CrispR analysis with Bioconductor

Jieun Jeong

2025-04-13

This vignette is based on a web page of Bioinformatics Resource Centre at Rockefeller University and it show how to use several Bioconductor packages in the analysis of data from a CrispR screen.

First steps

We start from setting the working directory and downloading the data if not downloaded already.

```
wd <- "/Users/jieun/Work/Crispr"
wdf <- function(...) file.path(wd,...)
Download <- function(url, filename) {
  file_path <- wdf(filename)
  if (file.exists(file_path))
    return()
  options(timeout = 1000)
  download.file(url, file_path)
  unzip(file_path,exdir = wd)
}
Download("https://github.com/LewisLabUCSD/PinAPL-Py/archive/master.zip",
         "pinaplData.zip")
Download("http://pinapl-py.ucsd.edu/example-data","TestData.zip")
dN <- wdf("New_Example_1580342908_4/Analysis")
Download("http://pinapl-py.ucsd.edu/run/download/example-run","Results.zip")
```

Quality control of fastq data

We may inspect several metrics of data quality. For larger data sets, we use random sampling that yields sufficiently accurate results, while time consuming steps may be avoided if data has clearly low quality.

```
require(ShortRead)
CRfile <- wdf("PinAPL-Py-master/Data/Tox-A_R01_98_S2_L008_R1_001_x.fastq.gz")
fqSample <- FastqSampler(CRfile,n=10^5)
fastq <- yield(fqSample)
fastq # check sample statistic
```

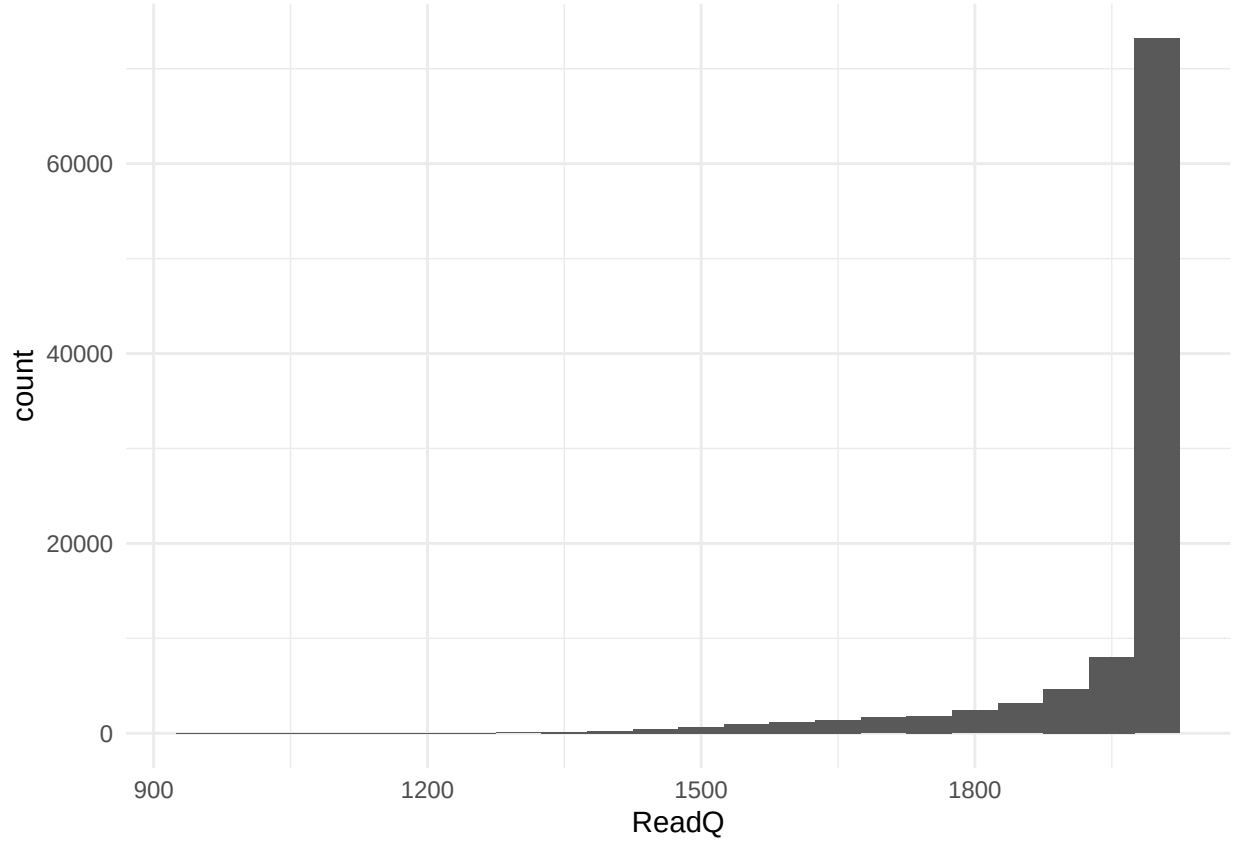
```
## class: ShortReadQ
## length: 100000 reads; width: 50 cycles
```

Now we use so-called accessor functions. The choice of bin width corresponds to the average bin quality rounded down to integer.

```
readQuality <- quality(fastq)
readQualities <- alphabetScore(readQuality)
readQualities[1:14]
```

```
##  [1] 1282 2020 1977 2020 2020 2016 2020 2012 1318 2003 1972 1900 2020 1716
```

```
require(ggplot2)
toPlot <- data.frame(ReadQ=readQualities)
ggplot(toPlot,aes(x=ReadQ))+geom_histogram(binwidth=50)+theme_minimal()
```



Q40 quality can be optimistic, so other measures can be more reliable, like how close the distribution of nucleotides on read position is close to what we can expect.

```
readSequences <- sread(fastq)
readSequences_AlpFreq <- alphabetFrequency(readSequences)
readSequences_AlpFreq[1:3,]
```

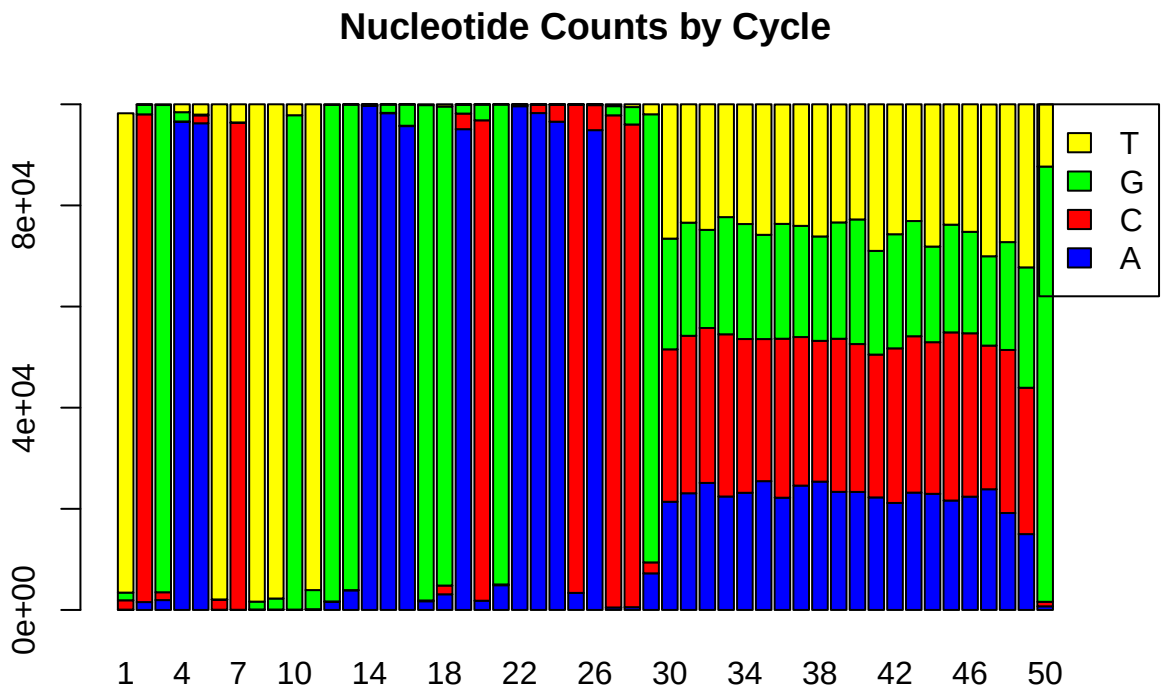
```
##       A  C  G  T M R W S Y K V H D B N - + .
## [1,] 16 10 12 12 0 0 0 0 0 0 0 0 0 0 0 0 0
## [2,] 13 13 10 14 0 0 0 0 0 0 0 0 0 0 0 0 0
## [3,] 17 13 14  6 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
readSequences_AlpbyCycle <- alphabetByCycle(readSequences)
readSequences_AlpbyCycle[1:4,1:10]
```

```
##          cycle
## alphabet  [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10]
##        A    72  1560  1952 96511 96246    51    37    36    33    56
##        C  1837 96450  1549    91  1568  1995 96286    63    52    22
##        G  1522  1889 96354  1836   148    35    91  1550  2179 97762
##        T 94791   101   145  1562  2038 97919  3586 98351 97736  2160
```

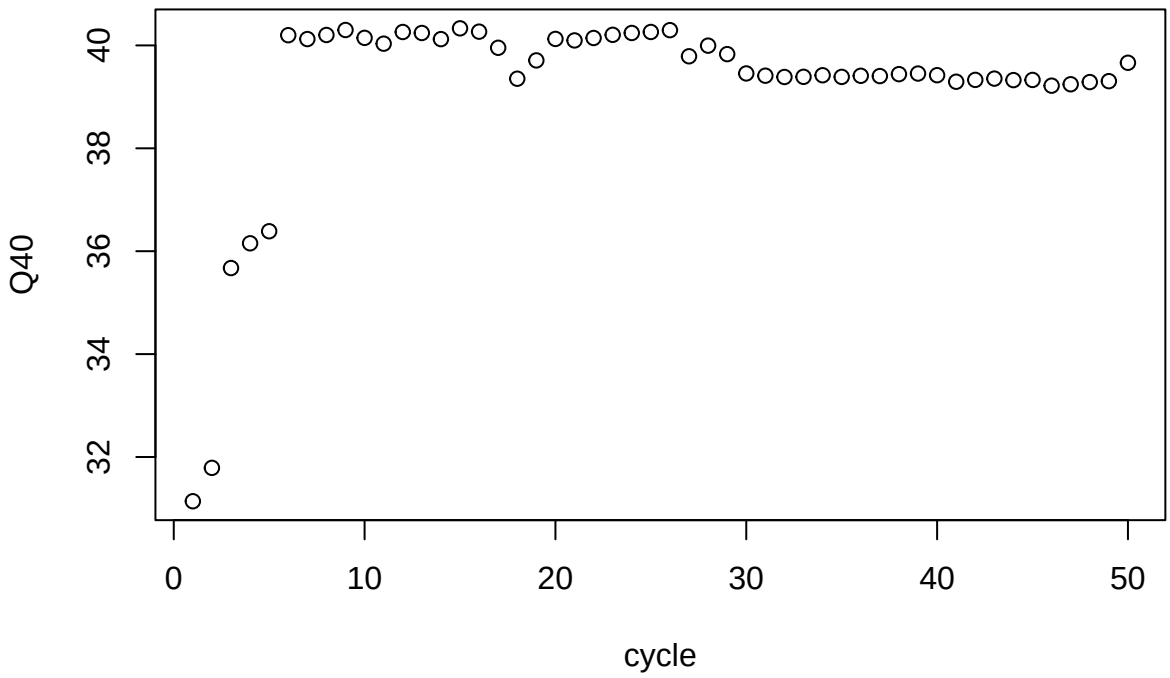
We can use barplot to visualize the distribution of the read frequencies. We see the reads have a constant prefix used to precipitate selectively DNA with guiding sequences, and a variable part, guiding sequences themselves, and effects of contamination as minorities in the prefix.

```
counts <- readSequences_AlpbyCycle[1:4,1:50]
barplot(counts, col = c("blue", "red", "green", "yellow"),
        legend.text = rownames(counts),
        args.legend = list( x = "topright", inset = c(-0.068, 0)),
        main = "Nucleotide Counts by Cycle",
        names.arg = 1:50)
```



The initial cycle may be affected by low quality rather than contamination, that can be checked as follows

```
qualAsMatrix <- as(readQuality,"matrix")
toPlot <- colMeans(qualAsMatrix)
plot(toPlot, xlab = "cycle", ylab = "Q40")
```



The conclusion here is that quality is fine, but in mapping tolerating one mismatch and trimming first two cycles can be considered.

Aligning single data file

The further analysis requires to align reads to the sgRNA require to quantify sgRNA abundance in our samples. We start by retrieving and indexing the reference.

```
GeCKO <- read.delim(wdf("PinAPL-py_demo_data/GeCKOv21_Human.tsv"))
GeCKO[1:2,]
```

```
##   gene_id          UID                  seq
## 1    A1BG HGLibA_00001 GTCGCTGAGCTCCGATTCGA
## 2    A1BG HGLibA_00002 ACCTGTAGTTGCCGGCGTGC
```

```
require(Biostrings)
sgRNAs <- DNAStringSet(GeCKO$seq)
names(sgRNAs) <- GeCKO$UID
writeXStringSet(sgRNAs, file=wdf("GeCKO.fa")) # sgRNA reference
Index <- wdf("GeCKO")
Index.fa <- paste0(Index,".fa")
require(Rsubread)
buildindex(Index,Index.fa, indexSplit=FALSE)
```

```
##
```

```
##          ==========     _____ _    _ ____  _____  ______          _____
##          =====         / ____| |  | |  _ \|  __ \|  ____|   /\   |  __ \
##            =====      | (___ | |  | | |_) | |__) | |__     /  \  | |  | |
##              ====      \___ \| |  | |  _ <|  _  /|  __|   / /\ \ | |  | |
##                ====    ____) | |__| | |_) | | \ \| |____ / ____ \| |__| |
##          ==========   |_____/ \____/|____/|_|  \_\______/_/    \_\_____/
##         Rsubread 2.20.0
##
## //================================ setting ==================================\\
## ||                                                                           ||
## ||                Index name : GeCKO                                         ||
## ||               Index space : base space                                    ||
## ||               Index split : no-split                                      ||
## ||          Repeat threshold : 100 repeats                                   ||
## ||              Gapped index : no                                            ||
## ||                                                                           ||
## ||       Free / total memory : 4.8GB / 16.0GB                                ||
## ||                                                                           ||
## ||               Input files : 1 file in total                               ||
## ||                             o GeCKO.fa                                    ||
## ||                                                                           ||
## \\===========================================================================//
##
## //================================ Running ==================================\\
## ||                                                                           ||
## || Check the integrity of provided reference sequences ...                   ||
## || No format issues were found                                               ||
## || Scan uninformative subreads in reference sequences ...                    ||
## || Estimate the index size...                                                ||
## || 3.0 GB of memory is needed for index building.                            ||
## || Build the index...                                                        ||
## || Save current index block...                                               ||
## ||  [ 0.0% finished ]                                                        ||
## ||  [ 10.0% finished ]                                                       ||
## ||  [ 20.0% finished ]                                                       ||
## ||  [ 30.0% finished ]                                                       ||
## ||  [ 40.0% finished ]                                                       ||
## ||  [ 50.0% finished ]                                                       ||
## ||  [ 60.0% finished ]                                                       ||
## ||  [ 70.0% finished ]                                                       ||
## ||  [ 80.0% finished ]                                                       ||
## ||  [ 90.0% finished ]                                                       ||
## ||  [ 100.0% finished ]                                                      ||
## ||                                                                           ||
## ||                      Total running time: 0.2 minutes.                     ||
## ||        Index /Users/jieun/Work/Crispr/GeCKO was successfully built.       ||
## ||                                                                           ||
## \\===========================================================================//
```

Now the test alignment

```
myFQs <- wdf("PinAPL-py_demo_data/Control_R1_S14_L008_R1_001_x.fastq.gz")
myMapped <- align(Index,myFQs,output_file=sub(".fastq.gz",".bam",myFQs),
                  nthreads=4,
```

```
                  unique=TRUE,
                  nBestLocations=1,
                  type = "DNA",
                  maxMismatches = 0,
                  TH1=1)
```

```
##
##          ==========     _____ _    _ ____  _____  ______          _____
##          =====         / ____| |  | |  _ \|  __ \|  ____|   /\   |  __ \
##            =====      | (___ | |  | | |_) | |__) | |__     /  \  | |  | |
##              ====      \___ \| |  | |  _ <|  _  /|  __|   / /\ \ | |  | |
##                ====    ____) | |__| | |_) | | \ \| |____ / ____ \| |__| |
##          ==========   |_____/ \____/|____/|_|  \_\______/_/    \_\_____/
##         Rsubread 2.20.0
##
## //================================ setting ==================================\\
## ||                                                                           ||
## || Function      : Read alignment (DNA-Seq)                                  ||
## || Input file    : Control_R1_S14_L008_R1_001_x.fastq.gz                     ||
## || Output file   : Control_R1_S14_L008_R1_001_x.bam (BAM)                    ||
## || Index name    : GeCKO                                                     ||
## ||                                                                           ||
## ||                    ------------------------------------                   ||
## ||                                                                           ||
## ||                              Threads : 4                                  ||
## ||                         Phred offset : 33                                 ||
## ||                            Min votes : 1 / 10                             ||
## ||                        Max mismatches : 0                                 ||
## ||                       Max indel length : 5                                ||
## ||             Report multi-mapping reads : no                               ||
## || Max alignments per multi-mapping read : 1                                 ||
## ||                                                                           ||
## \\===========================================================================//
##
## //================= Running (13-Apr-2025 18:36:12, pid=2935) =================\\
## ||                                                                           ||
## || Check the input reads.                                                    ||
## || The input file contains base space reads.                                 ||
## || Initialise the memory objects.                                            ||
## || Estimate the mean read length.                                            ||
## || The range of Phred scores observed in the data is [2,41]                  ||
## || Create the output BAM file.                                               ||
## || Check the index.                                                          ||
## || Init the voting space.                                                    ||
## || Global environment is initialised.                                        ||
## || Load the 1-th index block...                                              ||
## || The index block has been loaded.                                          ||
## || Start read mapping in chunk.                                              ||
## ||    5% completed, 2.1 mins elapsed, rate=110.5k reads per second           ||
## ||   11% completed, 2.1 mins elapsed, rate=183.3k reads per second           ||
## ||   18% completed, 2.1 mins elapsed, rate=246.7k reads per second           ||
## ||   24% completed, 2.1 mins elapsed, rate=303.1k reads per second           ||
## ||   31% completed, 2.1 mins elapsed, rate=350.0k reads per second           ||
```

```
## ||   38% completed, 2.1 mins elapsed, rate=393.6k reads per second           ||
## ||   44% completed, 2.1 mins elapsed, rate=428.2k reads per second           ||
## ||   51% completed, 2.1 mins elapsed, rate=459.8k reads per second           ||
## ||   57% completed, 2.1 mins elapsed, rate=481.7k reads per second           ||
## ||   64% completed, 2.1 mins elapsed, rate=504.4k reads per second           ||
## ||   70% completed, 2.1 mins elapsed, rate=2.8k reads per second             ||
## ||   73% completed, 2.1 mins elapsed, rate=2.9k reads per second             ||
## ||   76% completed, 2.1 mins elapsed, rate=3.0k reads per second             ||
## ||   79% completed, 2.1 mins elapsed, rate=3.2k reads per second             ||
## ||   83% completed, 2.1 mins elapsed, rate=3.3k reads per second             ||
## ||   86% completed, 2.1 mins elapsed, rate=3.4k reads per second             ||
## ||   89% completed, 2.1 mins elapsed, rate=3.6k reads per second             ||
## ||   93% completed, 2.1 mins elapsed, rate=3.7k reads per second             ||
## ||   96% completed, 2.1 mins elapsed, rate=3.8k reads per second             ||
## ||   99% completed, 2.1 mins elapsed, rate=3.9k reads per second             ||
## ||                                                                           ||
## ||                          Completed successfully.                          ||
## ||                                                                           ||
## \\==================================    ==================================//
##
## //================================   Summary ================================\\
## ||                                                                           ||
## ||                 Total reads : 500,000                                     ||
## ||                      Mapped : 413,480 (82.7%)                             ||
## ||             Uniquely mapped : 413,480                                     ||
## ||               Multi-mapping : 0                                           ||
## ||                                                                           ||
## ||                    Unmapped : 86,520                                      ||
## ||                                                                           ||
## ||                      Indels : 0                                           ||
## ||                                                                           ||
## ||                Running time : 2.1 minutes                                 ||
## ||                                                                           ||
## \\===========================================================================//
```

We got 82.7% of uniquely mapped reads, we can test what happens if we allowed number a mismatch by setting `maxMismatches = 1`. It increases the number of mapped reads by 8000, i.e. 1.6%. However, the reference has sequences of length 20, so the constant prefix is soft-clipped, and one mismatch among 20 or 19 may create data noise that could significantly affect some genes associated with the guiding sequences in the reference.

Quantifying alignment result for a collection of data files

To process bam files we will use packages that work with bam format without decompressing it into sam format. The scheme is that we create a vector of fasta-q files for samples in a CRISPR screen project, and then this vector together with the reference of gRNAs used in the project we create a data frame of counts. For reasons we will see later we do it for two examples.

```
# Files of PinAPL demo
d <- wdf("PinAPL-Py-master/Data/*fastq.gz")
Files <- Sys.glob(d)
```

Then we apply a "data frame maker" that can be used in other projects too.

```
require(GenomicAlignments)
Files_2_DF <- function(Files,indexFile) {
  f_no <- length(Files)
  for (i in seq(1,f_no)) {
    fi <- Files[i]
    # co <- CrisprCounts(fi,Index)
    output <- sub("\\.f[^.]*\\.gz$",".bam", Files[i])
    align(indexFile, Files[i], output_file=output, nthreads=4, unique=TRUE,
        nBestLocations=1, type = "DNA", maxMismatches = 0, TH1=1)
    # We decided to be most conservative and require that all 20 nucleotides of
    # gRNAs are in the alignment, but one can use different criteria too.
    temp <- readGAlignments(output)
    # temp <- temp[width(temp) == 20]
    co <- data.frame(table(seqnames(temp)),row.names = "Var1")
    if (i == 1)
      DF <- co
    else
      DF <- cbind(DF,co)
  }
  return(DF)
}
```

Now we use that code

```
Counts <- Files_2_DF(Files,Index)
```

```
head(Counts)
```

```
##              Freq Freq Freq Freq Freq Freq
## HGLibA_00001    7    0    3    2    2    8
## HGLibA_00002    0    1    0    0    0    0
## HGLibA_00003    0    1    0    0    0    0
## HGLibA_00004    0    9    0    0    0    1
## HGLibA_00005    8    1    1    1    2    4
## HGLibA_00006   26    1    0    0    0    0
```

Now we can apply DESeq to find differential guides and genes.

```
require(DESeq2)
useCol <- DataFrame(Group=factor(c("C","C","TA","TA","TB","TB"),
                    levels = c("C","TA","TB")),row.names = colnames(Counts))
dds <- DESeqDataSetFromMatrix(Counts,colData = useCol,design = ~Group)
dds <- DESeq(dds)
TAvsC <- results(dds,contrast=c("Group","TA","C"))
TAvsC <- TAvsC[order(TAvsC$pvalue),]
head(TAvsC)
```

```
## log2 fold change (MLE): Group TA vs C
## Wald test p-value: Group TA vs C
## DataFrame with 6 rows and 6 columns
##               baseMean log2FoldChange     lfcSE      stat      pvalue      padj
##              <numeric>      <numeric> <numeric> <numeric>   <numeric> <numeric>
```

```
## HGLibB_44908   584.528       10.06356   2.34056   4.29964 1.71079e-05  0.773168
## HGLibB_38324   414.723       11.15572   2.72463   4.09440 4.23260e-05  0.773168
## HGLibA_64400   597.672       11.21560   2.84370   3.94402 8.01272e-05  0.773168
## HGLibB_42762   192.966       11.20402   2.86267   3.91383 9.08433e-05  0.773168
## HGLibB_45454   115.173       10.31713   2.85665   3.61161 3.04296e-04  0.773168
## HGLibB_23505   159.617        9.73169   2.72577   3.57025 3.56642e-04  0.773168
```

Note that fold changes of top gRNA's are "decent" but adjusted p-values are not. Later we will see the reason. In general, DESeq is not recommended for CRISPR screens because interpretation of the read counts is different, we want to know essential genes in the screen context, not "differentially expressed". Thus we will try to apply recommended

```
require(CRISPRcleanR)
data(GeCKO_Library_v2) # GeCKO reference processed for use with CRISPRcleanR
```

With CRISPRcleanR (ccr from now) annotation for our CRISPR platform, we edit Counts to Counts.ccr, a format required by ccr. Some sgRNA rows will be removed, 1000 of them are for control sgRNAs, and a few for genes. The issue of annotating sgRNAs for those genes will be left for later and now we will remove them from consideration:

```
Counts.ccr <- Counts[rownames(GeCKO_Library_v2),]
Counts.ccr <- cbind(GeCKO_Library_v2[,c("CODE","GENES")],Counts.ccr)
#normANDfcs<-ccr.NormfoldChanges(Dframe=Counts.ccr, min_reads=10,
#                                 display=TRUE, saveToFig=FALSE,
#                                 libraryAnnotation = GeCKO_Library_v2,
#                                 ncontrols = 2)
```

It does not work. Probably the issue lies in very poor correlation of control samples, i.e. 0.13, compared to correlations within other pairs of replicates, 0.96 and 0.999.

```
cor(Counts.ccr[,3:8])
```

```
##              Freq     Freq.1     Freq.2     Freq.3     Freq.4     Freq.5
## Freq   1.00000000 0.14831035 0.07191373 0.08388689 0.08122845 0.08227878
## Freq.1 0.14831035 1.00000000 0.04210977 0.05026037 0.01562821 0.01546495
## Freq.2 0.07191373 0.04210977 1.00000000 0.95985160 0.02563935 0.02344570
## Freq.3 0.08388689 0.05026037 0.95985160 1.00000000 0.02457521 0.02274258
## Freq.4 0.08122845 0.01562821 0.02563935 0.02457521 1.00000000 0.99919456
## Freq.5 0.08227878 0.01546495 0.02344570 0.02274258 0.99919456 1.00000000
```

To study subsequent steps in the analysis, we consider another data set

```
data(KY_Library_v1.0) # sequence ids, annotations, sequence
d <- file.path(system.file("extdata",package = "CRISPRcleanR"),"*fq.gz")
# copied these files to my directory
d <- wdf("KY_extdata/*gz")
Files <- Sys.glob(d)
basename(Files)
```

```
## [1] "test_plasmid.fq.gz" "test_sample1.fq.gz" "test_sample2.fq.gz"
```

```r
d <- file.path(system.file("data",package = "CRISPRcleanR"))
data("KY_Library_v1.0.RData")
# create fasta of sequences in KY_Library_v1.0 and build the index
Index <- wdf("KY_Index")
Index.fa <- wdf("KY_Index.fa")
tempDF <- data.frame(ID = rownames(KY_Library_v1.0), seq = KY_Library_v1.0$seq)
faEntries <- paste0(">", tempDF$ID, "\n", tempDF$seq)
writeLines(faEntries,wdf("KY_Index.fa"))
buildindex(Index,Index.fa, indexSplit=FALSE)
```

```
##
##          ==========     _____ _    _ ____  _____  ______          _____
##          =====         / ____| |  | |  _ \|  __ \|  ____|   /\   |  __ \
##            =====      | (___ | |  | | |_) | |__) | |__     /  \  | |  | |
##              ====      \___ \| |  | |  _ <|  _  /|  __|   / /\ \ | |  | |
##                ====    ____) | |__| | |_) | | \ \| |____ / ____ \| |__| |
##          ==========   |_____/ \____/|____/|_|  \_\______/_/    \_\_____/
##         Rsubread 2.20.0
##
## //================================== setting ==================================\\
## ||                                                                             ||
## ||                Index name : KY_Index                                        ||
## ||               Index space : base space                                      ||
## ||               Index split : no-split                                        ||
## ||          Repeat threshold : 100 repeats                                     ||
## ||              Gapped index : no                                              ||
## ||                                                                             ||
## ||       Free / total memory : 5.6GB / 16.0GB                                  ||
## ||                                                                             ||
## ||               Input files : 1 file in total                                 ||
## ||                             o KY_Index.fa                                   ||
## ||                                                                             ||
## \\=============================================================================//
##
## //================================= Running ==================================\\
## ||                                                                             ||
## || Check the integrity of provided reference sequences ...                     ||
## || No format issues were found                                                 ||
## || Scan uninformative subreads in reference sequences ...                      ||
## || Estimate the index size...                                                  ||
## || 3.0 GB of memory is needed for index building.                              ||
## || Build the index...                                                          ||
## || Save current index block...                                                 ||
## ||   [ 0.0% finished ]                                                        ||
## ||   [ 10.0% finished ]                                                       ||
## ||   [ 20.0% finished ]                                                       ||
## ||   [ 30.0% finished ]                                                       ||
## ||   [ 40.0% finished ]                                                       ||
## ||   [ 50.0% finished ]                                                       ||
## ||   [ 60.0% finished ]                                                       ||
## ||   [ 70.0% finished ]                                                       ||
## ||   [ 80.0% finished ]                                                       ||
## ||   [ 90.0% finished ]                                                       ||
```

```
## ||  [ 100.0% finished ]                                                    ||
## ||                                                                         ||
## ||                       Total running time: 0.2 minutes.                  ||
## ||      Index /Users/jieun/Work/Crispr/KY_Index was successfully built.    ||
## ||                                                                         ||
## \\=========================================================================//
```

We repeat the construction of a data frame of read counts

```
Counts <- Files_2_DF(Files,Index)
colnames(Counts) <- c("Control","Sample1","Sample2")
Counts$gene <- KY_Library_v1.0$GENES
Counts <- Counts[,c(length(Files)+1,1:length(Files))]
# change of format required by CRISPRclearR
Counts <- tibble::rownames_to_column(Counts, var="sgRNA")
write.table(Counts,wdf("KY_Counts"), sep='\t', quote=FALSE, row.names=F)
```

Testing Files_2_DF and the second data set was very helpful, because the code was corrected: fastaq file have two possible suffices: .fq and .fastaq and converting the name to the name of .bam file had to be altered.

Finding essentiality (loss of fitness) and gain of fitness

The goal of CRISPR screen is to identify two types of interesting genes. The loss of fitness genes (in extreme case, essential) have smaller counts in knockout sample compared with control, and gain of fitness genes, have larger count. In tumors, the latter may be tumor supresing genes, typically, few but of special interest in designing therapies.

Without normalization, fold changes may have show artifacts caused by several mechanism, for that reason packages like CRISPRcleanR can be used to remove them, e.g. using the following function

```
ccr.AnalysisPipeline(
  file_counts = wdf("KY_Counts.tsv"),
  outdir= wdf("KY_pipeline/"),
  EXPname = "KY",
  library_builtin = "KY_Library_v1.0",
  run_mageck = FALSE,
  ncontrols = 1
)
```

```
## [1] "############################################"
## [1] "Input parameters"
## [1] "############################################"
## [1] "Parameter EXPname: KY (class: character-character)"
## [1] "Parameter outdir: /Users/jieun/Work/Crispr/KY_pipeline/ (class: character-character)"
## [1] "Parameter ncontrols: 1 (class: numeric-double)"
## [1] "Parameter min_reads: 30 (class: numeric-double)"
## [1] "Parameter method: ScalingByTotalReads (class: character-character)"
## [1] "Parameter FDRth: 0.05 (class: numeric-double)"
## [1] "Parameter library_builtin: KY_Library_v1.0 (class: character-character)"
## [1] "Parameter file_counts: /Users/jieun/Work/Crispr/KY_Counts.tsv (class: character-character)"
## [1] "Parameter aligner: Rsubreads (class: character-character)"
## [1] "Parameter maxMismatches: 0 (class: numeric-double)"
## [1] "Parameter nBestLocations: 2 (class: numeric-double)"
```

```
## [1] "Parameter nTrim5: 0 (class: character-character)"
## [1] "Parameter nTrim3: 0 (class: character-character)"
## [1] "Parameter strand: F (class: character-character)"
## [1] "Parameter duplicatedSeq: keep (class: character-character)"
## [1] "Parameter nthreads: 1 (class: numeric-double)"
## [1] "Parameter indexMemory: 2000 (class: numeric-double)"
## [1] "Parameter fastqc_plots: FALSE (class: logical-logical)"
## [1] "Parameter min.ngenes: 3 (class: numeric-double)"
## [1] "Parameter alpha: 0.01 (class: numeric-double)"
## [1] "Parameter nperm: 10000 (class: numeric-double)"
## [1] "Parameter p.method: hybrid (class: character-character)"
## [1] "Parameter min.width: 2 (class: numeric-double)"
## [1] "Parameter kmax: 25 (class: numeric-double)"
## [1] "Parameter nmin: 200 (class: numeric-double)"
## [1] "Parameter eta: 0.05 (class: numeric-double)"
## [1] "Parameter trim: 0.025 (class: numeric-double)"
## [1] "Parameter undo.splits: none (class: character-character)"
## [1] "Parameter undo.prune: 0.05 (class: numeric-double)"
## [1] "Parameter undo.SD: 3 (class: numeric-double)"
## [1] "Parameter run_mageck: FALSE (class: logical-logical)"
## [1] "Parameter path_to_mageck: mageck (class: character-character)"
## [1] "Parameter nseed: 679661 (class: numeric-double)"
```

```
## Error in value[[3L]](cond): ERROR: 3 | TYPE: CLIENT | MSG: Error in (function (counts, libraryAnnota
##  in step Check Library/Count data
```

This does not work either, this data set fails the requirement that 80% of the library sgRNA has at least 30 reads in control. We can check what threshold lower than 30 satisfies 80% criteria.

```
pr <- function(...) print(paste(...))
r <- dim(Counts)[1]*0.8
pr('required at least',r,'in every column, trying threshold 6')
```

```
## [1] "required at least 72568 in every column, trying threshold 6"
```

```
colSums(Counts[,3:5] >= 6)
```

```
## Control Sample1 Sample2
##   70261   65461   65209
```

```
pr('trying threshold 5')
```

```
## [1] "trying threshold 5"
```

```
colSums(Counts[,3:5] >= 5)
```

```
## Control Sample1 Sample2
##   75864   70825   70746
```

```
pr('threshold 5 almost satisfies 80%, but is 6 times to small')
```

```
## [1] "threshold 5 almost satisfies 80%, but is 6 times to small"
```

```
pr('sums of readcounts 6 times too small, and they are')
```

```
## [1] "sums of readcounts 6 times too small, and they are"
```

```
colSums(Counts[3:5])
```

```
## Control Sample1 Sample2
##  863214  864168  862881
```

Actually, the fastaq files had 1 million fragments each, not all of them were mapped, so we need 6 million fragments or more in fastaq files. Let's test an alternative example mapped to the same set of sgRNAs

```
fn <- file.path( system.file("extdata", package = "CRISPRcleanR"),
                 "HT-29_counts.tsv")
Counts <- read.delim(fn)
dim(Counts)
```

```
## [1] 90709     6
```

```
colSums(Counts[3:6])/1000/1000
```

```
## ERS717283.plasmid       HT29_c904R1       HT29_c904R2       HT29_c904R3
##          38.44579          44.00473          44.23206          50.09768
```

We see that sums of read counts are ca. 50 times larger than the minimum we estimated (we showed sums in millions). We want at most 18,000 entries below 30 in each column

```
colSums(Counts[,3:6] < 30)
```

```
## ERS717283.plasmid       HT29_c904R1       HT29_c904R2       HT29_c904R3
##              3828              4848              4895              4618
```

As expected, the critical statistics are small enough. In the second workout we will see how to use pipeline results.