therefore ideal to produce in-silico mixtures as done below. For more information, please view the following video: **RPKM, FPKM and TPM, clearly explained** (Starmer, n.d.)

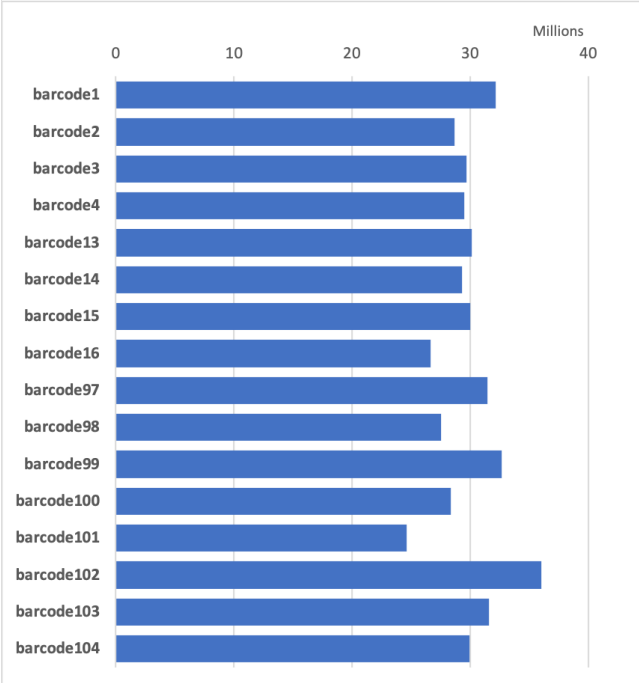
Method

Libraries and sequencing were performed by Kizi Coeck (Nucleomics Core) during the MGI training week (Feb-2020) and are not discussed here.

The Lane (L02) was loaded with a mixture of 16 final barcodes samples corresponding to respectively:

barcode	RNA sample	comment
1	HumanRef	100% HumanRef
2	HumanRef	100% HumanRef
3	HumanRef	100% HumanRef
4	HumanRef	100% HumanRef
13	HumanBrain	100% HumanBrain
14	HumanBrain	100% HumanBrain
15	HumanBrain	100% HumanBrain
16	HumanBrain	100% HumanBrain
97	R75B25	75% HumanRef + 25% HumanBrain
98	R75B25	75% HumanRef + 25% HumanBrain
99	R75B25	75% HumanRef + 25% HumanBrain
100	R75B25	75% HumanRef + 25% HumanBrain
101	R25B75	25% HumanRef + 75% HumanBrain
102	R25B75	25% HumanRef + 75% HumanBrain
103	R25B75	25% HumanRef + 75% HumanBrain
104	R25B75	25% HumanRef + 75% HumanBrain

After sequencing and demultiplexing (MGI pipeline), the QC report shows quasi equimolar distribution of the expected barcodes except for #101 and #102.



FastQC analysis of the reads

A global analysis was done on the full read batch from L02 and key results are shown next

Barcode	Sequence	Count
barcode1	ATCGGACCTA	32176409
barcode2	GATTCCGTCC	28714131
barcode3	CGGCAGTAAG	29691743
barcode4	TCAATTAGGT	29478861
barcode13	CGGATTGCCG	30146199
barcode14	GAATCCTGAT	29344487
barcode15	TCTGGAATGA	30035536
barcode16	ATCCAGCATC	26650272
barcode97	AAGAATACCT	31470122
barcode98	GTTGCATTCG	27561301
barcode99	CGCCGTTGAA	32671261
barcode100	TTCCGCCGAG	28349287
barcode101	CCATTACCGT	24658302
barcode102	ACGTCGGATC	36050304
barcode103	TGTATCGTGA	31612417
barcode104	GAAGAGAATC	29951579

FastQC Report

Fri 14 Feb 2020
stdin:V300043186_L02_1

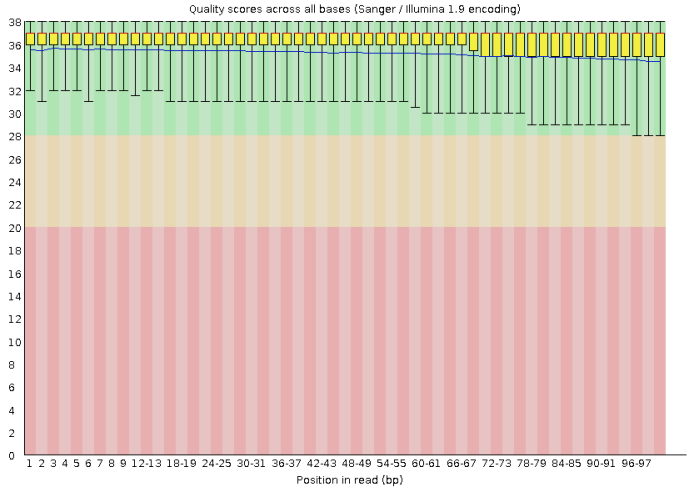
Summary

- Basic Statistics
- Per base sequence quality
- Per sequence quality scores
- Per base sequence content
- Per sequence GC content
- Per base N content
- Sequence Length Distribution
- Sequence Duplication Levels
- Overrepresented sequences
- Adapter Content
- Kmer Content

Basic Statistics

Measure	Value
Filename	V300043186_L02_1
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	490106410
Sequences flagged as poor quality	0
Sequence length	100
%GC	48

Per base sequence quality



Produced by FastQC (version 0.11.7)

FastQC Report

Fri 14 Feb 2020
stdin:V300043186_L02_2

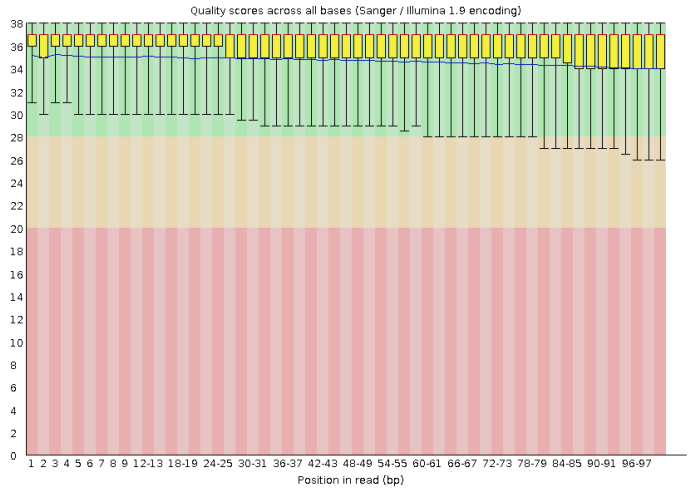
Summary

- Basic Statistics
- Per base sequence quality
- Per sequence quality scores
- Per base sequence content
- Per sequence GC content
- Per base N content
- Sequence Length Distribution
- Sequence Duplication Levels
- Overrepresented sequences
- Adapter Content
- Kmer Content

Basic Statistics

Measure	Value
Filename	V300043186_L02_2
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	490106410
Sequences flagged as poor quality	0
Sequence length	100
%GC	48

Per base sequence quality



Produced by FastQC (version 0.11.7)

STAR analysis

The classical STAR alignment method was applied based on info from the Nucleomics Core pipeline

```
    --outFilterMismatchNmax 10 \
    --outFilterMismatchNoverLmax 0.3 \
    --alignSJDBoverhangMin 3 \
    --alignSJoverhangMin 5 \
    --alignIntronMin 21 \
    --alignIntronMax 500000 \
    --outFilterMultimapNmax 10 \
    --outSJfilterOverhangMin 12 30 30 30 \
    --outWigType None \
    --outSAMprimaryFlag OneBestScore
```

And some additions from the STAR tutorial pages.

- sorted BAM output for PASS-2
- STAR counts per gene for PASS-2

In short, the reads were first mapped against the human reference genome hg38 to identify splice sites and the collection of obtained splice events were stored.

A second STAR alignment pass was then performed that takes into account the merge splice event database from pass-1 to optimize sequence alignments. Besides the usual NC settings, a few additional settings were added to directly produce gene counts (not using third party software as this is done for NC RNASeq analysis).

The full STAR bash script is shown next

```
#!/bin/env bash

# run_StarAlign2pass.sh
# takes all fastq_files from the local 'reads' folder
# align reads to a reference
# perform two passes:
# first pass to add existing junctions to the reference index
# second pass to align to the extended reference index
#
# Stéphane Plaisance - VIB-BITS - Feb-27-2020 v1.1
# including NC stringency options
#   but keeping other defaults as on STAR forum

version="1.1, 2020_03_04"

#####################################
# SERVER PATH and settings          #
# edit to match your own environment #
#####################################

workdir=${1:-"/data2/NC_projects/DNBSEQG400_validation/DNBSEQG400_eval3"}

cd ${workdir}

thr=84
readlen=100
idxlen=$((${readlen}-1))
refindex=$STAR_INDEXES/GRCh38.99
reffolder=$BIODATA/references
refgtf=${reffolder}/Homo_sapiens.GRCh38.99/Homo_sapiens.GRCh38.99.chr.gtf
reffasta=${reffolder}/Homo_sapiens.GRCh38.99/Homo_sapiens.GRCh38.dna.primary_assembly.fa

#####################################
# Nucleomics STAR optional settings #
#####################################

read -r -d '' STAR_OPTIONS <<'EOF'
    --outFilterMismatchNmax 10 \
    --outFilterMismatchNoverLmax 0.3 \
```

```
    --alignSJDBoverhangMin 3 \
    --alignSJoverhangMin 5 \
    --alignIntronMin 21 \
    --alignIntronMax 500000 \
    --outFilterMultimapNmax 10 \
    --outSJfilterOverhangMin 12 30 30 30 \
    --outWigType None \
    --outSAMprimaryFlag OneBestScore
EOF

#####################################
# build STAR index if does not exist #
#####################################

if [ -d "${refindex}" ]; then

echo "# STAR index already present, passing"

else

mkdir -p ${refindex}

cmd="STAR \
    --runMode genomeGenerate \
    --runThreadN ${thr} \
    --genomeDir ${refindex} \
    --genomeFastaFiles ${reffasta} \
    --sjdbGTFfile ${refgtf} \
    --sjdbOverhang ${idxlen}"

echo "# ${cmd}"
eval ${cmd}

fi


#################################################################
# first pass on all samples to collect all possible junctions
#################################################################

mkdir -p ${workdir}/STAR_mappings_PASS-1

for reads1 in ${workdir}/reads/*_1.fq.gz; do
    samplename=$(basename ${reads1/%1.f*.gz})
    reads2=${reads1/_1.fq.gz/_2.fq.gz}

    outpfx=${workdir}/STAR_mappings_PASS-1/${samplename}

    cmd="STAR \
        --runMode alignReads \
        --genomeDir ${refindex} \
        --genomeLoad LoadAndKeep \
        --readFilesCommand zcat \
        --readFilesIn ${reads1} ${reads2} \
        --outSAMtype None \
        --runThreadN ${thr} \
        --outFileNamePrefix ${outpfx}\
        ${STAR_OPTIONS}"

    echo "# first PASS alignment for $(basename ${reads1}) $(basename ${reads2})"
    echo "# ${cmd}"
    eval ${cmd}
done

# unload genome
cmd="STAR \
    --genomeDir ${refindex} \
    --genomeLoad Remove \
    --outSAMtype None \
    --outFileNamePrefix /dev/null/"
```

```
echo "# unloading the reference index"
echo "# ${cmd}"
eval ${cmd}

##############################
# merge and filter junctions
##############################

# 1. Filter out the junctions on chrM, those are most likely to be false.
# 2. Filter out non-canonical junctions (column5 == 0).
# 3. Filter out junctions supported by multimappers only (column7==0)
# 4. Filter out junctions supported by too few reads (e.g. column7<=2)

cmd="cat ${workdir}/STAR_mappings_PASS-1/*_SJ.out.tab \
    | awk 'BEGIN {OFS=\"\\t\"; strChar[0]=\".\"; strChar[1]=\"+\"; strChar[2]=\"-\";}
      {if((\$1!=\"M\") && (\$5>0) && (\$7>2)){print \$1,\$2,\$3,strChar[\$4]}}' \
    | sort -k 1V,1 -k 2n,2 -k 3n,3 \
    | uniq > ${workdir}/SJ.out.all.tab"

echo "# merging and filtering all junction files"
echo "# ${cmd}"
eval ${cmd}

#################################
# create new index with junctions
#################################

mkdir -p ${workdir}/SJ_index
outpfx=${workdir}/SJ_index/

cmd="STAR \
    --runMode genomeGenerate \
    --genomeDir ${workdir}/SJ_index \
    --genomeFastaFiles ${reffasta} \
    --sjdbGTFfile ${refgtf} \
    --sjdbFileChrStartEnd ${workdir}/SJ.out.all.tab \
    --runThreadN ${thr} \
    --sjdbOverhang ${idxlen} \
    --outFileNamePrefix ${outpfx}"

echo "# creating junction-aware reference index"
echo "# ${cmd}"
eval ${cmd}

###############################################
# second pass on all samples and counting genes
###############################################

mkdir -p ${workdir}/STAR_mappings_PASS-2

for reads1 in reads/*_1.fq.gz; do
    samplename=$(basename ${reads1/%1.f*.gz})
    reads2=${reads1/_1.fq.gz/_2.fq.gz}

    outpfx=${workdir}/STAR_mappings_PASS-2/${samplename}

    cmd="STAR \
        --runMode alignReads \
        --runThreadN ${thr} \
        --genomeDir ${workdir}/SJ_index \
        --readFilesCommand zcat \
        --readFilesIn ${reads1} ${reads2} \
        --outFileNamePrefix ${outpfx} \
        --sjdbFileChrStartEnd ${workdir}/SJ.out.all.tab \
        --outFilterType BySJout \
        --outSAMtype BAM SortedByCoordinate \
        --outSAMattributes Standard \
        --outSAMunmapped Within \
        --quantMode GeneCounts \
```

```
        ${STAR_OPTIONS}"

    echo "# second PASS alignment for $(basename ${reads1}) $(basename ${reads2})"
    echo "# ${cmd}"
    eval ${cmd}

done

# cleanup
rm -rf _STARtmp
```

R analysis of STAR counts

The raw STAR counts from the 16 separate STAR analyses were merged to a single large table (60617 rows x 16 columns)

Merging of the STAR raw count files

We converted the STAR raw counts to "transcripts per million" (TPM) for each of the 16 samples, then built expected counts in-silico for various ratio of the two samples and finally compared these to the actual counts obtained for the two in-vitro ration (namely **75% HumanRef + 25% HumanBrain** *[bc97-bc100]* and the reverse mix **25% HumanBrain+75% HumanRef** *[bc101-bc104]*).

The full table of results was used to plot PCA and hierarchical clustering figures shown below.

```
####################################
# Import and merge STAR count files #
####################################

# path to the STAR *ReadsPerGene.out.tab files
star.counts.path <- "/data2/NC_projects/DNBSEQG400_validation/DNBSEQG400_eval3/STAR_mappings_PASS-2"
gene_count_file="gene_counts.csv"

# create if not yet saved locally
if(!file.exists(gene_count_file)){
# read and merge local count files created by STAR
number <- 2 ;   #(could be 2,3, or 4 depending on read types)
ff <- list.files( path = star.counts.path, pattern = "*ReadsPerGene.out.tab$", full.names = TRUE )
counts.files <- lapply( ff, read.table, skip = 4 )
gene_counts <- as.data.frame( sapply( counts.files, function(x) x[ , number ] ) )
ff <- basename(ff)
ff <- gsub( "_ReadsPerGene.out.tab", "", ff )

colnames(gene_counts) <- ff
row.names(gene_counts) <- counts.files[[1]]$V1

# reorder columns
gene_counts <- gene_counts[, naturalsort(colnames(gene_counts))]
gene_counts <- cbind( 'ensembl_gene_id'=row.names(gene_counts), gene_counts)
row.names(gene_counts) <- NULL
write_delim(gene_counts, delim = ",", path = "gene_counts.csv")
} else {
  gene_counts <- read_csv("gene_counts.csv")
}

##############################################
# get gene sizes from BioMART for all EnsGID #
##############################################

gene_info_file="gene_info.csv"

# create if not yet saved locally (long to download)
if(!file.exists(gene_info_file)){
human <- useMart("ensembl", dataset="hsapiens_gene_ensembl")

gene_info=getBM(attributes=c("ensembl_gene_id","start_position","end_position"),
                filters="ensembl_gene_id",
                values=unique(rownames(all_counts)),
                mart=human)

# compute gene length
gene_info$size=gene_info$end_position - gene_info$start_position
gene_info <- gene_info[,c(1,4)]

write_delim(gene_info, delim = ",", path = "gene_info.csv")
} else {
```

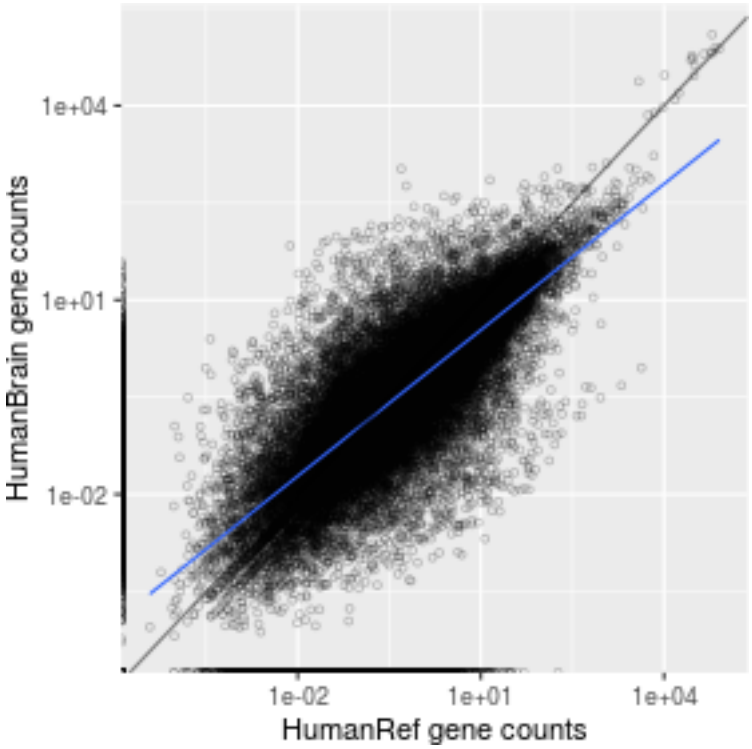
```
  gene_info <- read_csv("gene_info.csv")
}

#####################################################
# normalize data by length then per million => TPM #
#####################################################

# insert gene length as column #2
data <- merge(gene_info, gene_counts, by='ensembl_gene_id')

# normalize each row by gene size
data[, -(1:2)] <- sweep(data[, -(1:2)], 1, data[, 2], "/")

# divide all by column total and multiply by 1E+6 (per million)
data[, -(1:2)] <- sweep(data[, -(1:2)], 2, colSums(data[, -(1:2)])/1E+6, "/")

# also prepare variance-filtered table for PCA1
data <- cbind('ensembl_gene_id' = data$ensembl_gene_id, 'variance' = apply(data[,-(1:2)],1,var), data[,-(1:2)])
write_delim(data, delim = ",", path = "gene_counts_data.csv")

# subset variance greater than 0.1
f.data <- subset(data, data$variance>=0.1)
write_delim(f.data, delim = ",", path = "gene_counts_data_filtered.csv")

# average per group
Ref.iv=rowMeans(data[, 3:6], na.rm=TRUE)
Brain.iv=rowMeans(data[, 7:10], na.rm=TRUE)
R75B25.iv=rowMeans(data[, 11:14], na.rm=TRUE)
R25B75.iv=rowMeans(data[, 15:18], na.rm=TRUE)

data2 <- data.frame( data$ensembl_gene_id, Ref.iv, Brain.iv, R75B25.iv, R25B75.iv )

# re-normalize per million
data2[, -1] <-  sweep(data2[, -1], 2, colSums(data2[, -1])/1E+6, "/")

# add in-silico mix ratios every 5%
steps <- seq(5, 95, by=5)
for (mixH in steps) {
  mixB <- 100-mixH
  title <- paste("R", mixH, "B", mixB, ".is", sep="")
  data2[,title] <- (( data2$Ref.iv*mixH ) + ( data2$Brain.iv*mixB )) / 100
  }

# re-normalize per million
data2[, -1] <-  sweep(data2[, -1], 2, colSums(data2[, -1])/1E+6, "/")

# add rowwise variance
data3 <- cbind( 'ensembl_gene_id'=data2$data.ensembl_gene_id, 'variance'=apply(data2[,-1],1,var), data2[,-1] )
write_delim(data3, delim = ",", path = "gene_counts_data3.csv")

# subset variance greater than 0.1
f.data3 <- subset(data3, data3$variance>=0.1)
write_delim(f.data3, delim = ",", path = "gene_counts_data3_filtered.csv")
```

Data visualization

Plotting the scalled expression counts (TPM) of all genes in both HumanRef and Human brain as a scatter plot gives the following expected result. The black line shows the theoretical diagonal while the blur line represents the data average (loess). The data was not normalized as for a regular RNASeq analysis and should therefore be taken as-is.

```
# plot pairwise
max <- max(c(data2$Ref.iv, data2$Brain.iv))
ggplot(data2, aes(x=Ref.iv, Brain.iv)) +
  geom_point(size=1, shape=21, alpha=0.25) +
  geom_smooth(method=lm, se=FALSE, fullrange=TRUE, size=0.5) +
  xlim(0, max) +
```

```
  ylim(0, max) +
  scale_x_continuous(trans='log10') +
  scale_y_continuous(trans='log10') +
  xlab("HumanRef gene counts") +
  ylab("HumanBrain gene counts") +
  geom_abline(slope=1, intercept=0, size=0.5, alpha=0.5)
```



The following plot reports for each gene, the fraction of the max-variance across all genes (y-axis) as a function of the ratio of [the mean gene count in the HumanBrain] / [the mean count in the HumanRef] (x-axis; both axis are log-scaled). The bar-shaped group of dots on the left side are HumanRef-specific genes while the bar-shaped group on the right side are HumanBrain-specific genes (these weird shapes come from the log-transformation where one of the ratio value is very small). Genes shown in the center part of the plot are expressed in both samples but show more expression in the HumanRef (left half) or in the HumanBrain (rigt half).

```
# plot scatter with differential expression
maxvar <- max(data3$variance)
noexpcor=1e-5
de <- data.frame(ratio=(data3$Brain.iv+noexpcor)/(data3$Ref.iv+noexpcor), variancepc=100*data3$variance/maxvar, group=ifelse(log
ggplot(de, aes(x=ratio, y=variancepc, color=group)) +
  geom_point(size=1, shape=20, alpha=0.5) +
  scale_x_continuous(trans='log10', labels = function(x) log(x,10), limits = c(1e-5, 1e5)) +
  scale_y_continuous(trans='log10', labels = function(x) format(x, scientific = TRUE), limits = c(1e-12, 1e2)) +
  xlab("HumanBrain/Humanref ratio") +
  ylab("Percent of max variance") +
  guides(color = guide_legend(override.aes = list(size=5))) +
  scale_colour_discrete(na.translate = F)
```

PCA analysis

The normalized counts were further filtered to remove low-variance rows (variance<0.1 - arbitrary choice) where all samples have similar expression values as these rows do not (or much less) contribute to the PCA analysis or clustering in the first place.

PCA plots for the actual sequencing samples

The variance filtering retained 22407 rows out of 60617 EnsEMBL GeneID rows.

The normalized and filtered data is used to compute PCA as seen next.

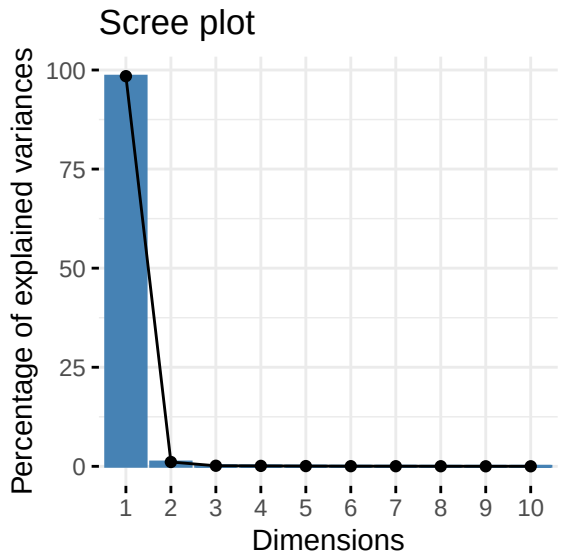
In the first PCA, only STAR counts from actual read data are used to show the variance between all 16 barcode sets.

```
# PCA

res.pca1 <- prcomp(t(f.data[, -(1:2)]), scale = FALSE)
summary(res.pca1)
```

```
Importance of components:
                            PC1       PC2       PC3       PC4       PC5
Standard deviation     3.587e+04 3.782e+03 1.381e+03 1.212e+03 950.84479
Proportion of Variance 9.844e-01 1.094e-02 1.460e-03 1.120e-03   0.00069
Cumulative Proportion  9.844e-01 9.953e-01 9.968e-01 9.979e-01   0.99860
                            PC6       PC7       PC8       PC9      PC10
Standard deviation     733.52666 672.59420 551.41993 430.48616 418.35834
Proportion of Variance   0.00041   0.00035   0.00023   0.00014   0.00013
Cumulative Proportion    0.99901   0.99936   0.99959   0.99973   0.99986
                           PC11      PC12      PC13      PC14      PC15
Standard deviation     282.58233 193.74108 164.40021 151.07585 104.68951
Proportion of Variance   0.00006   0.00003   0.00002   0.00002   0.00001
Cumulative Proportion    0.99992   0.99995   0.99997   0.99999   1.00000
                           PC16
Standard deviation     2.761e-12
Proportion of Variance 0.000e+00
```

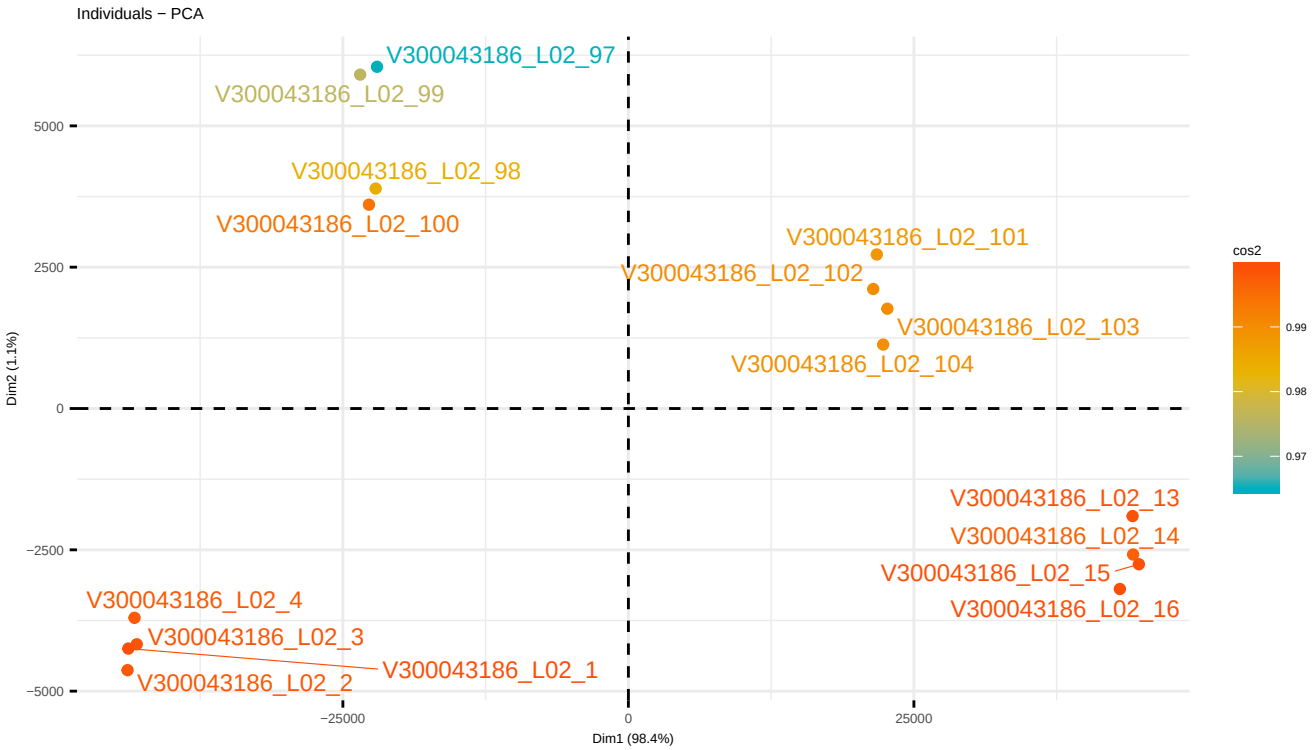
```
Cumulative Proportion  1.000e+00
```

```
fviz_eig(res.pca1)
```



More than 98% of the variance is explained by the first principal component.

```
fviz_pca_ind(res.pca1, col.ind = "cos2", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel = TRUE, labelsize = 3) +
theme(text = element_text(size = 5), axis.title = element_text(size = 5), axis.text = element_text(size = 5))
```



The groups clearly align on top of each other (dim1) based on their main RNA species and the four groups are distinct based on dim1 + dim2.

PCA plots for the means of sequencing groups and in-silico ratios

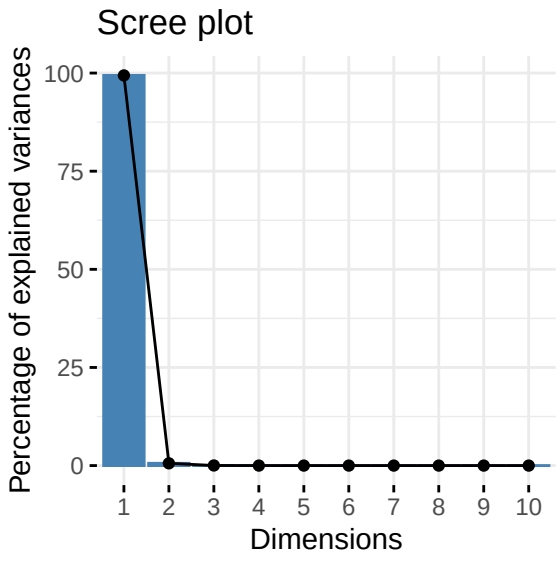
This plot shows the distribution of the 4 group count means together with a range of in-silico ratios

The variance filtering retained 15830 rows out of 60617 EnsEMBL GeneID rows.

```
# PCA

res.pca2 <- prcomp(t(f.data3[, -(1:2)]), scale = FALSE)
summary(res.pca2)
```

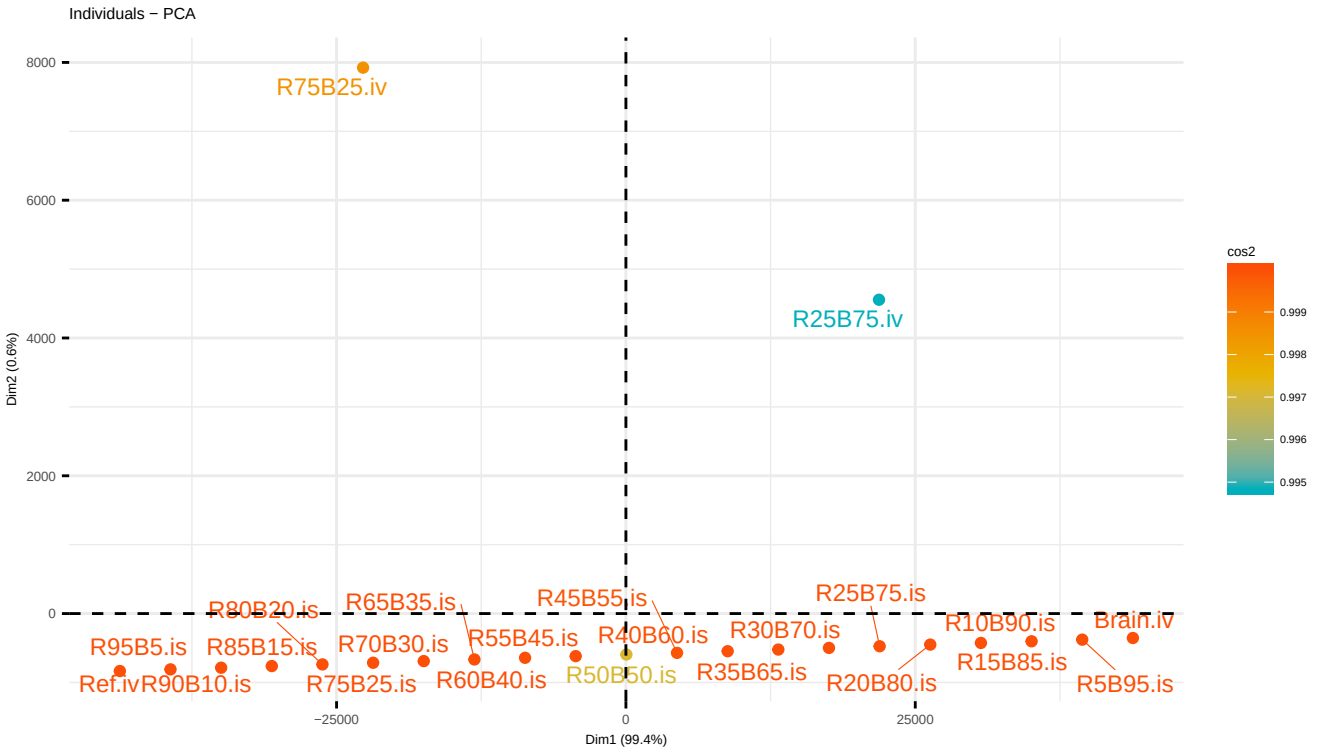
```
Importance of components:
                            PC1       PC2       PC3       PC4       PC5
Standard deviation     26749.109 2.038e+03 409.63989 3.065e-11 8.248e-12
Proportion of Variance     0.994 5.770e-03   0.00023 0.000e+00 0.000e+00
Cumulative Proportion      0.994 9.998e-01   1.00000 1.000e+00 1.000e+00
                            PC6       PC7       PC8       PC9      PC10
Standard deviation     7.363e-12 6.247e-12 4.139e-12 3.54e-12 3.475e-12
Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.00e+00 0.000e+00
Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.00e+00 1.000e+00
                           PC11      PC12      PC13      PC14      PC15
Standard deviation     2.856e-12 2.333e-12 2.046e-12 1.533e-12 1.276e-12
Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
                           PC16      PC17    PC18      PC19      PC20
Standard deviation     7.574e-13 6.157e-13 4.4e-13 3.908e-13 2.808e-13
Proportion of Variance 0.000e+00 0.000e+00 0.0e+00 0.000e+00 0.000e+00
Cumulative Proportion  1.000e+00 1.000e+00 1.0e+00 1.000e+00 1.000e+00
                           PC21      PC22      PC23
Standard deviation     2.546e-13 1.677e-13 1.643e-13
Proportion of Variance 0.000e+00 0.000e+00 0.000e+00
Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00
```

```
fviz_eig(res.pca2)
```



More than 99% of the variance is explained by the first principal component.

```
fviz_pca_ind(res.pca2, col.ind = "cos2", gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel = TRUE, labelsize = 3) +
theme(text = element_text(size = 5), axis.title = element_text(size = 5), axis.text = element_text(size = 5))
```



The two in-vitro mixes do not group with their in-silico counterpart in second PCA but they are nicely projecting in the first PCA. the second dimention represents less than 1% of the total variance and what seems a large distance between the lower and upper points in dim2 is infact minimal expressed in underlying variance. the four in-vitro groups are distributed as in the first PCA plot (one per quadrant).

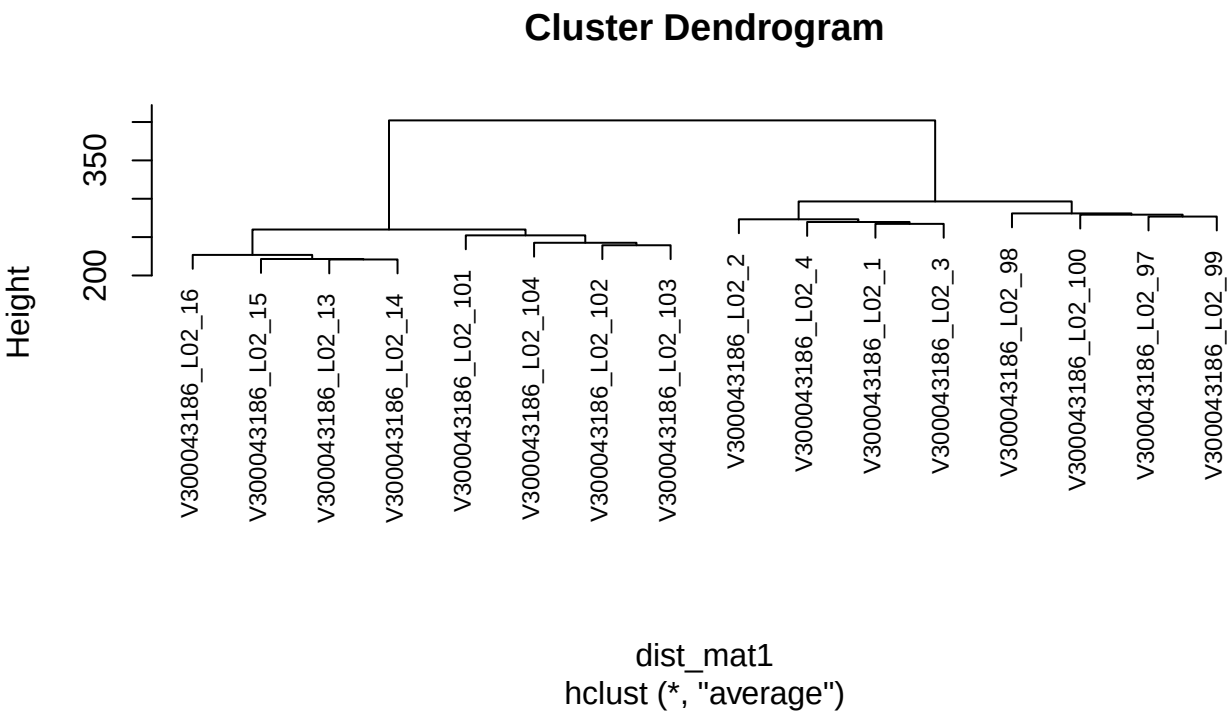
Hierarchical Clustering analysis

The same data can also be used to plot hierarchical clustering results for the 16 sequenced samples and build a tree.

The four groups appear in the order:

- 100% HumanBrain *[bc13..bc16]*
- 75% HUmanBrain 25% HUmanRef *[bc101..bc104]*
- 100% HumanRef *[bc1..bc4]*
- 75% HUmanRef 25% HUmanBrain *[bc97..bc100]*

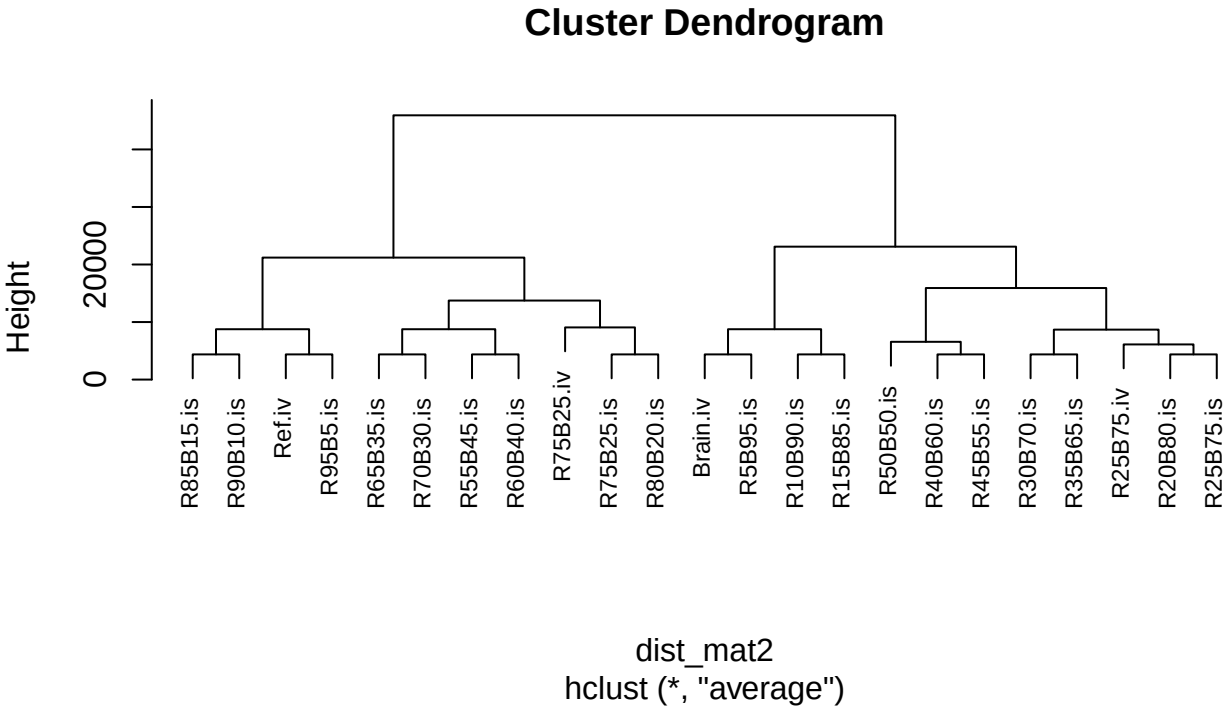
```
# scaled and centered data
dist_mat1 <- dist(scale(t(f.data[, -(1:2)]), center = T, scale = T))
hclust_avg1 <- hclust(dist_mat1, method = 'average')
plot(hclust_avg1, cex=0.75)
```



The four samples of each groups nicely cluster together. the mixed-samples with 75% Human Ref are close to the HumanRef samples (idem for the two Brain groups).

the result of comparing the four sample means to the in-silico ratios is shown next

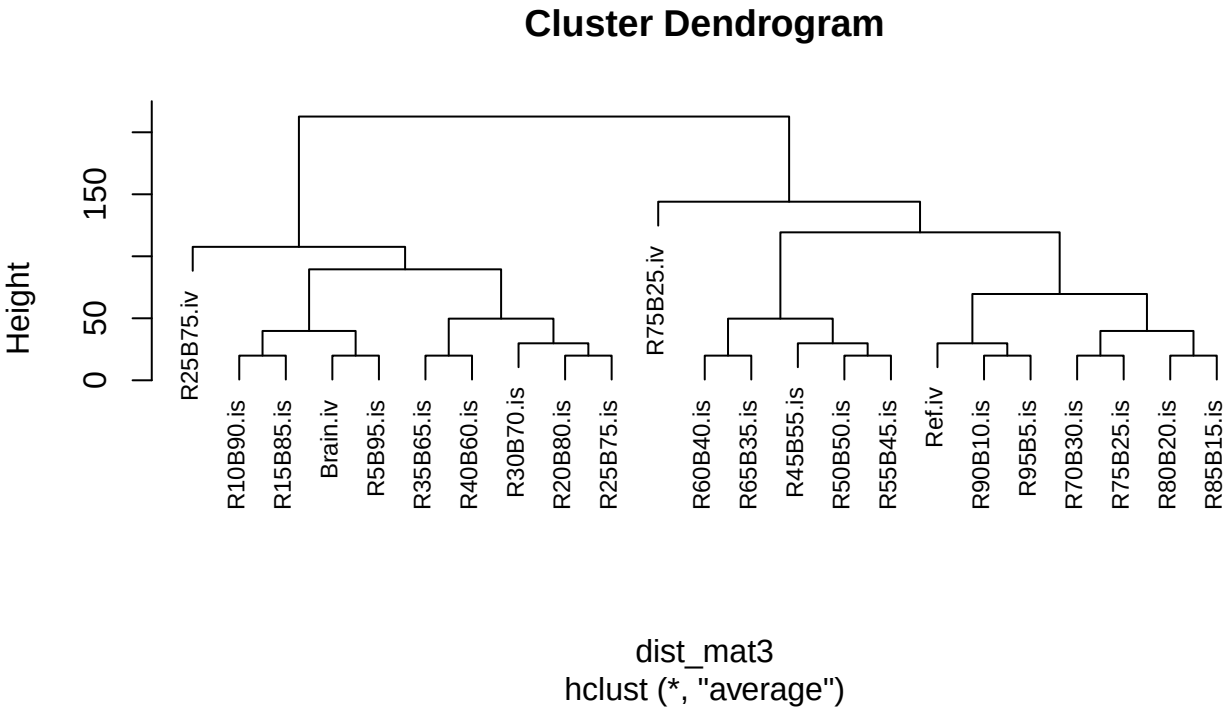
```
# HC

# as is
dist_mat2 <- dist(t(f.data3[, -(1:2)]), method = 'euclidean')
hclust_avg2 <- hclust(dist_mat2, method = 'average')
plot(hclust_avg2, cex=0.75)
```



As expected, the two pure in-vitro samples cluster with the closest in-silico ratios *(95%)*, and each in-vitro mix nicely clusters with the closest in-silico ratio's *(75% and 80%)*.

Below are similar plot after scaling and centering of the data (in case experts would prefer this approach).

```
# scaled and centered data
dist_mat3 <- dist(scale(t(f.data3[, -(1:2)]), center = T, scale = T))
hclust_avg3 <- hclust(dist_mat3, method = 'average')
plot(hclust_avg3, cex=0.75)
```



The scaling of the data does not change dramatically the aspect of the tree and the same samples are found at the same relative positions.

Conclusion

We show here that sequencing data obtained by mixing two defined samples reflects the abundance of each sample. This expected results comforts us the fact that the platform is robust enough to detect library complexities of similar nature.

Read barcode demultiplexing is apparently specific and efficient based on the clustering results obtained here and the sensitivity lies in a 5-10% range based on the review of the trees.

last edits: Thu Mar 05, 2020

References

Starmer, Josh. n.d. *RPKM, Fpkm and Tpm, Clearly Explained.* https://statquest.org/2015/07/09/rpkm-fpkm-and-tpm-clearly-explained/.