DRUG-seq analysis workflow

For compound profiling in 384 well plates, usually 6 libraries are run on the same Hiseq 4000 flow cell with each library indexed 4 indices to provide barcode diversity.

1. Convert bcl files into fastq and concatenate fastq files from the same library into a single fastq
    - This shell script assumes 20 cycle read1, 8 cycle index, and 52 cycle read2. Adjust these parameters accordingly.
    - Sample sheet file: drugseq_Hiseq.csv
    - Sample sheet used in the bcl2fastq conversion is standard if all 6 libraries were made using indices A-F and 1-4 barcodes, and all libraries were run for every lane on Hiseq. Adjust accordingly if only selected indexing primers are used.
    - By the end of this process, paired end fastq R1 and R2 files are generated for library A, B, C, D, E, F, as well as an input.txt file for the next step. An input.txt file would look like this: input.txt

bcl2fastq processing

```
#!/bin/sh
# Usage: drugseq.sh $bcl_folder

hiseq_folder=$(echo $1 | awk 'BEGIN {FS="/"}{print $(NF)}')
fastq_folder="/da/NS/tnt/data/by-assay/drugseq/U2OS/cmp_500/Hise
q_production/""$hiseq_folder"
mkdir $fastq_folder
fastq_input="$fastq_folder""/input"
mkdir $fastq_input
fastq_output="$fastq_folder""/output"

/usr/prog/bcl2fastq2/2.17.1.14/bin/bcl2fastq --runfolder-dir $1
--output-dir $fastq_input --mask-short-adapter-reads 0
--minimum-trimmed-read-length 0 --sample-sheet
/home/yech3/repos/drugseq/pre_process/drugseq_Hiseq.csv
--use-bases-mask y20,i8,y45

# merge 1-4 into a single fastq

for r in R1 R2
do
 for l in A B C D E F
 do
  to_cat=$(echo "$fastq_input""/library_"$l*$r"_001.fastq.gz")
  echo $to_cat
  nohup cat $to_cat | gunzip >
"$fastq_input""/library_"$l"_"$r".fastq" &
 done
done

write to input.txt
cat > "$fastq_input""/input.txt" << EOF
$fastq_input,library_A_R2.fastq,library_A_R1.fastq,/da/NS/tnt/pr
og/countumi/star_ref_env.sh
$fastq_input,library_B_R2.fastq,library_B_R1.fastq,/da/NS/tnt/pr
og/countumi/star_ref_env.sh
$fastq_input,library_C_R2.fastq,library_C_R1.fastq,/da/NS/tnt/pr
og/countumi/star_ref_env.sh
$fastq_input,library_D_R2.fastq,library_D_R1.fastq,/da/NS/tnt/pr
og/countumi/star_ref_env.sh
$fastq_input,library_E_R2.fastq,library_E_R1.fastq,/da/NS/tnt/pr
og/countumi/star_ref_env.sh
$fastq_input,library_F_R2.fastq,library_F_R1.fastq,/da/NS/tnt/pr
og/countumi/star_ref_env.sh
EOF
```

2. Run mapping and demultiplexing algorithm
    - Check Hiseq run parameters to determine trimming parameters -f, -t
    - Check experiment setup to determine whether to use single human genome run environment (/da/NS/tnt/prog/countumi/star_ref_env.sh) or combined genome environment (/da/NS/tnt/prog/countumi_KCl/star_ref_env.sh)

for co-culture experiment.

- Check Tripti's site for more details: Drug-seq UMI count Workflow on HP
- Parameters used on /usr/prog/scicomp/drug-seq/run_umi_workflow.sh are different from /da/NS/tnt/prog/countumi/run_umi_workflow.sh, but the basic input requirements are the same. Note in /usr/prog/scicomp/drug-seq/run_umi_workflow.sh, the input.txt file is called SampleSheet.csv, but this SampleSheet.csv should not be confused with Sample sheet file used for demultiplexing above.
- Lots of emails will be sent once run is initiated. Make sure to clean up inbox.
- Check Tripti's instruction for more details: Drug-seq UMI count Workflow on HP

/da/NS/tnt/prog/countumi/run_umi_workflow.sh -i /da/NS/tnt/data/by-assay/drugseq/U2OS/cmp_500/Hiseq_production/170602_K00381_0047_AHHGKTBBXX/input/input.txt -e chaoyang.ye@novartis.com -o /da/NS/tnt/data/by-assay/drugseq/U2OS/cmp_500/Hiseq_production/170602_K00381_0047_AHHGKTBBXX/output -s /da/NS/tnt/prog/countumi -f 5 -t 0

execute run

```
/da/NS/tnt/prog/countumi/run_umi_workflow.sh -i
/da/NS/tnt/data/by-assay/drugseq/U2OS/cmp_500/Hiseq_production/1
70602_K00381_0047_AHHGKTBBXX/input/input.txt -e
chaoyang.ye@novartis.com -o
/da/NS/tnt/data/by-assay/drugseq/U2OS/cmp_500/Hiseq_production/1
70602_K00381_0047_AHHGKTBBXX/output -s /da/NS/tnt/prog/countumi
-f 5 -t 0
```

3. R analysis for differential gene expression
   - A test data set for 3 plates used in the following code is here: /Volumes/ldrive$/NS/USCA-EXPERIMENTS-COMPILATION-I10269/CY/DRUG-seq compound test data
   - 384 barcode table is here: 384_barcode
   - The compound treatment metadata is here: DataSheet_AEOS3333479724.xlsx

```
library(DESeq2)
library(reshape2)
library(xlsx)
library(parallel)

setwd("/Users/yech3/Documents/Scripts/drugseq/by quatrant")
barcode=read.table("384_barcode") # barcode arranged by columns
in 384 well format
barcode_UMI=paste("UMI_", barcode[, 1], sep="")
barcode_reads=paste("Reads_", barcode[, 1], sep="")

setwd("/Volumes/ldrive$/NS/USCA-EXPERIMENTS-COMPILATION-I10269/C
Y/DRUG-seq compound test data") # define the test data directory
plate_A=read.table("A_compiled_results.dat", head=T)
plate_B=read.table("B_compiled_results.dat", head=T)
plate_C=read.table("C_compiled_results.dat", head=T)

# eliminate empty wells
plate_A_reads=plate_A[, c("Gene", barcode_reads)]
plate_A_UMI=plate_A[, c("Gene", barcode_UMI)]
plate_B_reads=plate_B[, c("Gene", barcode_reads)]
plate_B_UMI=plate_B[, c("Gene", barcode_UMI)]
plate_C_reads=plate_C[, c("Gene", barcode_reads)]
plate_C_UMI=plate_C[, c("Gene", barcode_UMI)]

# separate out ERCC counts
```

```
plate_A_UMI_noERCC=plate_A_UMI[grep(",ERCC", plate_A_UMI[,1],
invert=T),]
plate_A_UMI_ERCC=plate_A_UMI[grep(",ERCC", plate_A_UMI[,1]),]
plate_B_UMI_noERCC=plate_B_UMI[grep(",ERCC", plate_B_UMI[,1],
invert=T),]
plate_B_UMI_ERCC=plate_B_UMI[grep(",ERCC", plate_B_UMI[,1]),]
plate_C_UMI_noERCC=plate_C_UMI[grep(",ERCC", plate_C_UMI[,1],
invert=T),]
plate_C_UMI_ERCC=plate_C_UMI[grep(",ERCC", plate_C_UMI[,1]),]

# highest gene expression
hist(apply(plate_A_UMI_noERCC[, 2:385], 2, max))
hist(apply(plate_A_UMI_ERCC[, 2:385], 2, max))
hist(apply(plate_B_UMI_noERCC[, 2:385], 2, max))
hist(apply(plate_B_UMI_ERCC[, 2:385], 2, max))
hist(apply(plate_C_UMI_noERCC[, 2:385], 2, max))
hist(apply(plate_C_UMI_ERCC[, 2:385], 2, max))

# number of genes detected
hist(apply(plate_A_UMI_noERCC[, 2:385], 2, function(x)sum(x>0)))
hist(apply(plate_A_UMI_ERCC[, 2:385], 2, function(x)sum(x>0)))
hist(apply(plate_B_UMI_noERCC[, 2:385], 2, function(x)sum(x>0)))
hist(apply(plate_B_UMI_ERCC[, 2:385], 2, function(x)sum(x>0)))
hist(apply(plate_C_UMI_noERCC[, 2:385], 2, function(x)sum(x>0)))
hist(apply(plate_C_UMI_ERCC[, 2:385], 2, function(x)sum(x>0)))

# merge first 3 plates to produce replicates
set1_UMI=join_all(list(plate_A_UMI_noERCC, plate_B_UMI_noERCC,
plate_C_UMI_noERCC), type="inner", by="Gene")
rownames(set1_UMI)=set1_UMI[,1]
set1_UMI=set1_UMI[,2:ncol(set1_UMI)]

# normalization
normalization<-
  function(x){
    sf_data <- estimateSizeFactorsForMatrix(x)
    nCounts_data <- t( t(x) / sf_data )
    nCounts_data=as.data.frame(cbind(row.names(nCounts_data),
nCounts_data)) # prepare table for output
    colnames(nCounts_data)[1]="Gene"
    write.table(nCounts_data, file="normalized_count.txt",
quote=F, row.names=F, col.names=T, sep="\t") # write table as
starting data set
    nCounts_data=read.table(file="normalized_count.txt", head=T,
row.names=1)
  }

set1_UMI_norm=normalization(set1_UMI)

# retrieve experimental metadata info
cmp_ID=read.xlsx("/Users/yech3/Documents/NGS_analysis/Drug-seq/U
2OS/plate6_7/plate6/DataSheet_AEOS3333479724.xlsx", sheetIndex=1,
header=F)
```

```
cmp_ID=as.vector(as.matrix(cmp_ID)) # arrange by columns
cmp_ID=cmp_ID[!is.na(cmp_ID)] # remove NA
cmp_ID=rep(cmp_ID, 3)

dosage=read.xlsx("/Users/yech3/Documents/NGS_analysis/Drug-seq/U
2OS/plate6_7/plate6/DataSheet_AEOS3333479724.xlsx", sheetIndex=2,
header=F)
dosage=as.vector(as.matrix(dosage)) # arrange by columns
dosage=dosage[!is.na(dosage)]
dosage=rep(dosage, 3)

# differential gene expression analysis
control="DMSO"
cmp_ID_unique=unique(cmp_ID)
cmp_ID_unique=cmp_ID_unique[! cmp_ID_unique %in% control]
dose_unique=unique(dosage)
dose_unique=dose_unique[dose_unique !=0]

for (cmp in cmp_ID_unique) {
  for (dose in dose_unique) {
      untreated = set1_UMI[,cmp_ID==control & dosage==0]
      treatment = set1_UMI[,cmp_ID==cmp & dosage==dose]
      sample_names = c(colnames(untreated), colnames(treatment))
      condition = c(rep(control, length(cmp_ID[cmp_ID ==
control])), rep(paste(cmp,"_", as.character(dose), sep=""), 3))
      colData=data.frame(condition, row.names=sample_names)
      dds <- DESeqDataSetFromMatrix(countData =
data.frame(untreated, treatment), colData = colData, design = ~
condition)
      dds$condition=relevel(dds$condition, ref=control)
      dds <- DESeq(dds, fitType='local')
      res <- results(dds)
      resOrdered <- res[order(res$padj),]
      sig_gene <- rownames(resOrdered[resOrdered$padj<0.05 &
!is.na(resOrdered$padj),])
      if (length(sig_gene) == 0){
        to_write <- c(cmp, dose, length(sig_gene))
      } else {
        to_write <- c(cmp, dose, length(sig_gene),
paste(unlist(strsplit(sig_gene, split=","))[seq(1,
2*length(sig_gene), 2)], collapse = " "))
      }
      write.table(as.list(to_write), quote=F, col.names=F,
row.names=x, sep=",", file=paste(as.character(x), "_gene.csv",
sep=""))
      write.table(resOrdered, quote=F, row.names=T, sep="\t",
file=paste(x, cmp_ID[x], dosage[x], "diff", sep="_"))
  }
}



# in terminal
```

```
# cat *_gene.csv > diff_gene_list1.csv
```

```
# sort by first column in excel to rearrange
# for f in *_padj.csv; do cat diff-padj.csv | paste - $f >temp;
cp temp diff-padj.csv; done; rm temp
```

For CRISPR KO profiling in 384 well plates, usually only 1 plate/library is run on a single Nextseq flow cell with no need to sequence library index.

1. Convert bcl files into fastq
   - In this example, paired end run on Nextseq with 20 and 62 cycles

```
/usr/prog/bcl2fastq2/2.17.1.14/bin/bcl2fastq --runfolder-dir
/labdata/ldrive/NS/USCA-NEXSEQ-I10507/07092017_CRISPRSeq37grnas_
ngn2/170709_NB501111_0029_AHYHVHBGX2 --output-dir
/da/NS/tnt/data/by-assay/drugseq/H1/CRISPRseq/37gRNAs/trial2_0717
--mask-short-adapter-reads 0 --minimum-trimmed-read-length 0
--sample-sheet
/da/NS/tnt/data/by-assay/drugseq/H1/CRISPRseq/37gRNAs/trial2_071
7/SampleSheet.csv --use-bases-mask y20,y62
```

2. Run mapping and demultiplexing algorithm
   - This is the input.txt file to run UMI counts

```
/da/NS/tnt/data/by-assay/drugseq/H1/CRISPRseq/37gRNAs/trial2_071
7,CRISPRSeq_37gRNAs_R2.fastq,CRISPRSeq_37gRNAs_R1.fastq,/da/NS/t
nt/prog/countumi_KCl/star_ref_env.sh
```

   - Here because 5' end has high error rate and 3' end has lower Q30 scores, I decided to trim on both ends of the reads

```
/usr/prog/scicomp/drug-seq/run_umi_workflow.sh --samplesheet
/da/NS/tnt/data/by-assay/drugseq/H1/CRISPRseq/37gRNAs/trial2_071
7/input.txt --email chaoyang.ye@novartis.com --outdir
/da/NS/tnt/data/by-assay/drugseq/H1/CRISPRseq/37gRNAs/trial2_071
7/output --scriptdir /usr/prog/scicomp/drug-seq --clip5prime 1
--clip3prime 2 --umiruntime 48 --umimemory 15 --runmode run
```

3. R analysis for differential gene expression
   - A test data set can be found here: /Volumes/ldrive$/NS/USCA-EXPERIMENTS-COMPILATION-I10269/CY/DRUG-seq CRISPR test data/compiled_results.dat
   - CRISPR treatment meta data is here: Plate map.xlsx
   - Because there are two cell lines studies together in the same plate, differential gene expression was carried out separately
   -

```
library(DESeq2)
library(reshape2)
library(xlsx)
library(plyr)
setwd("/Users/yech3/Documents/Scripts/drugseq/by quatrant")
barcode=read.table("384_barcode") # barcode arranged by columns
in 384 well format
```

```
barcode_UMI=paste("UMI_", barcode[, 1], sep="")
barcode_reads=paste("Reads_", barcode[, 1], sep="")

setwd("/Volumes/ldrive$/NS/USCA-EXPERIMENTS-COMPILATION-I10269/C
Y/DRUG-seq CRISPR test data") # define test data directory
CRISPR_seq=read.table("compiled_results.dat", head=T, row.names =
1)

# eliminate empty wells
CRISPR_reads=CRISPR_seq[, barcode_reads]
CRISPR_UMI=CRISPR_seq[, barcode_UMI]

pos_well=as.logical(as.vector(as.matrix(read.xlsx("Plate
map.xlsx", sheetIndex=3, header=F))))
CRISPR_reads=CRISPR_reads[, pos_well]
CRISPR_UMI=CRISPR_UMI[, pos_well]

CRISPR_reads_noERCC=CRISPR_reads[grep(",ERCC",
rownames(CRISPR_reads), invert=T),]

# separate out ERCC counts
CRISPR_UMI_noERCC=CRISPR_UMI[grep(",ERCC", rownames(CRISPR_UMI),
invert=T),]
CRISPR_UMI_ERCC=CRISPR_UMI[grep(",ERCC", rownames(CRISPR_UMI)),]
CRISPR_UMI_human=CRISPR_UMI_noERCC[grep("grch38_",
rownames(CRISPR_UMI_noERCC)),]
CRISPR_UMI_rat=CRISPR_UMI_noERCC[grep("rn5_",
rownames(CRISPR_UMI_noERCC)),]

# total gene detection
summary(rowSums(CRISPR_UMI_human)>0)

# range of gene detection in each sample
genes_detected=apply(CRISPR_UMI_human, 2, function(x)sum(x>0))
genes_detected=apply(CRISPR_UMI_rat, 2, function(x)sum(x>0))
UMI_total=colSums(CRISPR_UMI_human)
plot(UMI_total, genes_detected, pch=20)

summary(apply(CRISPR_UMI_human, 2, function(x)sum(x>0)))

write.table(matrix(genes_detected, nrow=16, byrow=F),
row.names=F, col.names=F, file="genes_detected", sep="\t")

# differential gene expression analysis using DESeq2
all_guide=read.xlsx("Plate map analysis.xlsx", sheetIndex=2,
header=F) # define metadata file directory
all_guide=all_guide[!is.na(all_guide)]

control=c("HDFN390_Ctrl001", "HDFN390_Ctrl002",
"HDFN390_Ctrl003", "HDFN390_Ctrl004", "HDFN390_Ctrl005",
"HDFN390_Ctrl007", "HDFN390_Ctrl009", "HDFN390_GFPg1",
"HDFN390_REF004")
guide_unique=unique(all_guide)[!(unique(all_guide) %in% control)]
```

```
guide_unique=guide_unique[grep("HDFN390", guide_unique)]

# for the other cell line diff analysis
# control=c("H1_36_Ctrl001", "H1_36_Ctrl002", "H1_36_Ctrl003",
"H1_36_Ctrl004", "H1_36_Ctrl005", "H1_36_Ctrl007",
"H1_36_Ctrl009", "H1_36_GFPg1", "H1_36_REF004")
# guide_unique=unique(all_guide)[!(unique(all_guide) %in%
control)]
# guide_unique=guide_unique[grep("H1_36", guide_unique)]


diff_analysis <-
  function(x){
    untreated = CRISPR_UMI_human[,all_guide == control[x]]
    for (guide in guide_unique) {
      treatment = CRISPR_UMI_human[,all_guide == guide]
      sample_names = c(colnames(untreated), colnames(treatment))
      condition = c(rep(control[x], length(all_guide[all_guide ==
control[x]])), rep(guide, length(all_guide[all_guide == guide])))
      colData=data.frame(condition, row.names=sample_names)
      dds <- DESeqDataSetFromMatrix(countData =
data.frame(untreated, treatment), colData = colData, design = ~
condition)
      dds$condition=relevel(dds$condition, ref=control[x])
      dds <- DESeq(dds, fitType='local')
      res <- results(dds)
      resOrdered <- res[order(res$padj),]
      sig_gene <- rownames(resOrdered[resOrdered$padj<0.05 &
!is.na(resOrdered$padj),])
      if (length(sig_gene) == 0){
        to_write <- c(guide, control[x], length(sig_gene))
      } else {
        to_write <- c(guide, control[x], length(sig_gene),
paste(unlist(strsplit(sig_gene, split=","))[seq(1,
2*length(sig_gene), 2)], collapse = " "))
      }
      write.table(as.list(to_write), quote=F, col.names=F,
row.names=x, sep=",", file=paste(guide, control[x], "gene.csv",
sep="_"))
      padj=ifelse(is.na(res$padj), 1, res$padj)
      write.table(resOrdered, quote=F, row.names=T, sep="\t",
file=paste(guide, control[x], "diff", sep="_"))
    }
  }

# make sure no NA in data frame
summary(apply(CRISPR_UMI_human, 2, function(x)any(is.na(x))))

lapply(seq(1:length(control), function(x) diff_analysis(x))
```

```
# in terminal
# cat *_gene.csv > diff_gene_list1.csv
# sort by first column in excel to rearrange
```