Library fragment alignment

Tomas Bjorklund

Thu Oct 29 10:20:18 2020

This workflow identifies correct fragments from the Cre-recombined AAV plasmid library and aligns them to the CustumArray ordered nucleotide fragments using Blastn. Consistant mutations in each fragment/barcode combination are also registered as is the putity of each barcode.

```
suppressPackageStartupMessages(library(knitr))
```

Load the trimmed reads

```
load("data/LUTdna.rda")

fragments.file <- "data/fragments_AAVlibrary_complete.fastq.gz"
barcodes.file <- "data/barcodes_AAVlibrary_complete.fastq.gz"

reads.trim <- readFastq(fragments.file)
reads.BC <- readFastq(barcodes.file)
```

Make CustomArray reference index for Blast

```
LUT.fa <- tempfile(pattern = "LUT_", tmpdir = tempdir(), fileext = ".fa")
LUT.seq = ShortRead(DNAStringSet(LUT.dna$Sequence), BStringSet(1:length(LUT.dna$LUTnr)))
writeFasta(LUT.seq, LUT.fa)
```

Save unique fragments as fasta file

```
unique.reads <- unique(sread(reads.trim))
```

Select subset

```
# unique.reads <- unique.reads[sample(length(unique.reads), 50000)]

unique.reads <- ShortRead(DNAStringSet(unique.reads), BStringSet(1:length(unique.reads)))
fragments.unique.fa <- tempfile(pattern = "FragUnique_", tmpdir = tempdir(),
    fileext = ".fa")
writeFasta(unique.reads, fragments.unique.fa)
```

Align against the library using blast

```
blast.db <- tempfile(pattern = "blastDB_", tmpdir = tempdir(), fileext = ".db")
blast.out <- tempfile(pattern = "blastOut_", tmpdir = tempdir(), fileext = ".txt")
```

```r
sys.out <- system(paste("makeblastdb -in ", LUT.fa, " -out ", blast.db, " -dbtype nucl -title LUT -parse_se
    sep = ""), intern = TRUE, ignore.stdout = FALSE)

sys.out <- as.data.frame(sys.out)

colnames(sys.out) <- c("blastn database generation")
invisible(sys.out[" "] <- " ")
knitr::kable(sys.out[1:(nrow(sys.out)), ], format = "latex", booktabs = T) %>%
    kable_styling(latex_options = "striped")
```

blastn database generation
Building a new DB, current time: 10/29/2020 10:21:46
New DB name: /tmp/RtmplRTACf/blastDB_8244757c376.db
New DB title: LUT
Sequence type: Nucleotide
Keep MBits: T
Maximum file size: 1000000000B
Adding sequences from FASTA; added 92343 sequences in 4.87025 seconds.

```r
sys.out <- system(paste("export SHELL=/bin/sh; cat ", fragments.unique.fa, " | parallel --block ",
    floor(length(unique.reads)/detectCores()), " --recstart '>' --pipe 'blastn -max_target_seqs 25 -word_siz
    " -num_threads 1 -outfmt 10 -db ", blast.db, " -query - '> ", blast.out,
    " 2>&1", sep = ""), intern = TRUE, ignore.stdout = FALSE)  #

# table.blastn <- data.table(read.table(blast.out, header = FALSE, skip = 0,
# sep=';', stringsAsFactors = FALSE, fill=FALSE) , keep.rownames=FALSE,
# key='V1')

system(paste("gzip -c ", blast.out, " > ./data/blastOutput.csv.gz", sep = " "))

table.blastn <- data.table(scan(file = "./data/blastOutput.csv.gz", what = "character",
    sep = ";"), keep.rownames = FALSE, key = "V1")


if (length(grep("Warning", table.blastn$V1)) != 0) {
    warnings.out <- unique(table.blastn[grep("Warning", table.blastn$V1), ])
    table.blastn <- table.blastn[-grep("Warning", table.blastn$V1), ]
    setnames(warnings.out, "V1", c("blastn Warnings"))
    # knitr::kable(warnings.out[1:(nrow(warnings.out)),], format = 'markdown')
}

table.blastn[, `:=`(c("Reads", "Sequence", "identity", "alignmentLength", "mismatches",
    "gapOpens", "q_start", "q_end", "s_start", "s_end", "evalue", "bitScore"),
    tstrsplit(V1, ",", fixed = TRUE)), ]



table.blastn[, `:=`(Reads, as.character(sread(unique.reads[as.integer(Reads)])))]
table.blastn[, `:=`(Sequence, as.character(sread(LUT.seq[as.integer(Sequence)])))]
setkey(table.blastn, Sequence)
setkey(LUT.dna, Sequence)
table.blastn <- table.blastn[LUT.dna, nomatch = 0]
table.blastn[, `:=`(c("V1", "identity", "alignmentLength", "gapOpens", "q_start",
    "q_end", "s_start", "s_end", "evalue", "Sequence", "Names"), NULL)]
gc()  #garbage collection to reduce memory foot print. Can be removed for speed
```

```
            used   (Mb) gc trigger    (Mb)   max used    (Mb)
Ncells   9349520  499.4  216418051 11558.0  172911958  9234.5
Vcells 964075928 7355.4 4426097702 33768.5 5509384329 42033.3
```

```
table.blastn[, `:=`(bitScore, as.numeric(bitScore))]
table.blastn[, `:=`(mismatches, as.numeric(mismatches))]

setkeyv(table.blastn, c("Reads", "LUTnr"))
setorder(table.blastn, Reads, LUTnr, -bitScore)  #This makes sure that a fragment is only aligned once to th
table.blastn <- unique(table.blastn, by = c("Reads", "LUTnr"))

gc()  #garbage collection to reduce memory foot print. Can be removed for speed
```

```
            used   (Mb) gc trigger    (Mb)   max used    (Mb)
Ncells   9350423  499.4  173134440  9246.4  172911958  9234.5
Vcells 942290094 7189.2 3540878161 27014.8 5509384329 42033.3
```

```
table.blastn.topHit <- table.blastn[table.blastn[, .I[which.max(bitScore)],
    by = "Reads"]$V1]  # Select only rows with the highest bitScore

full.table <- data.table(Reads = as.character(sread(reads.trim)), BC = as.character(sread(reads.BC)),
    key = "Reads")
all.reads <- nrow(full.table)

full.table <- full.table[table.blastn.topHit, nomatch = 0]  # Merge reads with the top hit alignment

print(paste("Alignment percentage:", percent(nrow(full.table)/all.reads)))
```

```
[1] "Alignment percentage: 99.6%"
```

Starcode based barcode reduction

```
out.name.BC.star <- tempfile(pattern = "BCsc_", tmpdir = tempdir(), fileext = ".txt")

command.args <- paste("-c ", barcodes.file, " | starcode -t ", detectCores() -
    1, " --print-clusters -d", 1, " -r5 -q -o ", out.name.BC.star, sep = "")

system2("gunzip", args = command.args, stdout = TRUE, stderr = TRUE)
```

```
character(0)
```

```
table.BC.sc <- data.table(read.table(out.name.BC.star, header = FALSE, row.names = 1,
    skip = 0, sep = "\t", stringsAsFactors = FALSE, fill = FALSE), keep.rownames = TRUE,
    key = "rn")  #, nrows = 1000
table.BC.sc[, `:=`(V2, NULL)]

table.BC.sc <- table.BC.sc[, strsplit(as.character(V3), ",", fixed = TRUE),
    by = rn]

SC.droppedBC <- length(unique(sread(reads.BC))) - length(unique(table.BC.sc$V1) %in%
    unique(sread(reads.BC)))
print(paste("Dropped BCs in Starcode:", SC.droppedBC))
```

```
[1] "Dropped BCs in Starcode: 16603"
```

```r
# rm(reads.BC,reads.trim)

setnames(table.BC.sc, c("V1", "rn"), c("BC", "scBC"))

DNA_pscAAVlib <- table.BC.sc
save(DNA_pscAAVlib, file = "data/scBC_DNA_pscAAVlib.rda")
```

Replacing barcodes with Starcode reduced versions

```r
setkey(full.table, BC)
setkey(table.BC.sc, BC)
full.table <- full.table[table.BC.sc, nomatch = 0]
# full.table <- merge(full.table,table.BC.sc, by='BC', all = FALSE, all.x =
# FALSE) rm(table.BC.sc)

setnames(full.table, c("BC", "scBC"), c("oldBC", "BC"))

setkey(full.table, BC)

RetainedBC <- length(unique(full.table$oldBC))
scBC <- length(unique(full.table$BC))
print(paste("Original unique barcodes:", RetainedBC))
```

```
[1] "Original unique barcodes: 3579215"
```

```r
print(paste("SC reduced unique barcodes:", scBC))
```

```
[1] "SC reduced unique barcodes: 3188119"
```

```r
table.frag <- data.table(as.data.frame((rev(sort(table(full.table$oldBC))))[1:10],
    row.names = "Var1"), keep.rownames = TRUE)
# In R versions below 3.3 remove, row.names = 'Var1' to make this compatible
setnames(table.frag, colnames(table.frag), c("Original BC", "Count"))
knitr::kable(table.frag, format = "latex", booktabs = T) %>% kable_styling(latex_options = "striped")
```

Original BC	Count
GTGTCCATCTGGACGCGTAG	165
CATGCATGGATGACTTATGT	137
GATGGTATATATGCGGATGG	133
GTTTAAAGCAATATGCACAC	118
AATCGCTGGATGGTTTATGG	118
GCGCAATCGTGGGAACGCAC	108
GTACATTGCTGTGAGCCTGC	101
GTGCCTTTATTGGCGGCATT	100
GCGGGCGGGTGGAATGGCGT	99
GCGTATGTGCAGGCTCATTG	97

```r
table.frag <- data.table(as.data.frame((rev(sort(table(full.table$BC))))[1:10],
    row.names = "Var1"), keep.rownames = TRUE)
# In R versions below 3.3 remove, row.names = 'Var1' to make this compatible
setnames(table.frag, colnames(table.frag), c("SC reduced BC", "Count"))
knitr::kable(table.frag, format = "latex", booktabs = T) %>% kable_styling(latex_options = "striped")
```

```r
invisible(full.table[, `:=`(oldBC, NULL)])
```

SC reduced BC	Count
GTGTCCATCTGGACGCGTAG	173
GATGGTATATATGCGGATGG	151
CATGCATGGATGACTTATGT	151
AATCGCTGGATGGTTTATGG	125
GTTTAAAGCAATATGCACAC	122
GCGCAATCGTGGGAACGCAC	119
GCGGGCGGGTGGAATGGCGT	117
GTACATTGCTGTGAGCCTGC	110
GCGTATGTGCAGGCTCATTG	105
CTGCGCACGCTTGTACGCGG	103

Splitting reads into single-read and multi-read barcodes

```
full.table <- full.table[order(full.table$BC), ]
full.table[, `:=`(mismatches, as.numeric(mismatches))]

temp.table.single <- full.table[full.table[, .I[.N == 1], by = "BC"]$V1]
temp.table.multi <- full.table[full.table[, .I[.N > 1], by = "BC"]$V1]

temp.table.single[, `:=`(c("mCount", "tCount"), 1)]
temp.table.single$Mode <- "Amb"
setkeyv(temp.table.multi, c("BC", "LUTnr"))


temp.table.multi[, `:=`(c("bitScore", "mismatches", "tCount"), list(mean(bitScore),
    median(mismatches), .N)), by = key(temp.table.multi)]

temp.table.multi$Mode <- "Def"
temp.table.multi <- unique(temp.table.multi)

print("Utilized reads.......")
```

```
[1] "Utilized reads......."
```

```
print(nrow(full.table))
```

```
[1] 10642223
```

```
print("Whereof single reads.......")
```

```
[1] "Whereof single reads......."
```

```
print(nrow(temp.table.single))
```

```
[1] 1353656
```

Splitting multi-read barcodes into clean and chimeric

```
setkeyv(temp.table.multi, "BC")

temp.table.multi.clean <- temp.table.multi[temp.table.multi[, .I[.N == 1], by = "BC"]$V1]
temp.table.multi <- temp.table.multi[temp.table.multi[, .I[.N > 1], by = "BC"]$V1]
temp.table.multi.clean[, `:=`(mCount, tCount)]

print("Clean multi-read barcodes.......")
```

```
[1] "Clean multi-read barcodes......."
```

```
print(nrow(temp.table.multi.clean))
```

```
[1] 280333
```

```
print("Chimeric multi-read barcodes.......")
```

```
[1] "Chimeric multi-read barcodes......."
```

```
print(length(unique(temp.table.multi$BC)))
```

```
[1] 1554130
```

Calculate consensus alignment of chimeric barcodes

```
setkey(temp.table.multi, "BC")
temp.table.multi[, `:=`("mCount", tCount)]
temp.table.multi[, `:=`("tCount", sum(tCount)), by = "BC"]

setkey(temp.table.multi, "Reads")
temp.table.multi[, `:=`(c("LUTnr", "bitScore", "mismatches", "Structure"), NULL)]
setkey(table.blastn, "Reads")
temp.table.multi <- temp.table.multi[table.blastn, nomatch = 0, allow.cartesian = TRUE]

setkeyv(temp.table.multi, c("BC", "LUTnr"))
temp.table.multi[, `:=`(c("bitScore", "mismatches", "mCount"), list(max(bitScore),
    median(mismatches), sum(mCount))), by = key(temp.table.multi)]
gc()  #garbage collection to reduce memory foot print. Can be removed for speed
```

```
             used    (Mb) gc trigger    (Mb)   max used    (Mb)
Ncells   12952744   691.8   56732692  3029.9  173134440  9246.4
Vcells 2136119126 16297.3 3540878161 27014.8 5509384329 42033.3
```

```
temp.table.multi <- unique(temp.table.multi, by = c("BC", "LUTnr"))

setkeyv(temp.table.multi, "BC")
temp.table.multi <- temp.table.multi[temp.table.multi[, .I[mCount == max(mCount)],
    by = key(temp.table.multi)]$V1]
# Select only rows with the highest mCount

temp.table.multi <- temp.table.multi[temp.table.multi[, .I[which.max(bitScore)],
    by = key(temp.table.multi)]$V1]
# Select only rows with the highest bitScore

temp.table.multi[temp.table.multi$mCount == 1]$Mode <- "Amb"

print(paste("Number of barcodes with false mCount:", nrow(temp.table.multi[mCount >
    tCount])))
```

```
[1] "Number of barcodes with false mCount: 0"
```

```
temp.table.multi.consensus <- rbind(temp.table.multi, temp.table.multi.clean)

print(paste("Total number of definitive Barcodes:", length(grep("Def", temp.table.multi.consensus$Mode))))
```

```
[1] "Total number of definitive Barcodes: 1575111"
```

```r
print(paste("Total number of ambiguous Barcodes:", length(grep("Amb", temp.table.multi.consensus$Mode))))
```

```
[1] "Total number of ambiguous Barcodes: 259352"
```

```r
print(paste("Total number of single-read Barcodes:", nrow(temp.table.single)))
```

```
[1] "Total number of single-read Barcodes: 1353656"
```

```r
output.Table <- rbind(temp.table.multi.consensus, temp.table.single)
save(output.Table, file = "data/multipleContfragmentsComplete.rda")

print("Total analysis time:")
```

```
[1] "Total analysis time:"
```

```r
print(Sys.time() - strt1)
```

```
Time difference of 5.317156 hours
```

```r
devtools::session_info()
```

```
Session info -------------------------------------------------------------

 setting  value
 version  R version 3.4.2 (2017-09-28)
 system   x86_64, linux-gnu
 ui       X11
 language (EN)
 collate  en_US.UTF-8
 tz       UTC
 date     2020-10-29

Packages -----------------------------------------------------------------

 package                * version  date       source
 acepack                  1.4.1    2016-10-29 CRAN (R 3.4.2)
 AnnotationDbi          * 1.38.2   2017-11-29 Bioconductor
 AnnotationFilter         1.0.0    2017-11-29 Bioconductor
 AnnotationHub            2.8.3    2017-11-29 Bioconductor
 backports                1.1.1    2017-09-25 CRAN (R 3.4.2)
 base                   * 3.4.2    2017-10-06 local
 base64enc                0.1-3    2015-07-28 CRAN (R 3.4.2)
 Biobase                * 2.36.2   2017-11-29 Bioconductor
 BiocGenerics           * 0.22.1   2017-11-29 Bioconductor
 BiocInstaller            1.26.1   2017-10-10 Bioconductor
 BiocParallel           * 1.10.1   2017-11-29 Bioconductor
 biomaRt                  2.32.1   2017-11-29 Bioconductor
 Biostrings             * 2.44.2   2017-11-29 Bioconductor
 biovizBase             * 1.24.0   2017-11-29 Bioconductor
 bit                      1.1-12   2014-04-09 CRAN (R 3.4.2)
 bit64                    0.9-7    2017-05-08 CRAN (R 3.4.2)
 bitops                   1.0-6    2013-08-17 CRAN (R 3.4.2)
 blob                     1.1.0    2017-06-17 CRAN (R 3.4.2)
 BSgenome               * 1.44.2   2017-11-29 Bioconductor
 checkmate                1.8.4    2017-09-25 CRAN (R 3.4.2)
 cluster                  2.0.6    2017-03-16 CRAN (R 3.4.2)
 codetools                0.2-15   2016-10-05 CRAN (R 3.4.2)
 colorspace               1.3-2    2016-12-14 CRAN (R 3.4.2)
 compiler                 3.4.2    2017-10-06 local
 curl                     2.8.1    2017-07-21 CRAN (R 3.4.2)
 data.table             * 1.10.4-2 2017-10-12 url
```

```
datasets              * 3.4.2   2017-10-06 local
DBI                     0.7     2017-06-18 CRAN (R 3.4.2)
DelayedArray          * 0.2.7   2017-11-29 Bioconductor
devtools              * 1.13.3  2017-08-02 CRAN (R 3.4.2)
dichromat               2.0-0   2013-01-24 CRAN (R 3.4.2)
digest                  0.6.12  2017-01-27 CRAN (R 3.4.2)
doParallel            * 1.0.11  2017-09-28 CRAN (R 3.4.2)
ensembldb               2.0.4   2017-11-29 Bioconductor
evaluate                0.10.1  2017-06-24 CRAN (R 3.4.2)
foreach               * 1.4.3   2015-10-13 CRAN (R 3.4.2)
foreign                 0.8-69  2017-06-21 CRAN (R 3.4.2)
formatR                 1.5     2017-04-25 CRAN (R 3.4.2)
Formula               * 1.2-2   2017-07-10 CRAN (R 3.4.2)
GenomeInfoDb          * 1.12.3  2017-11-29 Bioconductor
GenomeInfoDbData        0.99.0  2017-11-29 Bioconductor
GenomicAlignments     * 1.12.2  2017-11-29 Bioconductor
GenomicFeatures       * 1.28.5  2017-11-29 Bioconductor
GenomicRanges         * 1.28.6  2017-11-29 Bioconductor
ggplot2               * 2.2.1   2016-12-30 CRAN (R 3.4.2)
graphics              * 3.4.2   2017-10-06 local
grDevices             * 3.4.2   2017-10-06 local
grid                  * 3.4.2   2017-10-06 local
gridExtra               2.3     2017-09-09 CRAN (R 3.4.2)
gtable                  0.2.0   2016-02-26 CRAN (R 3.4.2)
Gviz                  * 1.20.0  2017-11-29 Bioconductor
Hmisc                 * 4.0-3   2017-05-02 CRAN (R 3.4.2)
hms                     0.3     2016-11-22 CRAN (R 3.4.2)
htmlTable               1.9     2017-01-26 CRAN (R 3.4.2)
htmltools               0.3.6   2017-04-28 CRAN (R 3.4.2)
htmlwidgets             0.9     2017-07-10 CRAN (R 3.4.2)
httpuv                  1.3.5   2017-07-04 CRAN (R 3.4.2)
httr                    1.3.1   2017-08-20 CRAN (R 3.4.2)
hwriter                 1.3.2   2014-09-10 CRAN (R 3.4.2)
interactiveDisplayBase  1.14.0  2017-11-29 Bioconductor
IRanges               * 2.10.5  2017-11-29 Bioconductor
iterators             * 1.0.8   2015-10-13 CRAN (R 3.4.2)
kableExtra            * 0.5.2   2017-09-15 url
knitr                 * 1.17    2017-08-10 CRAN (R 3.4.2)
lattice               * 0.20-35 2017-03-25 CRAN (R 3.4.2)
latticeExtra            0.6-28  2016-02-09 CRAN (R 3.4.2)
lazyeval                0.2.0   2016-06-12 CRAN (R 3.4.2)
magrittr                1.5     2014-11-22 CRAN (R 3.4.2)
Matrix                  1.2-11  2017-08-21 url
matrixStats           * 0.52.2  2017-04-14 CRAN (R 3.4.2)
memoise                 1.1.0   2017-04-21 CRAN (R 3.4.2)
methods               * 3.4.2   2017-10-06 local
mime                    0.5     2016-07-07 CRAN (R 3.4.2)
munsell                 0.4.3   2016-02-13 CRAN (R 3.4.2)
nnet                    7.3-12  2016-02-02 CRAN (R 3.4.2)
parallel              * 3.4.2   2017-10-06 local
plyr                  * 1.8.4   2016-06-08 CRAN (R 3.4.2)
ProtGenerics            1.8.0   2017-11-29 Bioconductor
R6                      2.2.2   2017-06-17 CRAN (R 3.4.2)
RColorBrewer            1.1-2   2014-12-07 CRAN (R 3.4.2)
Rcpp                    0.12.13 2017-09-28 url
RCurl                   1.95-4.8 2016-03-01 CRAN (R 3.4.2)
readr                   1.1.1   2017-05-16 CRAN (R 3.4.2)
```

```
 rlang                    0.1.2    2017-08-09 CRAN (R 3.4.2)
 rmarkdown                1.6      2017-06-15 url
 rpart                    4.1-11   2017-04-21 CRAN (R 3.4.2)
 rprojroot                1.2      2017-01-16 CRAN (R 3.4.2)
 Rsamtools              * 1.28.0   2017-11-29 Bioconductor
 RSQLite                  2.0      2017-06-19 CRAN (R 3.4.2)
 rtracklayer            * 1.36.6   2017-11-29 Bioconductor
 rvest                    0.3.2    2016-06-17 CRAN (R 3.4.2)
 S4Vectors              * 0.14.7   2017-11-29 Bioconductor
 scales                 * 0.5.0    2017-08-24 CRAN (R 3.4.2)
 shiny                    1.0.5    2017-08-23 CRAN (R 3.4.2)
 ShortRead              * 1.34.2   2017-11-29 Bioconductor
 splines                  3.4.2    2017-10-06 local
 stats                  * 3.4.2    2017-10-06 local
 stats4                 * 3.4.2    2017-10-06 local
 stringdist             * 0.9.4.6  2017-07-31 CRAN (R 3.4.2)
 stringi                  1.1.5    2017-04-07 url
 stringr                  1.2.0    2017-02-18 CRAN (R 3.4.2)
 SummarizedExperiment   * 1.6.5    2017-11-29 Bioconductor
 survival               * 2.41-3   2017-04-04 CRAN (R 3.4.2)
 tibble                   1.3.4    2017-08-22 CRAN (R 3.4.2)
 tools                    3.4.2    2017-10-06 local
 utils                  * 3.4.2    2017-10-06 local
 VariantAnnotation        1.22.3   2017-11-29 Bioconductor
 withr                    2.0.0    2017-07-28 url
 XML                      3.98-1.9 2017-06-19 CRAN (R 3.4.2)
 xml2                     1.1.1    2017-01-24 CRAN (R 3.4.2)
 xtable                   1.8-2    2016-02-05 CRAN (R 3.4.2)
 XVector                * 0.16.0   2017-11-29 Bioconductor
 yaml                     2.1.14   2016-11-12 CRAN (R 3.4.2)
 zlibbioc                 1.22.0   2017-11-29 Bioconductor
```