MULTI-seq_Barcodes

Jaime So

11/25/2019

deMULTIplex pre-processing

```
# cellranger data and raw fastq files for the miSeq eq run are in my dropbox
# use Seurat to read the cellranger output
cell.mat <- Read10X(data.dir="~/Dropbox (CahanLab)/Jamie.So/MULTI-seq/filtered_feature_bc_matrix/")
# the column names of the 10x cell ranger matrix are the sequences of the cell IDs
cell.IDs <- data.frame(cell = colnames(cell.mat))
# save the cell ID list
write.table(cell.IDs, "~/Desktop/Rotations/Rot1.Patrick.Cahan/Abby-MULTIseq/110616-BC_cellIDs.txt", sep = "\t", col.names=F, row.names=F, quote=F)
# make it into a character vector
cell.id.vec <- as.character(cell.IDs$cell)

# import the list of MULTI-seq barcodes which were provided in an excel spreadsheet
# I made them into a .txt file
bar.ref <- read.delim("multiBC.txt", sep="\n", header=F)
# make it into a character vector
bar.ref <- as.character(bar.ref$V1)

# run MULTI-seq preProcessing function
# specify UMI as c(17,26) for V2 chemistry, c(17,28) for V3
readTable <- MULTIseq.preProcess(R1="~/Dropbox (CahanLab)/Jamie.So/MULTI-seq/110616-BC_S2_L001_R1_001.fastq.gz", R2="~/Dropbox (CahanLab)/Jamie.So/MULTI-seq/110616-BC_S2_L001_R2_001.fastq.gz", cellIDs=cell.id.vec, cell=c(1,16), umi=c(17,26), tag=c(1,8))
```

```
## [1] "Reading in R1..."
## [1] "Reading in R2..."
## [1] "Assembling read table..."
```

```
# save output as a .csv
write.csv(readTable, file="110616-BC_readTable.csv", quote=F)

# perform MULTI-seq sample barcode alignment
# we only used the first 11 barcodes
bar.table <- MULTIseq.align(readTable, cell.id.vec, bar.ref[1:11])
```

```
## [1] "Bucketing cell IDs..."
## [1] "Bucketing read tables..."
## [1] "Aligning bucket #1..."
## [1] "2019-11-25 14:46:18 EST"
## [1] "Done aligning bucket #1..."
## Time difference of 1.059585 mins
```

```
# save result
write.csv(bar.table, file="110616-BC_barcode_counts.csv", quote=F)
```

Visually inspect barcode quality

```
# Visualize barcode space
bar.tsne <- barTSNE(bar.table[,1:11])
```

```
## Performing PCA
## Read the 6254 x 11 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.79 seconds (sparsity = 0.020540)!
## Learning embedding...
## Iteration 50: error is 91.141098 (50 iterations in 0.81 seconds)
## Iteration 100: error is 77.471414 (50 iterations in 0.84 seconds)
## Iteration 150: error is 75.272178 (50 iterations in 0.81 seconds)
## Iteration 200: error is 74.882142 (50 iterations in 0.84 seconds)
## Iteration 250: error is 74.744272 (50 iterations in 0.89 seconds)
## Iteration 300: error is 2.379554 (50 iterations in 0.78 seconds)
## Iteration 350: error is 1.954574 (50 iterations in 0.81 seconds)
## Iteration 400: error is 1.733490 (50 iterations in 0.87 seconds)
## Iteration 450: error is 1.594570 (50 iterations in 0.90 seconds)
## Iteration 500: error is 1.499474 (50 iterations in 0.87 seconds)
## Iteration 550: error is 1.431406 (50 iterations in 0.92 seconds)
## Iteration 600: error is 1.385731 (50 iterations in 0.92 seconds)
## Iteration 650: error is 1.355958 (50 iterations in 0.88 seconds)
## Iteration 700: error is 1.336083 (50 iterations in 0.93 seconds)
## Iteration 750: error is 1.320641 (50 iterations in 0.95 seconds)
## Iteration 800: error is 1.310680 (50 iterations in 0.91 seconds)
## Iteration 850: error is 1.301776 (50 iterations in 0.95 seconds)
## Iteration 900: error is 1.292991 (50 iterations in 0.96 seconds)
## Iteration 950: error is 1.284970 (50 iterations in 0.94 seconds)
## Iteration 1000: error is 1.277686 (50 iterations in 0.96 seconds)
## Iteration 1050: error is 1.271350 (50 iterations in 0.96 seconds)
## Iteration 1100: error is 1.264615 (50 iterations in 0.95 seconds)
## Iteration 1150: error is 1.258259 (50 iterations in 1.01 seconds)
## Iteration 1200: error is 1.252716 (50 iterations in 0.99 seconds)
## Iteration 1250: error is 1.247408 (50 iterations in 0.97 seconds)
## Iteration 1300: error is 1.243504 (50 iterations in 1.02 seconds)
## Iteration 1350: error is 1.240548 (50 iterations in 0.97 seconds)
## Iteration 1400: error is 1.237636 (50 iterations in 1.01 seconds)
## Iteration 1450: error is 1.234901 (50 iterations in 1.05 seconds)
## Iteration 1500: error is 1.231819 (50 iterations in 1.01 seconds)
## Iteration 1550: error is 1.228889 (50 iterations in 1.02 seconds)
## Iteration 1600: error is 1.226001 (50 iterations in 1.05 seconds)
## Iteration 1650: error is 1.223236 (50 iterations in 1.01 seconds)
## Iteration 1700: error is 1.220340 (50 iterations in 1.04 seconds)
## Iteration 1750: error is 1.217394 (50 iterations in 1.03 seconds)
## Iteration 1800: error is 1.215018 (50 iterations in 1.03 seconds)
## Iteration 1850: error is 1.212430 (50 iterations in 1.05 seconds)
## Iteration 1900: error is 1.210008 (50 iterations in 1.01 seconds)
## Iteration 1950: error is 1.207933 (50 iterations in 1.02 seconds)
## Iteration 2000: error is 1.206128 (50 iterations in 1.05 seconds)
## Iteration 2050: error is 1.203856 (50 iterations in 0.98 seconds)
## Iteration 2100: error is 1.202327 (50 iterations in 1.03 seconds)
## Iteration 2150: error is 1.201017 (50 iterations in 1.02 seconds)
## Iteration 2200: error is 1.199225 (50 iterations in 0.99 seconds)
## Iteration 2250: error is 1.197326 (50 iterations in 1.05 seconds)
```

```
## Iteration 2300: error is 1.195469 (50 iterations in 0.98 seconds)
## Iteration 2350: error is 1.193743 (50 iterations in 0.99 seconds)
## Iteration 2400: error is 1.192210 (50 iterations in 1.01 seconds)
## Iteration 2450: error is 1.190526 (50 iterations in 0.99 seconds)
## Iteration 2500: error is 1.189014 (50 iterations in 0.98 seconds)
## Fitting performed in 48.01 seconds.
```

```
pdf("bc.check.pdf")
for (i in 3:ncol(bar.tsne)){
  g <- ggplot(bar.tsne, aes(x = TSNE1, y = TSNE2, color = bar.tsne[,i])) + geom_point()
 + scale_color_gradient(low = "black", high = "red") + ggtitle(colnames(bar.tsne)[i]) +
 theme(legend.position = "none")
  print(g)
}

dev.off()
```

```
## quartz_off_screen
##                 2
```

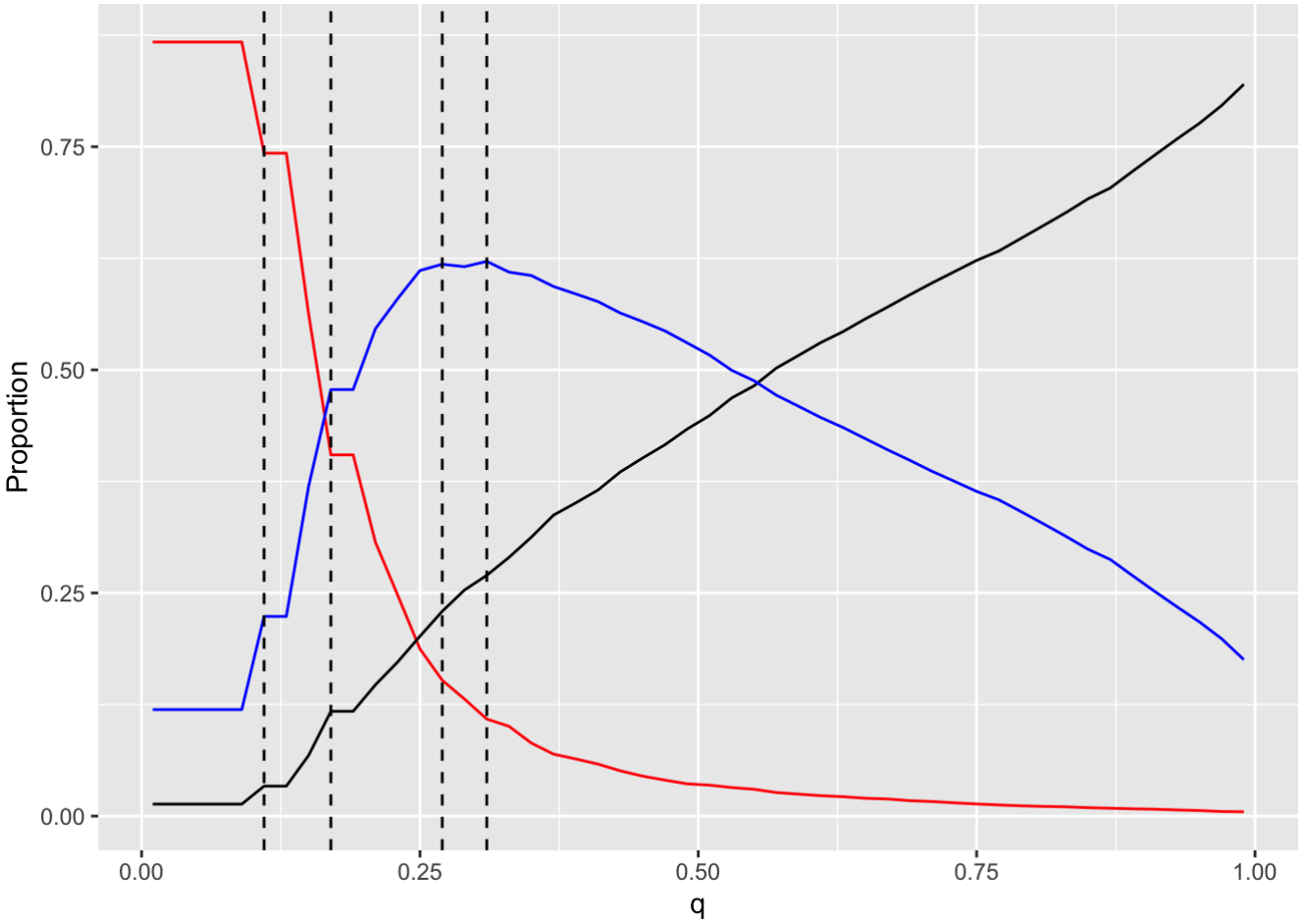
Sample Classification

```
# remove summary columns from bar.table
bar.table <- bar.table[, 1:11]

# perform quantile sweep
bar.table_sweep.list <- list()
n <- 0
for (q in seq(0.01, 0.99, by=0.02)) {
  print(q)
  n <- n + 1
  bar.table_sweep.list[[n]] <- classifyCells(bar.table, q=q)
  names(bar.table_sweep.list)[n] <- paste("q=",q,sep="")
}
```

```
## [1] 0.01
## [1] 0.03
## [1] 0.05
## [1] 0.07
## [1] 0.09
## [1] 0.11
## [1] 0.13
## [1] 0.15
## [1] 0.17
## [1] 0.19
## [1] 0.21
## [1] 0.23
## [1] 0.25
## [1] 0.27
## [1] 0.29
## [1] 0.31
## [1] 0.33
## [1] 0.35
## [1] 0.37
## [1] 0.39
## [1] 0.41
## [1] 0.43
## [1] 0.45
## [1] 0.47
## [1] 0.49
## [1] 0.51
## [1] 0.53
## [1] 0.55
## [1] 0.57
## [1] 0.59
## [1] 0.61
## [1] 0.63
## [1] 0.65
## [1] 0.67
## [1] 0.69
## [1] 0.71
## [1] 0.73
## [1] 0.75
## [1] 0.77
## [1] 0.79
## [1] 0.81
## [1] 0.83
## [1] 0.85
## [1] 0.87
## [1] 0.89
## [1] 0.91
## [1] 0.93
## [1] 0.95
## [1] 0.97
## [1] 0.99
```

```
# identify ideal inter-maxima quantile to set barcode-specific thresholds
threshold.results1 <- findThresh(call.list=bar.table_sweep.list)
ggplot(data=threshold.results1$res, aes(x=q, y=Proportion, color=Subset)) + geom_line()
 + theme(legend.position = "none") + geom_vline(xintercept=threshold.results1$extrema, l
ty=2) + scale_color_manual(values=c("red","black","blue"))
```



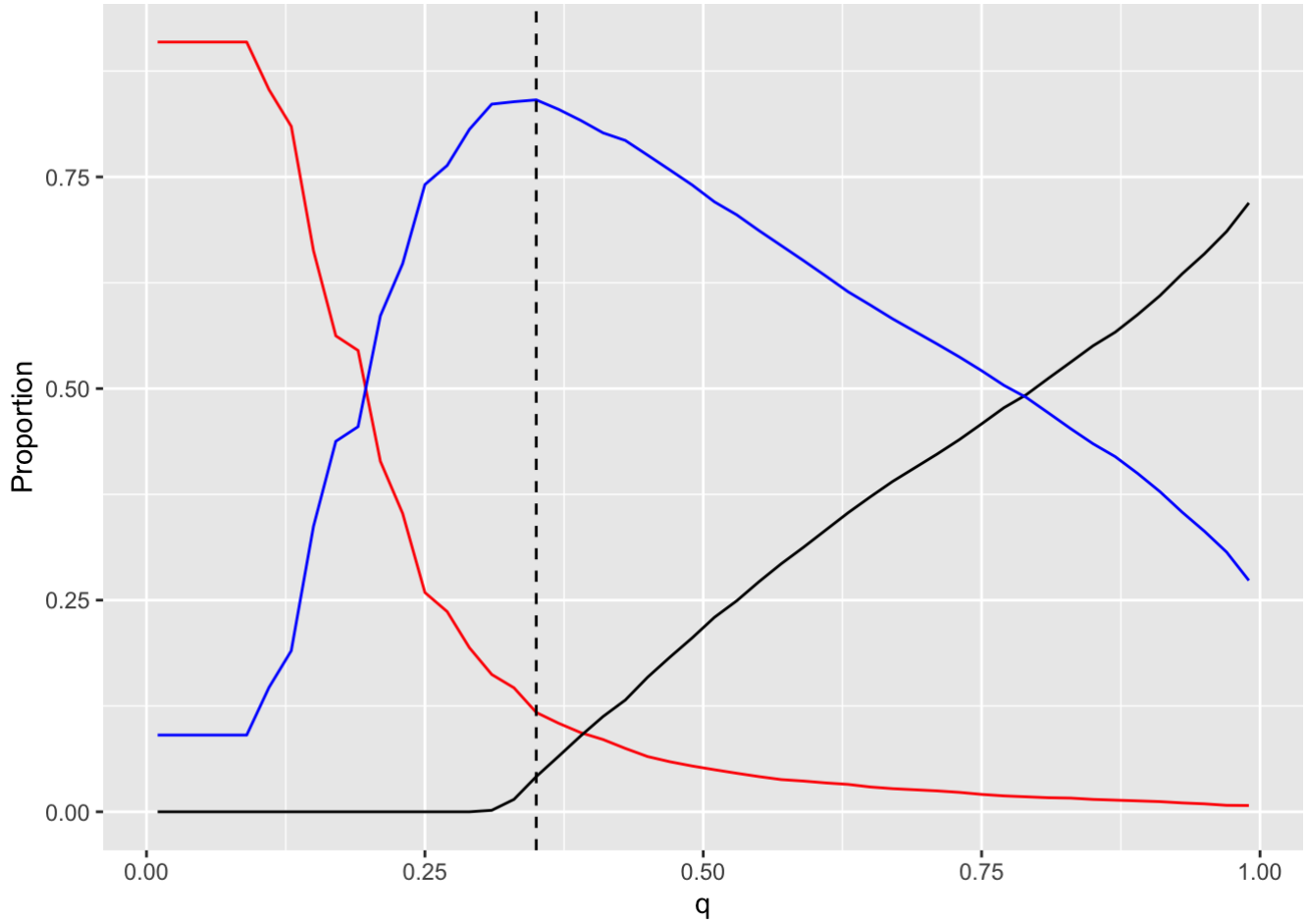

```
# finalize classifications, remove negative cells
round1.calls <- classifyCells(bar.table, q=findQ(threshold.results1$res, threshold.resul
ts1$extrema))
neg.cells <- names(round1.calls)[which(round1.calls=="Negative")]
bar.table1 <- bar.table[-which(rownames(bar.table) %in% neg.cells), ]

# do the sweep again: round 2
bar.table_sweep.list <- list()
n <- 0
for (q in seq(0.01, 0.99, by=0.02)) {
  print(q)
  n <- n + 1
  bar.table_sweep.list[[n]] <- classifyCells(bar.table1, q=q)
  names(bar.table_sweep.list)[n] <- paste("q=",q,sep="")
}
```

```
## [1] 0.01
## [1] 0.03
## [1] 0.05
## [1] 0.07
## [1] 0.09
## [1] 0.11
## [1] 0.13
## [1] 0.15
## [1] 0.17
## [1] 0.19
## [1] 0.21
## [1] 0.23
## [1] 0.25
## [1] 0.27
## [1] 0.29
## [1] 0.31
## [1] 0.33
## [1] 0.35
## [1] 0.37
## [1] 0.39
## [1] 0.41
## [1] 0.43
## [1] 0.45
## [1] 0.47
## [1] 0.49
## [1] 0.51
## [1] 0.53
## [1] 0.55
## [1] 0.57
## [1] 0.59
## [1] 0.61
## [1] 0.63
## [1] 0.65
## [1] 0.67
## [1] 0.69
## [1] 0.71
## [1] 0.73
## [1] 0.75
## [1] 0.77
## [1] 0.79
## [1] 0.81
## [1] 0.83
## [1] 0.85
## [1] 0.87
## [1] 0.89
## [1] 0.91
## [1] 0.93
## [1] 0.95
## [1] 0.97
## [1] 0.99
```

```
threshold.results2 <- findThresh(call.list=bar.table_sweep.list)
round2.calls <- classifyCells(bar.table1, q=findQ(threshold.results2$res, threshold.resu
lts2$extrema))
neg.cells <- c(neg.cells, names(round2.calls)[which(round2.calls == "Negative")])

ggplot(data=threshold.results2$res, aes(x=q, y=Proportion, color=Subset)) + geom_line()
 + theme(legend.position = "none") + geom_vline(xintercept=threshold.results2$extrema, l
ty=2) + scale_color_manual(values=c("red","black","blue"))
```




```
bar.table2 <- bar.table1[-which(rownames(bar.table1) %in% neg.cells), ]


# Repeat until all no negative cells remain (I think it looks good after 2)
final.calls <- c(round2.calls, rep("Negative",length(neg.cells)))
names(final.calls) <- c(names(round2.calls),neg.cells)
head(final.calls)
```

```
## AAACCCACACTGAATC AAACCCAGTTTACCAG AAACCCATCAATCCAG AAACCCATCCTAGCTC
##           "Bar9"        "Doublet"           "Bar9"           "Bar9"
## AAACCCATCGAAACAA AAACGAACACAAGTTC
##           "Bar4"        "Doublet"
```

```
str(final.calls)
```

```
##  Named chr [1:6444] "Bar9" "Doublet" "Bar9" "Bar9" "Bar4" "Doublet" "Bar10" ...
##  - attr(*, "names")= chr [1:6444] "AAACCCACACTGAATC" "AAACCCAGTTTACCAG" "AAACCCATCAAT
CCAG" "AAACCCATCCTAGCTC" ...
```

```
# try to plot classifications
bar.tsne$Classification <- "Singlet"
bar.tsne$Classification[which(final.calls[rownames(bar.tsne)]=="Doublet")] <- "Doublet"
bar.tsne$Classification[which(final.calls[rownames(bar.tsne)]=="Negative")] <- "Negativ
e"

ggplot(bar.tsne, aes(x=TSNE1, y=TSNE2)) + geom_point(size=0.25, aes(color=Classificatio
n)) + theme_classic() +  scale_color_manual(values = c("red", "grey", "black"))
```