Target all TSS for CRISPRi

Timothy Daley

Recent work ([https://elifesciences.org/content/5/e19760] and [http://nar.oxfordjournals.org/content/44/18/e141]) has shown that TSS annotation from the FANTOM5 project (http://fantom.gsc.riken.jp/) improves CRISPRi sgRNA effeciency. We expect this to also hold in the case of CRISPRa, so we will use FANTOM5 TSS annotation in our CRISPRa libraries. Here we'll look how to target all TSS's.

FANTOM5 files can be found at [http://fantom.gsc.riken.jp/5/datafiles/phase1.3/extra/TSS_classifier/]. I remove the first line of the file.

```
TSS_human = read.table(file = "TSS_human.bed", sep = "\t")
head(TSS_human)
```

```
##      V1        V2        V3                 V4  V5 V6        V7        V8
## 1 chr10 100008587 100008589 p1@CU680531,0.1352 -89  + 100008587 100008589
## 2 chr10 100015362 100015397    p2@LOXL4,0.1291  55  - 100015362 100015397
## 3 chr10 100017518 100017519    p3@LOXL4,0.1842  13  - 100017518 100017519
## 4 chr10 100027943 100027958    p1@LOXL4,0.2200  48  - 100027943 100027958
## 5 chr10 100174900 100174956  p1@PYROXD2,0.2721   0  - 100174900 100174956
## 6 chr10 100174957 100174982  p2@PYROXD2,0.2448   0  - 100174957 100174982
##            V9
## 1 211,211,211
## 2 211,211,211
## 3  30,144,255
## 4  30,144,255
## 5  60,179,113
## 6  60,179,113
```

According to [https://elifesciences.org/content/5/e19760], we should look for identified peak on the same strand as the Ensembl TSS and is labeled p1@gene_name, p2@gene_name, and so on in column 4.

```
all_tss = c()
genes = scan("genes.txt", what = character())
for(x in genes){
  all_tss = rbind(all_tss, TSS_human[grep(paste0("@", x, ","), TSS_human[,4]), 1:6])
}
dim(all_tss)
```

```
## [1] 632   6
```

```
head(all_tss)
```

```
##         V1        V2        V3                 V4  V5 V6
## 96847 chr2 183006325 183006341 p30@PDE1A,0.2336  10  -
## 96848 chr2 183006380 183006392 p28@PDE1A,0.2416 -28  -
## 96849 chr2 183050734 183050735 p32@PDE1A,0.1916  66  -
## 96850 chr2 183050784 183050808 p21@PDE1A,0.1857   0  -
## 96851 chr2 183106741 183106754 p17@PDE1A,0.3037  18  -
## 96852 chr2 183106761 183106781 p13@PDE1A,0.3073   0  -
```

```
write.table(all_tss, file = "FANTOM5allTSS.bed", sep = "\t", quote = FALSE, row.names = FALSE, col.names
```

I used the UCSC liftover tool to convert hg19 to hg38, available at [https://genome.ucsc.edu/cgi-bin/hgLiftOver].]

```
sort -k 1,1 -k 2,2n hglft_genome_7ea8_9a5e70.bed > liftoverTSS.bed
```

```
liftoverTSS = read.table(file = "liftoverTSS.bed")
head(liftoverTSS, 1)
```

```
##     V1       V2       V3               V4 V5 V6
## 1 chr1 65792317 65792320 p80@PDE4B,0.3717  1  +
```

I'll use the midpoint of the region as the TSS.

```
start_relative2tss = 0
end_relative2tss = 250
tss_pos = floor(apply(liftoverTSS[,2:3], 1, mean))
strand = sapply(liftoverTSS[,6], function(x) if(x == "+"){return(1)} else{ return(-1)} )
start_pos = tss_pos + as.numeric(strand)*start_relative2tss + 1;
end_pos = tss_pos + as.numeric(strand)*end_relative2tss;

all_tss = data.frame(chrom = liftoverTSS[,1],
                     strand = sapply(liftoverTSS[,6],
                                     function(x) if(x == "+"){return(1)} else{ return(-1)} ),
                     gene = sapply(liftoverTSS[,4],
                                   function(x) gsub("\\,.*","", sub(".*@", "", x))),
                     start_pos = apply(cbind(start_pos, end_pos), 1, min),
                     end_pos = apply(cbind(start_pos, end_pos), 1, max)
                     )
# reorder by gene then by start position
all_tss = all_tss[order(all_tss$gene, all_tss$start_pos), ]


merge_tss = function(start_pos, end_pos, chrom, genes, strand){
  # assume positions are ordered
  # ensure all vectors are same length
  stopifnot(length(start_pos) == length(end_pos),
            length(end_pos) == length(chrom),
            length(chrom) == length(genes),
            length(genes) == length(strand));

  merged_regions = c();
  for(x in unique(genes)){
    regions = data.frame(start = start_pos[which(genes == x)],
                         end = end_pos[which(genes == x)],
                         chrom = chrom[which(genes == x)],
                         gene = genes[which(genes == x)],
                         strand = strand[which(genes == x)])
    if(dim(regions)[1] == 1){
      merged_regions = rbind(merged_regions, regions)
    }
```

```
    else{
      current_region = regions[1,];
      for(i in 2:dim(regions)[1]){
        next_region = regions[i,]
        if(next_region$start <= current_region$end){
          current_region$end = next_region$end;
        }
        else{
          merged_regions = rbind(merged_regions, current_region);
          current_region = next_region;
        }
      }
    }
  }
  return(merged_regions)
}
merged_regions = merge_tss(all_tss$start_pos, all_tss$end_pos, all_tss$chrom, all_tss$gene, all_tss$stra
dim(merged_regions)
```

```
## [1] 106   5
```

```
length(all_tss$gene)
```

```
## [1] 632
```

Looks like merging reduces the number of regions from ~600 to ~100. Now we'll get the sequences.

```
library(seqinr);
library(GenomicRanges)
library(BSgenome.Hsapiens.UCSC.hg38)
hg38 = BSgenome.Hsapiens.UCSC.hg38
wanted_ranges = GRanges(merged_regions$chrom, IRanges(apply(cbind(merged_regions$start,
                                                                  merged_regions$end), 1, min),
                                                       apply(cbind(merged_regions$start, merged_regions$
seqs = c()
for(i in 1:dim(merged_regions)[1]){
    seqs = c(seqs, getSeq(Hsapiens, wanted_ranges[i], as.character=TRUE))
}
wanted_seqs = list(genes = merged_regions$gene, seqs = seqs,
                   start = apply(cbind(merged_regions$start, merged_regions$end), 1, min),
                   end = apply(cbind(merged_regions$start, merged_regions$end), 1, max),
                   strand = merged_regions$strand, chrom = merged_regions$chrom)
write_seqs <- function(seqs, gene_names, chrom, start_pos, end_pos, strand, filename){
    stopifnot(dim(seqs)[1] == length(gene_names))
    write.fasta(file.out = filename, sequences = seqs[1], names = paste0(gene_names[1], "\t", chrom[1],
    if(length(gene_names) > 1){
        for(i in 2:length(gene_names)){
            write.fasta(file.out = filename, sequences = seqs[i], names = paste0(gene_names[i], "\t", ch
        }
  }
}
write_seqs(wanted_seqs$seqs, wanted_seqs$genes, wanted_seqs$chrom, wanted_seqs$start, wanted_seqs$end, w
```

Now I'll use the tool propose_sgRNAs that I wrote in C++ to extract all guides from the above regions, but excluding guides with trinucleotides (AAA, CCC, GGG, TTT) and those with enzyme cutting sequences given to me by Yanxia.

BstXI | CCANNNNNNTGG |
Blpl | GCTNAGC |
Xhol | CTCGAG |

```
~/sgRNA/sgRNAdesign/propose_sgRNAs -i regions_for_zhihua_12_20_2016.fa -V -R -T -c ~/sgRNA/Meng/enzyme_
while read gene; do
n_lines="$(grep ${gene} guides_for_zhihua_12_20_2016.txt | wc -l)";
printf "%s\t%s\n" "${gene}" "${n_lines}";
done < genes.txt

wc -l guides_for_zhihua_12_20_2016.txt
```

```
##     3074 guides_for_zhihua_12_20_2016.txt
```

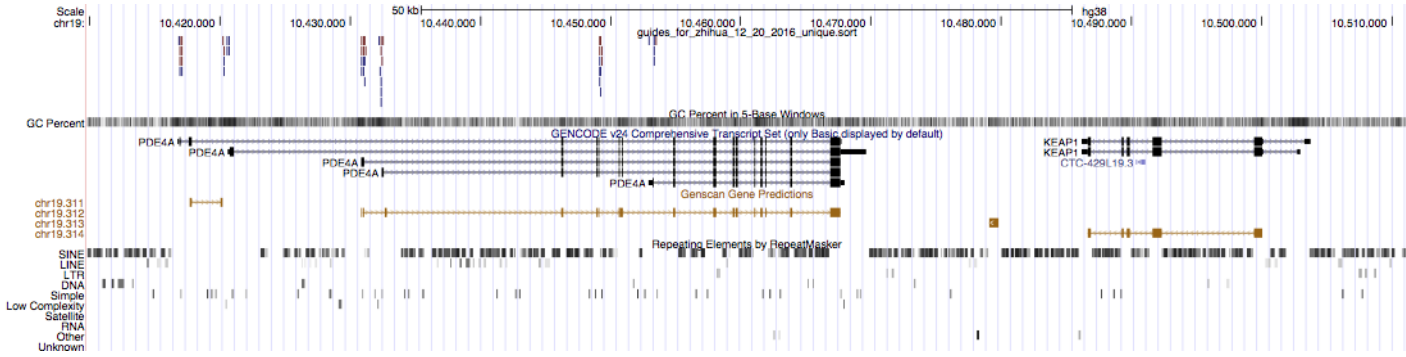
The guides were then mapped to the genome with bowtie2. Guides that mapped more than once were flagged with the XS flag and were removed.

```
~/scratch/programs/aligners/bowtie2/bowtie2-2.2.7/bowtie2 -f -a -x ~/scratch/genomes/hg38/hg38 -U guide
1533 reads; of these:
  1533 (100.00%) were unpaired; of these:
    0 (0.00%) aligned 0 times
    854 (55.71%) aligned exactly 1 time
    679 (44.29%) aligned >1 times
100.00% overall alignment rate
~/scratch/programs/samtools-1.3/samtools view -S guides_for_zhihua_12_20_2016.sam | grep --invert-match
```

```
cut -f 1 guides_for_zhihua_12_20_2016_unique.sam | sort  | uniq -c
```

```
##    3 PDE10A
##   34 PDE11A
##   30 PDE1A
##   43 PDE1B
##   55 PDE1C
##   78 PDE2A
##   64 PDE3A
##   53 PDE4A
##  139 PDE4B
##   61 PDE4C
##   34 PDE5A
##    5 PDE6A
##   61 PDE6B
##   26 PDE6D
##   12 PDE6G
##   14 PDE7A
##    7 PDE7B
##   52 PDE8A
##   41 PDE8B
##   42 PDE9A
```

Let's make sure these are designed right in the genome browser.



That looks good, but unfortunately bowtie2 outputs the forward strand for reads that map to the reverse complement. Reads that map to the forward strand are fine.

```
~/scratch/programs/samtools-1.3/samtools view -F 0x10 guides_for_zhihua_12_20_2016_unique.sort.bam | cu
~/scratch/programs/samtools-1.3/samtools view -f 0x10 guides_for_zhihua_12_20_2016_unique.sort.bam | cu
```

I'll use python to reverse complement the sequences of the reverse reads.

```
import csv
complement = {'A': 'T', 'C': 'G', 'G': 'C', 'T': 'A'}
with open('guides_for_zhihua_12_20_2016_reverse.txt','r') as f:
  reader=csv.reader(f,delimiter='\t')
  with open('guides_for_zhihua_12_20_2016_reverse_complement.txt', 'w') as out_f:
    #writer = csv.writer(out_f, delimiter='\t')
    for gene,chrom,pos,seq in reader:
      bases = list(seq)
      bases = reversed([complement.get(base,base) for base in bases])
      bases = ''.join(bases)
      out_f.writelines(''.join(gene) + '\t' + ''.join(chrom) + '\t' + ''.join(pos) + '\t' + ''.join(bas
  out_f.close()
```

```
cat guides_for_zhihua_12_20_2016_forward.txt guides_for_zhihua_12_20_2016_reverse_complement.txt | sort
head guides_for_zhihua_12_20_2016_unique_rev_comp.txt
```

```
## PDE4B    chr1    65792338    GCGAGTGACTGACACGTTCC
## PDE4B    chr1    65792339    CGAGTGACTGACACGTTCCA
## PDE4B    chr1    65792401    GTGTAGTGGCAGACGGCCGC
## PDE4B    chr1    65792402    TGTAGTGGCAGACGGCCGCT
## PDE4B    chr1    65792509    TGTGCGTAATCCTTCAGCTC
## PDE4B    chr1    65792512    GCGTAATCCTTCAGCTCTGG
## PDE4B    chr1    65792518    TCCTTCAGCTCTGGTGGTAA
## PDE4B    chr1    65792519    CTTACCACCAGAGCTGAAGG
## PDE4B    chr1    65792603    CCTCCTGCAATATTCCGCGG
## PDE4B    chr1    65792609    AATATTGCAGGAGGTCTGTG
```