Supplementary Data 3: Comparison of BASiCS and scran using Zeisel et al (2015) data

Catalina Vallejos, Davide Risso, Antonio Scialdone, Sandrine Dudoit and John Marioni

Contents

Introduction **2**

Data pre-processing **2**

`scran` normalization **3**

`BASiCS` normalization **3**

Creation of `BASiCS_Data` objects . . . 3
Running MCMC chains . . . 6
Convergence of MCMC chains . . . 7
Estimating scaling factors and denoised counts . . . 9

Estimates comparison **10**

Highly variable genes detection comparison **13**

Subpopulation 1: astrocytes-ependymal . . . 13
Subpopulation 2: oligodendrocytes . . . 15
Subpopulation 3: microglia . . . 17

References **19**

Introduction

This document compares the performance of `BASiCS` (Vallejos, Marioni, and Richardson 2015) and `scran` (Lun, Bach, and Marioni 2016) based on the mouse cortex and hippocampus cells from (Zeisel et al. 2015). This contains 3,005 cells collected with the Fluidigm C1 (microfluidics-based) technology and sequenced at the average depth of 14,000 reads per cell (with UMI). As in Supplementary File 1, in order to consider homogeneous groups of cells, we select three of the subpopulations identified in the original publication ("astrocytes_ependymal", "oligodendrocytes" and "microglia") and analyse them separately.

In order to obtain meaningful comparisons between the methods, the estimated scaling factors are scaled so that their average is equal to 1.

Data pre-processing

After loading the data, the same gene filter adopted in Supplementary File 1 is applied.

```
zei <- read.table("../../datasets/Zeisel2015/expression_mRNA_17-Aug-2014.txt",
                  header=FALSE, stringsAsFactors=FALSE, row.names=1, sep='\t', skip=11)[,-1]
colnames(zei) <- paste0("zei_", seq_len(ncol(zei)))

tmp <- read.table("../../datasets/Zeisel2015/expression_mRNA_17-Aug-2014.txt",
                  nrows=3, skip=7, sep='\t', stringsAsFactors = FALSE)
tmp <- as.matrix(tmp[,-(1:2)])
idx.astro <- which(tmp[2,]=="astrocytes_ependymal")
idx.micro <- which(tmp[2,]=="microglia")
idx.oligo <- which(tmp[2,]=="oligodendrocytes")

save(zei, idx.astro,
     idx.micro, idx.oligo,
     file="../../datasets/Zeisel2015/zeisel.RData")
```

```
load("../../datasets/Zeisel2015/zeisel.RData")
Cell.Colour<-"lightblue"
```

Three different subpopulations are selected for the analysis:

First, we remove the lowly expressed genes (see "Supplementary file 1" for a description of the filtering):

```
RawCounts1<-zei[,idx.astro]
filter1<-filter.genes(RawCounts1)
Select1<-filter1$Select
RawCountsZeiFilter1 <- RawCounts1[Select1,]
```

```
RawCounts2<-zei[,idx.oligo]
filter2<-filter.genes(RawCounts2)
Select2<-filter2$Select
RawCountsZeiFilter2 <- RawCounts2[Select2,]
```

```
RawCounts3<-zei[,idx.micro]
filter3<-filter.genes(RawCounts3)
Select3<-filter3$Select
RawCountsZeiFilter3 <- RawCounts3[Select3,]
```

A number N=7971, 8973 and 7073 of genes was respectively selected from the three subpopulations. The minimum average expression among the selected genes in each group is equal to 17.57, 9.03 and 21.42 reads per million, respectively.

scran normalization

Firtly, we normalize the data using `scran` (Lun, Bach, and Marioni 2016). Since sample size is small for the microglia subpopulation (98 cells), the `sizes` parameters used for the `scran` normalization must be altered (sample size must be at least twice as the size of the largest pool).

```
NormScran1 <- scran_norm(as.matrix(RawCountsZeiFilter1))
NormScran2 <- scran_norm(as.matrix(RawCountsZeiFilter2))
NormScran3 <- scran_norm(as.matrix(RawCountsZeiFilter3), sizes = c(20, 40))
```

BASiCS normalization

Creation of BASiCS_Data objects

`BASiCS_Data` spike-ins information must be loaded. For each subpopulation of cells, we exclude spike-in genes with zero total counts across all cells.

```
# Loading spike-in counts
Spikes <- read.table("../../datasets/Zeisel2015/expression_spikes_17-Aug-2014.txt",
                     header=FALSE, stringsAsFactors=FALSE, row.names=1, sep='\t', skip=11)[,-1]
colnames(Spikes) <- paste0("zei_", seq_len(ncol(zei)))
Spike_Molecules=fread("../../datasets/Zeisel2015/SilverBulletCTRLConc.txt",
                      select=c("ERCC_ID","molecules_in_each_chamber"))
names(Spike_Molecules); setkey(Spike_Molecules,ERCC_ID)
```

```
## [1] "ERCC_ID"                   "molecules_in_each_chamber"
```

```
RawCountsSpikes1 <- Spikes[,idx.astro]
RawCountsSpikes2 <- Spikes[,idx.oligo]
RawCountsSpikes3 <- Spikes[,idx.micro]

keep1 <- which(rowSums(RawCountsSpikes1) > 0)
keep2 <- which(rowSums(RawCountsSpikes2) > 0)
keep3 <- which(rowSums(RawCountsSpikes3) > 0)

RawCountsSpikes1 <- RawCountsSpikes1[keep1,]
RawCountsSpikes2 <- RawCountsSpikes2[keep2,]
RawCountsSpikes3 <- RawCountsSpikes3[keep3,]

Spike_ID1=data.table("ERCC_ID" = rownames(RawCountsSpikes1),
                     "Aux.ID" = 1:nrow(RawCountsSpikes1))
setkey(Spike_ID1,ERCC_ID)
Spike_ID2=data.table("ERCC_ID" = rownames(RawCountsSpikes2),
                     "Aux.ID" = 1:nrow(RawCountsSpikes2))
setkey(Spike_ID2,ERCC_ID)
Spike_ID3=data.table("ERCC_ID" = rownames(RawCountsSpikes3),
                     "Aux.ID" = 1:nrow(RawCountsSpikes3))
setkey(Spike_ID3,ERCC_ID)

Spike_Molecules1 = merge(Spike_ID1,Spike_Molecules)
Spike_Molecules2 = merge(Spike_ID2,Spike_Molecules)
Spike_Molecules3 = merge(Spike_ID3,Spike_Molecules)
```

```r
SpikeInfo1 = subset(Spike_Molecules1, select = c(ERCC_ID, molecules_in_each_chamber))
SpikeInfo2 = subset(Spike_Molecules2, select = c(ERCC_ID, molecules_in_each_chamber))
SpikeInfo3 = subset(Spike_Molecules3, select = c(ERCC_ID, molecules_in_each_chamber))
```

And the required `BASiCS_Data` objects are created as follows

```r
RawCountsAll1 <- as.matrix(rbind(RawCountsZeiFilter1, RawCountsSpikes1))
RawCountsAll2 <- as.matrix(rbind(RawCountsZeiFilter2, RawCountsSpikes2))
RawCountsAll3 <- as.matrix(rbind(RawCountsZeiFilter3, RawCountsSpikes3))

Data1 = newBASiCS_Data(Counts = RawCountsAll1,
                       Tech = grepl("ERCC", row.names(RawCountsAll1)),
                       SpikeInfo = SpikeInfo1)
```

```
## An object of class BASiCS_Data
##  Dataset contains 8028 genes (7971 biological and 57 technical) and 224 cells.
##  Elements (slots): Counts, Tech, SpikeInput, GeneNames and BatchInfo.
##  The data contains 1 batch.
##
## NOTICE: BASiCS requires a pre-filtered dataset
##     - You must remove poor quality cells before creating the BASiCS data object
##     - We recommend to pre-filter very lowly expressed transcripts before creating the object.
##       Inclusion criteria may vary for each data. For example, remove transcripts
##           - with very low total counts across of all of the samples
##           - that are only expressed in a few cells
##             (by default genes expressed in only 1 cell are not accepted)
##           - with very low total counts across the samples where the transcript is expressed
##
##  BASiCS_Filter can be used for this purpose
```

```r
Data2 = newBASiCS_Data(Counts = RawCountsAll2,
                       Tech = grepl("ERCC", row.names(RawCountsAll2)),
                       SpikeInfo = SpikeInfo2)
```

```
## An object of class BASiCS_Data
##  Dataset contains 9030 genes (8973 biological and 57 technical) and 820 cells.
##  Elements (slots): Counts, Tech, SpikeInput, GeneNames and BatchInfo.
##  The data contains 1 batch.
##
## NOTICE: BASiCS requires a pre-filtered dataset
##     - You must remove poor quality cells before creating the BASiCS data object
##     - We recommend to pre-filter very lowly expressed transcripts before creating the object.
##       Inclusion criteria may vary for each data. For example, remove transcripts
##           - with very low total counts across of all of the samples
##           - that are only expressed in a few cells
##             (by default genes expressed in only 1 cell are not accepted)
##           - with very low total counts across the samples where the transcript is expressed
##
##  BASiCS_Filter can be used for this purpose
```

```r
Data3 = newBASiCS_Data(Counts = RawCountsAll3,
                       Tech = grepl("ERCC", row.names(RawCountsAll3)),
                       SpikeInfo = SpikeInfo3)
```

```
## An object of class BASiCS_Data
##  Dataset contains 7130 genes (7073 biological and 57 technical) and 98 cells.
```

```
##  Elements (slots): Counts, Tech, SpikeInput, GeneNames and BatchInfo.
##  The data contains 1 batch.
##
## NOTICE: BASiCS requires a pre-filtered dataset
##     - You must remove poor quality cells before creating the BASiCS data object
##     - We recommend to pre-filter very lowly expressed transcripts before creating the object.
##       Inclusion criteria may vary for each data. For example, remove transcripts
##           - with very low total counts across of all of the samples
##           - that are only expressed in a few cells
##             (by default genes expressed in only 1 cell are not accepted)
##           - with very low total counts across the samples where the transcript is expressed
##
##  BASiCS_Filter can be used for this purpose
```

Running MCMC chains

The code used to generate MCMC chains is provided below. Chains were run for 40,000 iterations, with a burn-in and thinning period of 20,000 and 20, respectively.

```
chains.path = "../data_analysis/Chains/"
N = 40000; Thin = 20; Burn = 20000
RunName1 <- paste0("Zeisel_astro_",N)
RunName2 <- paste0("Zeisel_oligo_",N)
RunName3 <- paste0("Zeisel_micro_",N)

Chain1 <- BASiCS_MCMC(Data1, N = N, Thin = Thin, Burn = Burn,
                      PriorDelta = "log-normal",
                      PrintProgress = TRUE, StoreChains = TRUE,
                      StoreDir = chains.path, RunName = RunName1)

Chain2 <- BASiCS_MCMC(Data2, N = N, Thin = Thin, Burn = Burn,
                      PriorDelta = "log-normal",
                      PrintProgress = TRUE, StoreChains = TRUE,
                      StoreDir = chains.path, RunName = RunName2)

Chain3 <- BASiCS_MCMC(Data3, N = N, Thin = Thin, Burn = Burn,
                      PriorDelta = "log-normal",
                      PrintProgress = TRUE, StoreChains = TRUE,
                      StoreDir = chains.path, RunName = RunName3)
```

Here, results are reported based on precomputed MCMC chains.

```
Chain1 <- BASiCS_LoadChain(RunName = "Zeisel_astro_40000", StoreDir = chains.path)
```

```
## An object of class BASiCS_Chain
##  1000 MCMC samples.
##  Dataset contains 7971 biological genes and 224 cells (1 batch).
##  Elements (slots): mu, delta, phi, s, nu and theta.
```

```
Chain2 <- BASiCS_LoadChain(RunName = "Zeisel_oligo_40000", StoreDir = chains.path)
```

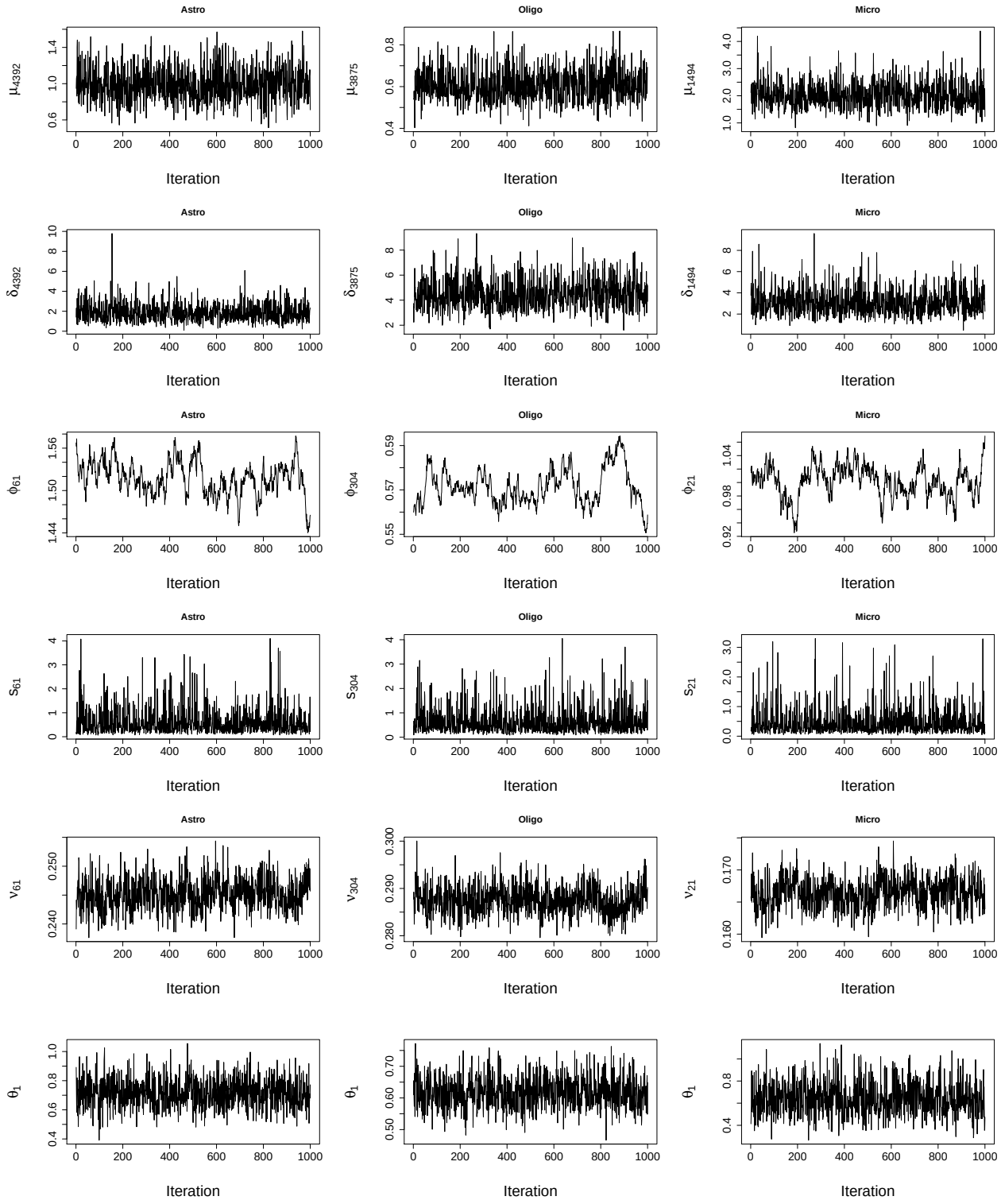
```
## An object of class BASiCS_Chain
##  1000 MCMC samples.
##  Dataset contains 8973 biological genes and 820 cells (1 batch).
##  Elements (slots): mu, delta, phi, s, nu and theta.
```

```
Chain3 <- BASiCS_LoadChain(RunName = "Zeisel_micro_40000", StoreDir = chains.path)
```

```
## An object of class BASiCS_Chain
##  1000 MCMC samples.
##  Dataset contains 7073 biological genes and 98 cells (1 batch).
##  Elements (slots): mu, delta, phi, s, nu and theta.
```

Convergence of MCMC chains

```
par(mgp = c(5,1,0)); par(mar = c(7,9,4,0.5)); par(mfrow = c(6,3))
par(cex.lab = 2, cex.axis = 1.5)
set.seed(20161102); seeds = sample(1:1e6, 6)
set.seed(seeds[1]); gene1 = sample(1:length(Select1), size = 1)
set.seed(seeds[2]); gene2 = sample(1:length(Select2), size = 1)
set.seed(seeds[3]); gene3 = sample(1:length(Select3), size = 1)
set.seed(seeds[4]); cell1 = sample(1:ncol(Data1@Counts), size = 1)
set.seed(seeds[5]); cell2 = sample(1:ncol(Data2@Counts), size = 1)
set.seed(seeds[6]); cell3 = sample(1:ncol(Data3@Counts), size = 1)

plot(Chain1, Param = "mu", Gene = gene1, main = "Astro")
plot(Chain2, Param = "mu", Gene = gene2, main = "Oligo")
plot(Chain3, Param = "mu", Gene = gene3, main = "Micro")

plot(Chain1, Param = "delta", Gene = gene1, main = "Astro")
plot(Chain2, Param = "delta", Gene = gene2, main = "Oligo")
plot(Chain3, Param = "delta", Gene = gene3, main = "Micro")

plot(Chain1, Param = "phi", Cell = cell1, main = "Astro")
plot(Chain2, Param = "phi", Cell = cell2, main = "Oligo")
plot(Chain3, Param = "phi", Cell = cell3, main = "Micro")

plot(Chain1, Param = "s", Cell = cell1, main = "Astro")
plot(Chain2, Param = "s", Cell = cell2, main = "Oligo")
plot(Chain3, Param = "s", Cell = cell3, main = "Micro")

plot(Chain1, Param = "nu", Cell = cell1, main = "Astro")
plot(Chain2, Param = "nu", Cell = cell2, main = "Oligo")
plot(Chain3, Param = "nu", Cell = cell3, main = "Micro")

plot(Chain1, Param = "theta")
plot(Chain2, Param = "theta")
plot(Chain3, Param = "theta")
```


Astro
Oligo
Micro
μ_{4392}
μ_{3875}
μ_{1494}
δ_{4392}
δ_{3875}
δ_{1494}
ϕ_{61}
ϕ_{304}
ϕ_{21}
s_{61}
s_{304}
s_{21}
ν_{61}
ν_{304}
ν_{21}
θ_1
Iteration

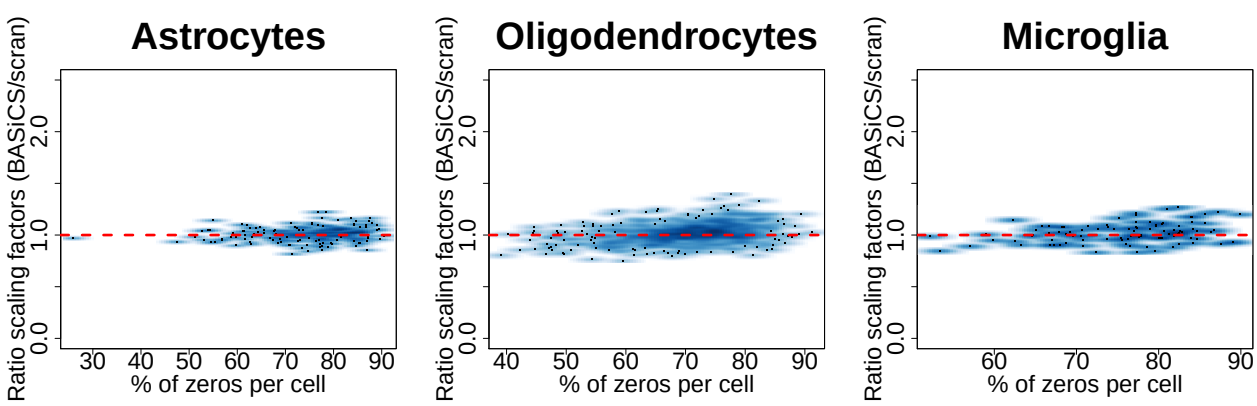
Estimating scaling factors and denoised counts

```
Summary1 <- Summary(Chain1)
Summary2 <- Summary(Chain2)
Summary3 <- Summary(Chain3)

NormBasics1 <- list()
NormBasics1$s <- Summary1@phi[,1] * Summary1@nu[,1]
NormBasics1$s <- length(NormBasics1$s) * NormBasics1$s / sum(NormBasics1$s)
NormBasics1$counts <- t( t(RawCountsZeiFilter1) / NormBasics1$s )

NormBasics2 <- list()
NormBasics2$s <- Summary2@phi[,1] * Summary2@nu[,1]
NormBasics2$s <- length(NormBasics2$s) * NormBasics2$s / sum(NormBasics2$s)
NormBasics2$counts <- t( t(RawCountsZeiFilter2) / NormBasics2$s )

NormBasics3 <- list()
NormBasics3$s <- Summary3@phi[,1] * Summary3@nu[,1]
NormBasics3$s <- length(NormBasics3$s) * NormBasics3$s / sum(NormBasics3$s)
NormBasics3$counts <- t( t(RawCountsZeiFilter3) / NormBasics3$s )
```

Estimates comparison

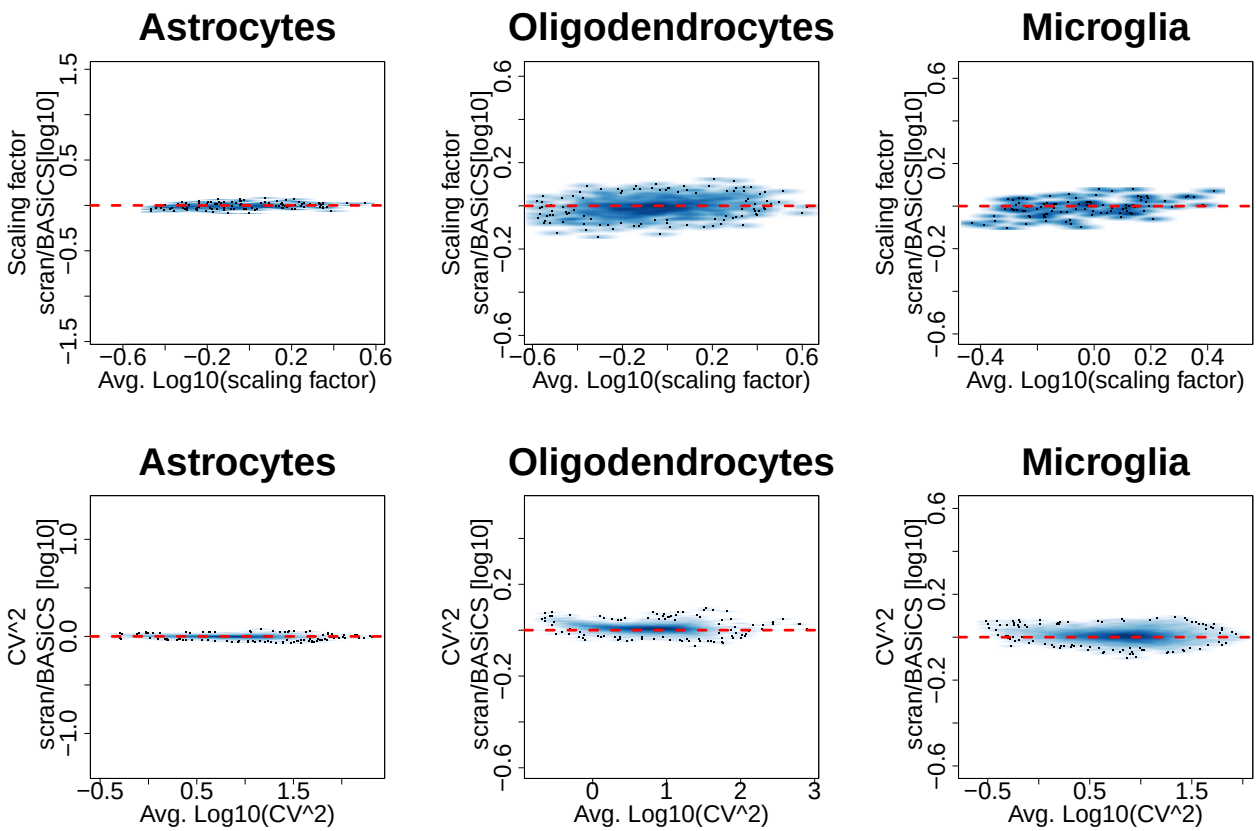
Let us compare the size factors and the corresponding coefficient of variation computed when using `scran` and `BASiCS` normalization. The range limit of these plots have been defined to match the ones displayed in Supplementary File 1 in order to allow a direct comparison.

```
#Count the fractions of zeros in each cell
frac.zeros1<-colMeans(RawCountsZeiFilter1 == 0)
frac.zeros2<-colMeans(RawCountsZeiFilter2 == 0)
frac.zeros3<-colMeans(RawCountsZeiFilter3 == 0)

df1<-data.frame(zeros = frac.zeros1,
                x = NormScran1$s,
                y = NormBasics1$s)
df2<-data.frame(zeros = frac.zeros2,
                x = NormScran2$s,
                y = NormBasics2$s)
df3<-data.frame(zeros = frac.zeros3,
                x = NormScran3$s,
                y = NormBasics3$s)

par(mfrow = c(1,3))
par(cex.lab = 2.5, cex.axis = 2.5)
par(mar = c(7, 6, 6, 2) + 0.1)
plot.sf.ratio.frac.zeros.cata.sc(df1, ylim.max = 2.5, main = "Astrocytes", cex.main = 4)
plot.sf.ratio.frac.zeros.cata.sc(df2, ylim.max = 2.5, main = "Oligodendrocytes", cex.main = 4)
plot.sf.ratio.frac.zeros.cata.sc(df3, ylim.max = 2.5, main = "Microglia", cex.main = 4)
```



```r
CVscran1<-apply(NormScran1$counts, 1, function(x) sd(x)/mean(x))
CVscran2<-apply(NormScran2$counts, 1, function(x) sd(x)/mean(x))
CVscran3<-apply(NormScran3$counts, 1, function(x) sd(x)/mean(x))

CVbasics1<-apply(NormBasics1$counts, 1, function(x) sd(x)/mean(x))
CVbasics2<-apply(NormBasics2$counts, 1, function(x) sd(x)/mean(x))
CVbasics3<-apply(NormBasics3$counts, 1, function(x) sd(x)/mean(x))

df1.1<-log10(data.frame(x=NormScran1$s, y = NormBasics1$s))
df2.1<-log10(data.frame(x=CVscran1^2, y = CVbasics1^2))
df1.2<-log10(data.frame(x=NormScran2$s, y = NormBasics2$s))
df2.2<-log10(data.frame(x=CVscran2^2, y = CVbasics2^2))
df1.3<-log10(data.frame(x=NormScran3$s, y = NormBasics3$s))
df2.3<-log10(data.frame(x=CVscran3^2, y = CVbasics3^2))

# Group 1 lims
xlim1.1 = c(-0.7027151, 0.5896570)
ylim1.1 = c(-1.415949, 1.469485)
xlim2.1 = c(-0.4884052, 2.3309913)
ylim2.1 = c(-1.355370, 1.342332)

# Group 2 lims
xlim1.2 = c(-0.5874167,  0.6248240)
ylim1.2 = c(-0.5930040,  0.6194333)
xlim2.2 = c(-0.7742081,  2.9076748)
ylim2.2 = c(-0.6008204,  0.5408244)

# Group 3 lims
xlim1.3 = c(-0.4394103,  0.5206227)
ylim1.3 = c(-0.6000288,  0.6294210)
xlim2.3 = c(-0.6793962,  1.9912261)
ylim2.3 = c(-0.6095458,  0.6107463)

par(mfrow = c(2, 3))
par(cex.lab = 2.5, cex.axis = 2.5, cex.main = 4)
par(mar = c(7, 10, 6, 2) + 0.1)
smoothScatter(rowMeans(df1.1), df1.1$x - df1.1$y,
              ylab = "Scaling factor\n scran/BASiCS[log10] ",
              xlab = "Avg. Log10(scaling factor)", main = "Astrocytes",
              xlim = xlim1.1, ylim = ylim1.1)
abline(h = 0, lty = 2, lwd = 3, col = "red")
smoothScatter(rowMeans(df1.2), df1.2$x - df1.2$y,
              ylab = "Scaling factor\n scran/BASiCS[log10] ",
              xlab = "Avg. Log10(scaling factor)", main = "Oligodendrocytes",
              xlim = xlim1.2, ylim = ylim1.2)
abline(h = 0, lty = 2, lwd = 3, col = "red")
smoothScatter(rowMeans(df1.3), df1.3$x - df1.3$y,
              ylab = "Scaling factor\n scran/BASiCS[log10] ",
              xlab = "Avg. Log10(scaling factor)", main = "Microglia",
              xlim = xlim1.3, ylim = ylim1.3)
abline(h = 0, lty = 2, lwd = 3, col = "red")

smoothScatter(rowMeans(df2.1), df2.1$x - df2.1$y,
```

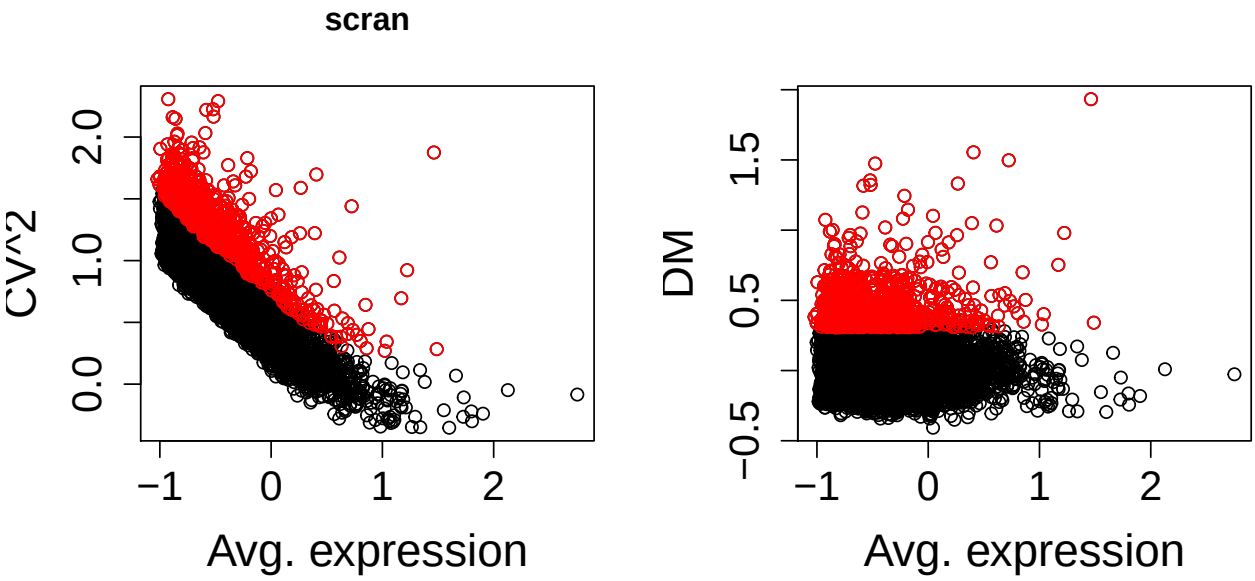
```
                ylab = "CV^2\n scran/BASiCS [log10]",
                xlab = "Avg. Log10(CV^2)", main = "Astrocytes", xlim =
                  xlim2.1, ylim = ylim2.1)
abline(h = 0, lty = 2, lwd = 3, col = "red")
smoothScatter(rowMeans(df2.2), df2.2$x - df2.2$y,
                ylab = "CV^2\n scran/BASiCS [log10]",
                xlab = "Avg. Log10(CV^2)", main = "Oligodendrocytes",
                xlim = xlim2.2, ylim = ylim2.2)
abline(h = 0, lty = 2, lwd = 3, col = "red")
smoothScatter(rowMeans(df2.3), df2.3$x - df2.3$y,
                ylab = "CV^2\n scran/BASiCS [log10]",
                xlab = "Avg. Log10(CV^2)", main = "Microglia",
                xlim = xlim2.3, ylim = ylim2.3)
abline(h = 0, lty = 2, lwd = 3, col = "red")
```



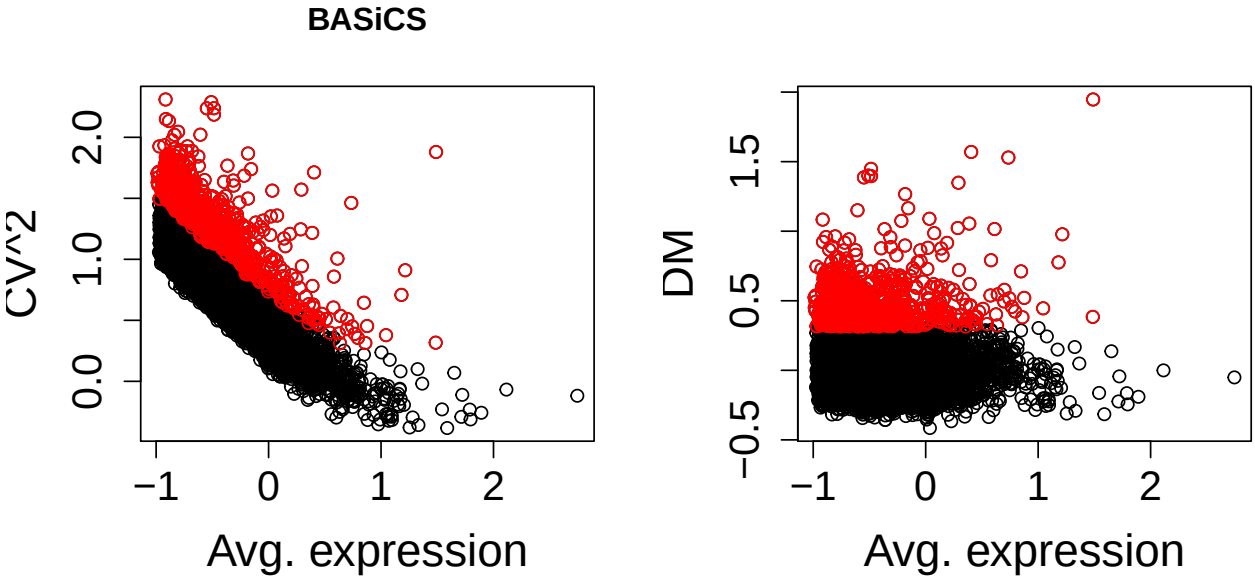
Highly variable genes detection comparison

Subpopulation 1: astrocytes-ependymal

```
par(mfrow = c(1,2))
f<-0.1
n.hvg<-floor(f*length(Select1))
hvg.scran1<-find.high.var.dm(data=NormScran1$counts,n.hvg =n.hvg,
                             plot = TRUE,
                             title = "scran")
```

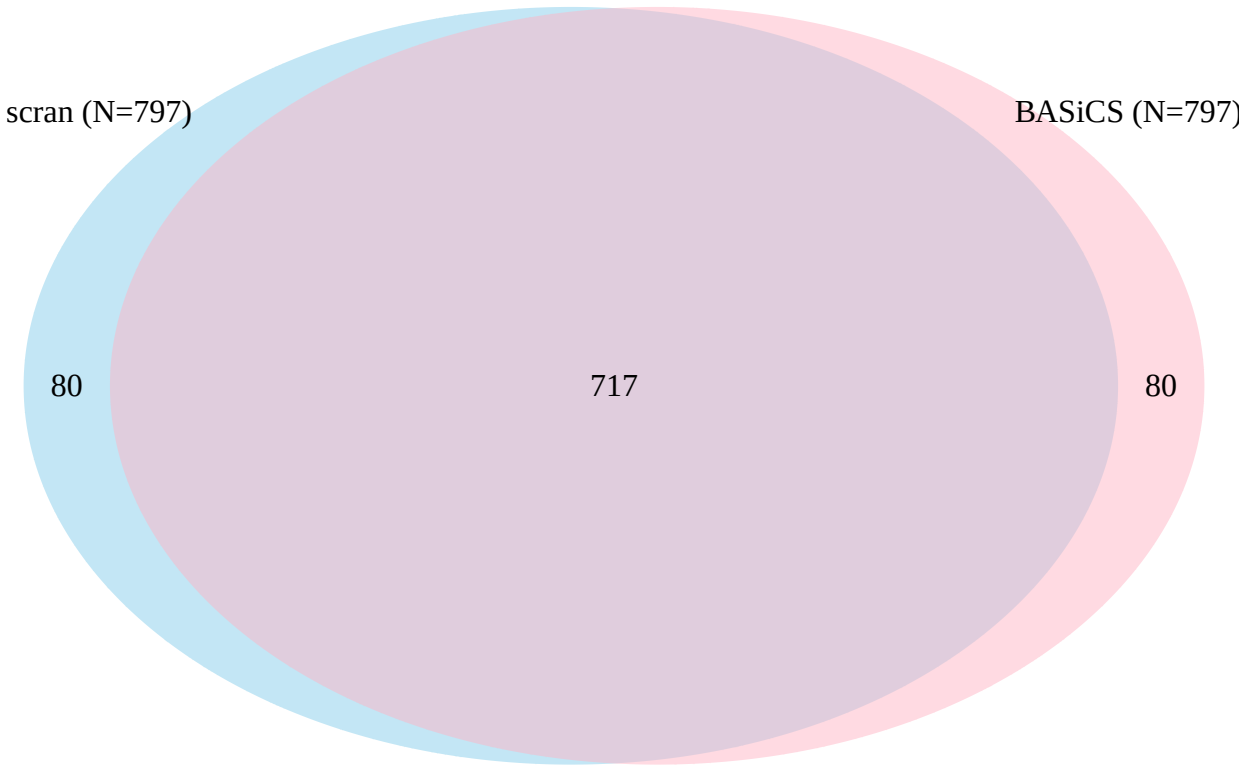



```
hvg.basics1<-find.high.var.dm(data=NormBasics1$counts,n.hvg =n.hvg,
                              plot = TRUE,
                              title = "BASiCS")
```




```
#define a list of highly variable genes
list.hvg1<-list(scran=hvg.scran1$genes.high.var,
```

```
                basics=hvg.basics1$genes.high.var)

grid.newpage()
draw.pairwise.venn(area1 = length(list.hvg1$scran),
                   area2 = length(list.hvg1$basics),
                   cross.area = length(intersect(list.hvg1$scran,list.hvg1$basics)),
                   category = c(paste0("scran (N=",length(list.hvg1$scran), ")"),
                                paste0("BASiCS (N=",length(list.hvg1$basics), ")")),
                   lty = "blank",
                   fill = c("skyblue", "pink1"))
```

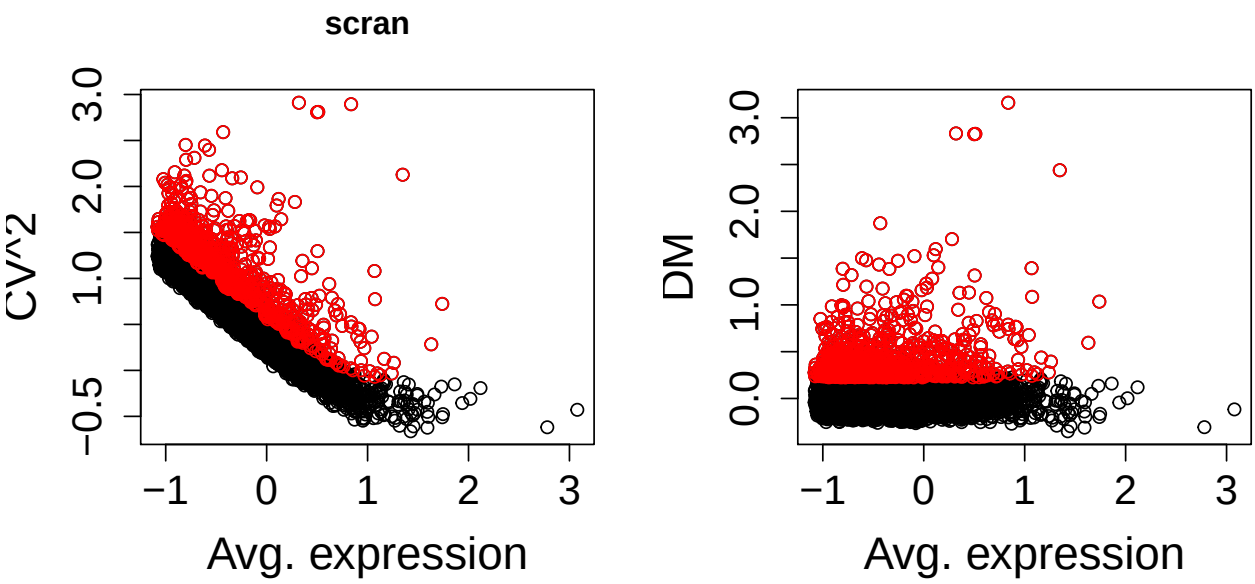



```
## (polygon[GRID.polygon.11], polygon[GRID.polygon.12], polygon[GRID.polygon.13], polygon[GRID.polygon.
```

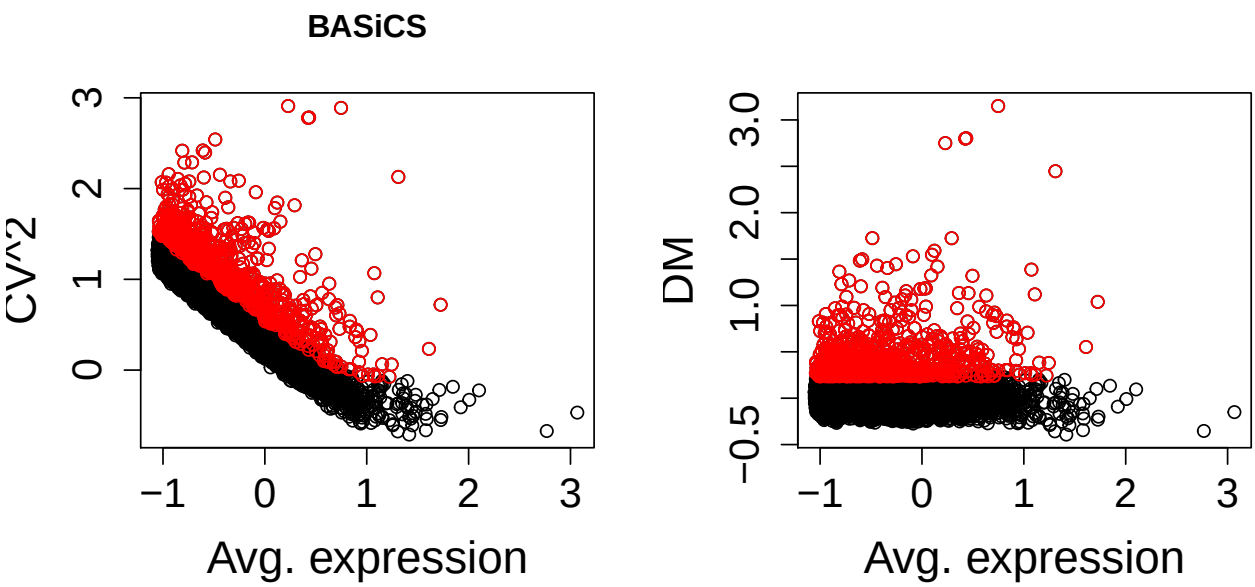
717 out of 797 highly variable genes (90%) are shared across the two normalization methods.

Subpopulation 2: oligodendrocytes

```
par(mfrow = c(1,2))
f<-0.1
n.hvg<-floor(f*length(Select2))
hvg.scran2<-find.high.var.dm(data=NormScran2$counts,n.hvg =n.hvg,
                             plot = TRUE,
                             title = "scran")
```

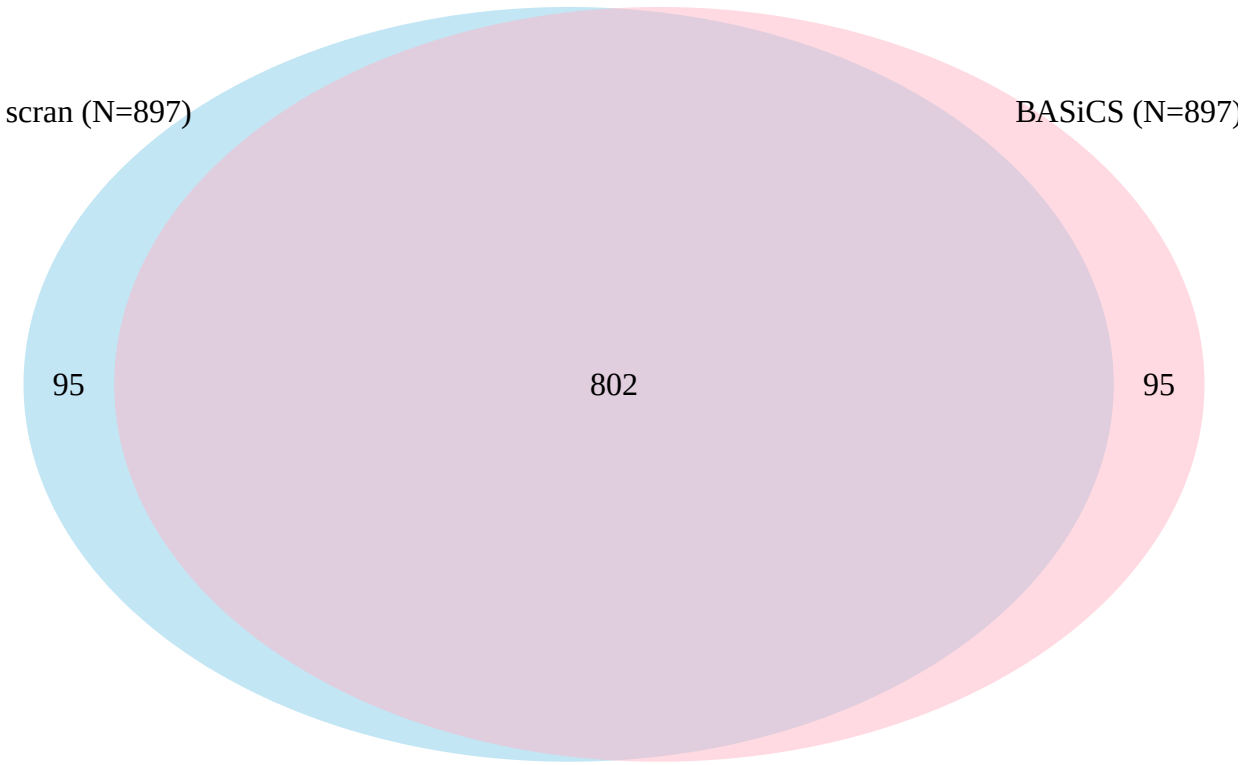



```
hvg.basics2<-find.high.var.dm(data=NormBasics2$counts,n.hvg =n.hvg,
                              plot = TRUE,
                              title = "BASiCS")
```




```
#define a list of highly variable genes
list.hvg2<-list(scran=hvg.scran2$genes.high.var,
                basics=hvg.basics2$genes.high.var)
```

```r
grid.newpage()
draw.pairwise.venn(area1 = length(list.hvg2$scran),
                   area2 = length(list.hvg2$basics),
                   cross.area = length(intersect(list.hvg2$scran,list.hvg2$basics)),
                   category = c(paste0("scran (N=",length(list.hvg2$scran), ")"),
                                paste0("BASiCS (N=",length(list.hvg2$basics), ")")),
                   lty = "blank",
                   fill = c("skyblue", "pink1"))
```

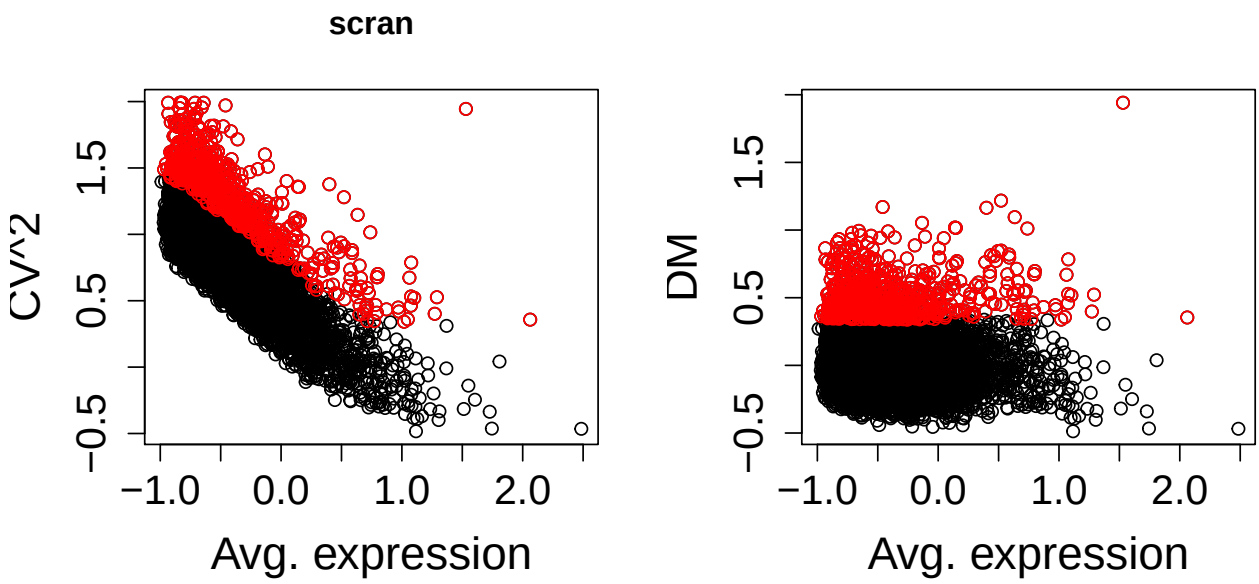



```
## (polygon[GRID.polygon.20], polygon[GRID.polygon.21], polygon[GRID.polygon.22], polygon[GRID.polygon.2
```

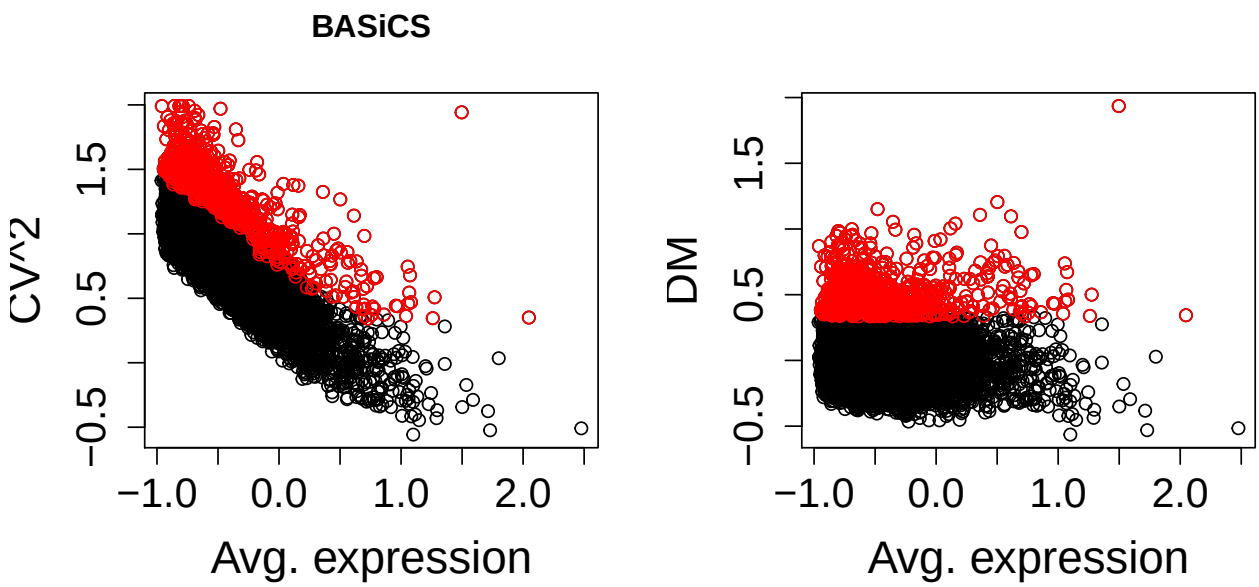
out of highly variable genes (89%) are shared across the two normalization methods.

Subpopulation 3: microglia

```
par(mfrow = c(1,2))
f<-0.1
n.hvg<-floor(f*length(Select3))
hvg.scran3<-find.high.var.dm(data=NormScran3$counts,n.hvg =n.hvg,
                             plot = TRUE,
                             title = "scran")
```

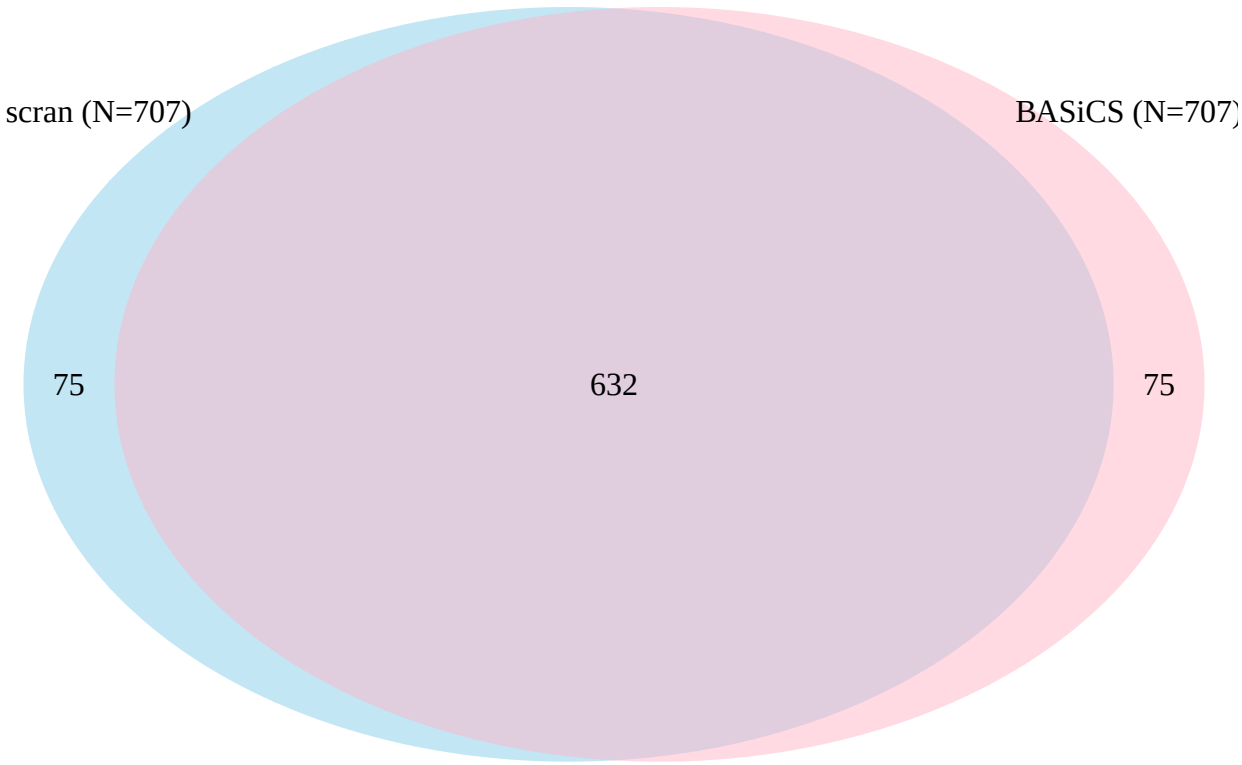



```
hvg.basics3<-find.high.var.dm(data=NormBasics3$counts,n.hvg =n.hvg,
                              plot = TRUE,
                              title = "BASiCS")
```




```
#define a list of highly variable genes
list.hvg3<-list(scran=hvg.scran3$genes.high.var,
                basics=hvg.basics3$genes.high.var)
```

```r
grid.newpage()
draw.pairwise.venn(area1 = length(list.hvg3$scran),
                   area2 = length(list.hvg3$basics),
                   cross.area = length(intersect(list.hvg3$scran,list.hvg3$basics)),
                   category = c(paste0("scran (N=",length(list.hvg3$scran), ")"),
                                paste0("BASiCS (N=",length(list.hvg3$basics), ")")),
                   lty = "blank",
                   fill = c("skyblue", "pink1"))
```




```
## (polygon[GRID.polygon.29], polygon[GRID.polygon.30], polygon[GRID.polygon.31], polygon[GRID.polygon.
```

632 out of 707 highly variable genes (89%) are shared across the two normalization methods.

References

Lun, Aaron TL, Karsten Bach, and John C Marioni. 2016. "Pooling Across Cells to Normalize Single-Cell RNA Sequencing Data with Many Zero Counts." *Genome Biology* 17 (1). BioMed Central: 1.

Vallejos, Catalina A, John C Marioni, and Sylvia Richardson. 2015. "BASiCS: Bayesian Analysis of Single-Cell Sequencing Data." *PLoS Computational Biology* 11 (6). Public Library of Science: e1004333.

Zeisel, Amit, Ana B Muñoz-Manchado, Simone Codeluppi, Peter Lönnerberg, Gioele La Manno, Anna Juréus, Sueli Marques, et al. 2015. "Cell Types in the Mouse Cortex and Hippocampus Revealed by Single-Cell RNA-Seq." *Science* 347 (6226). American Association for the Advancement of Science: 1138–42.