Untitled

Helen Lindsay

29 January 2016

Supplementary Note 2: Sequencing errors and alignment uncertainty affect

variant count, placement and size

Analyses with Burger *et al* Sanger data

AVAILABILITY OF SANGER DATA

In this section, we use a pre-generated transcript database (txdb) to access structures of the transcripts overlapping the guide. The txdb was generated from a gtf file downloaded from Ensembl. For more information about generating a txdb, see the R Bioconductor package GenomicFeatures.

```r
library("CrispRVariants")
library("ShortRead")
library("rtracklayer")
library("BSgenome.Drerio.UCSC.danRer7")
library("gdata")
library("GenomicFeatures")
library("grid")

ind <- "../idx/danRer7.fa"

# Import guides, add 5 bases to each end
gd <- rtracklayer::import("../annotation/Burger_Sanger_guides.bed")
gdl <- resize(gd, width(gd) + 10, fix = "center")
names(gdl) <- gdl$name

# Get the reference sequences from the genome
danRer7 <- BSgenome.Drerio.UCSC.danRer7
refs <- getSeq(danRer7, gdl)
names(refs) <- names(gdl)

txdb <- loadDb("~/zebrafish_txdb.sqlite")

ab1ToFastq <- function(raw, fdir, recall = TRUE){
  # List files ending in .ab1 in the raw directory
  dir.create(fdir, showWarnings = FALSE)
  ab1s <- dir(raw, pattern="ab1$", recursive = TRUE)
  # Get sequence names from filenames
  sns <- gsub(".ab1","",basename(ab1s))

  if (all(dirname(ab1s) == ".")){
    # No subdirectory
    fqs <- paste0(basename(dirname(fastq)), ".fastq")
  } else {
    # Name after subdirectory
    fqs  <- paste0(gsub("[\ |\\/]", "_", dirname(ab1s)), ".fastq")
  }
  fqs <- file.path(fdir, fqs)
  ab1s <- file.path(raw, ab1s)

  # Convert to fastq, optionally recalling bases
  dummy <- mapply(function(u,v,w,rc) {
    print(c(u,v,w))
    suppressWarnings(CrispRVariants::abifToFastq(u,v,w, recall = rc))
  }, sns, ab1s, fqs, recall)

  fqs <- unique(fqs)
  fqs
}


runMapping <- function(fqs, bdir, ind){
  # Map a list of fastq files with bwa mem, sort and index
  dir.create(bdir, showWarnings = FALSE)
  bms <- gsub(".fastq$",".bam",basename(fqs))
  sbms <- file.path(bdir, gsub(".bam","_s",bms))
  for(i in 1:length(fqs)) {
    cmd <- paste0("bwa mem ", ind, " ",
                  fqs[i]," | samtools view -Sb - > ", bms[i])
      system(cmd)
      indexBam(sortBam(bms[i],sbms[i]))
      unlink(bms[i])
  }
  sbms <- unique(paste0(sbms, ".bam"))
  sbms
}
```

Sequencing errors in *tbx16*

```
raw <- "../Burger_Sanger_data/spt_ccA_F1/ab1"
fastq <- "../Burger_Sanger_data/spt_ccA_F1/fastq"
bam <- "../Burger_Sanger_data/spt_ccA_F1/bam"

fqs <- ab1ToFastq(raw, fastq)
```

```
## [1] "AB1025"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_11_ph/AB1025.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_11_ph.fastq"
## [1] "AB1026"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_11_ph/AB1026.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_11_ph.fastq"
## [1] "AB1027"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_11_ph/AB1027.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_11_ph.fastq"
## [1] "AB1028"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_11_ph/AB1028.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_11_ph.fastq"
## [1] "AB1029"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_11_ph/AB1029.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_11_ph.fastq"
## [1] "AB1030"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_11_ph/AB1030.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_11_ph.fastq"
## [1] "AB1031"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_11_ph/AB1031.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_11_ph.fastq"
## [1] "AB1032"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_11_ph/AB1032.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_11_ph.fastq"
## [1] "AB1033"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_12_ph/AB1033.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_12_ph.fastq"
## [1] "AB1034"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_12_ph/AB1034.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_12_ph.fastq"
## [1] "AB1035"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_12_ph/AB1035.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_12_ph.fastq"
## [1] "AB1036"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_12_ph/AB1036.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_12_ph.fastq"
## [1] "AB1037"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_12_ph/AB1037.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_12_ph.fastq"
## [1] "AB1038"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_12_ph/AB1038.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_12_ph.fastq"
## [1] "AB1039"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_12_ph/AB1039.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_12_ph.fastq"
## [1] "AB1040"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_12_ph/AB1040.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_12_ph.fastq"
## [1] "AB1049"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/AB1049.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph.fastq"
## [1] "AB1050"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/AB1050.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph.fastq"
## [1] "AB1051"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/AB1051.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph.fastq"
## [1] "AB1052"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/AB1052.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph.fastq"
## [1] "AB1053"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/AB1053.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph.fastq"
## [1] "AB1054"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/AB1054.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph.fastq"
## [1] "AB1055"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/AB1055.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph.fastq"
## [1] "AB1056"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/AB1056.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph.fastq"
## [1] "AB1057"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/embryo 8/AB1057.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph_embryo_8.fastq"
## [1] "AB1058"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/embryo 8/AB1058.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph_embryo_8.fastq"
## [1] "AB1059"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/embryo 8/AB1059.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph_embryo_8.fastq"
## [1] "AB1060"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/embryo 8/AB1060.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph_embryo_8.fastq"
## [1] "AB1061"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/embryo 8/AB1061.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph_embryo_8.fastq"
## [1] "AB1062"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/embryo 8/AB1062.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph_embryo_8.fastq"
## [1] "AB1063"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/embryo 8/AB1063.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph_embryo_8.fastq"
## [1] "AB1064"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_14_ph/embryo 8/AB1064.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph_embryo_8.fastq"
## [1] "IM2009"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_5_wt/IM2009.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_5_wt.fastq"
## [1] "IM2010"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_5_wt/IM2010.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_5_wt.fastq"
## [1] "IM2011"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_5_wt/IM2011.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_5_wt.fastq"
## [1] "IM2012"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_5_wt/IM2012.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_5_wt.fastq"
## [1] "IM2013"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_5_wt/IM2013.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_5_wt.fastq"
## [1] "IM2014"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_5_wt/IM2014.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_5_wt.fastq"
## [1] "IM2015"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_5_wt/IM2015.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_5_wt.fastq"
## [1] "IM2016"
## [2] "../Burger_Sanger_data/spt_ccA_F1/ab1/spt_ccA_F1_5_wt/IM2016.ab1"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_5_wt.fastq"
```

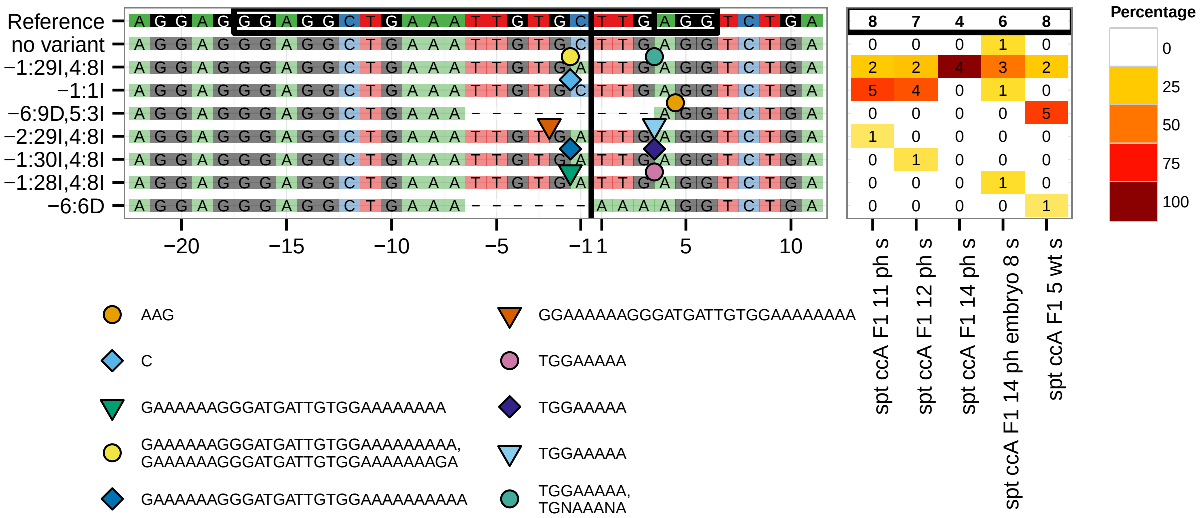
```r
print(fqs)
```

```
## [1] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_11_ph.fastq"
## [2] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_12_ph.fastq"
## [3] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph.fastq"
## [4] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_14_ph_embryo_8.fastq"
## [5] "../Burger_Sanger_data/spt_ccA_F1/fastq/spt_ccA_F1_5_wt.fastq"
```

```r
bms <- runMapping(fqs, bam, ind)
nms <- gsub(".bam", "", gsub("_", " ", basename(bms)))

guide <- gdl["spt_ccA"]
ref <- refs["spt_ccA"]

cset <- readsToTarget(bms, target = guide, reference = ref,
            names = nms, target.loc = 22, verbose = FALSE)

print(cset)
```

```
## CrisprSet object containing 5 CrisprRun samples
## Target location:
## GRanges object with 1 range and 2 metadata columns:
##           seqnames               ranges strand |        name     score
##              <Rle>            <IRanges>  <Rle> | <character> <numeric>
##   spt_ccA     chr8 [54029513, 54029545]      + |     spt_ccA         0
##   -------
##   seqinfo: 2 sequences from an unspecified genome; no seqlengths
## [1] "Most frequent variants:"
##             spt ccA F1 11 ph s spt ccA F1 12 ph s spt ccA F1 14 ph s
## no variant                   0                  0                  0
## -1:29I,4:8I                  2                  2                  4
## -1:1I                        5                  4                  0
## -6:9D,5:3I                   0                  0                  0
## -1:30I,4:8I                  0                  1                  0
## -1:28I,4:8I                  0                  0                  0
##             spt ccA F1 14 ph embryo 8 s spt ccA F1 5 wt s
## no variant                            1                 0
## -1:29I,4:8I                           3                 2
## -1:1I                                 1                 0
## -6:9D,5:3I                            0                 5
## -1:30I,4:8I                           0                 0
## -1:28I,4:8I                           1                 0
```

```r
plotVariants(cset,
             plotAlignments.args = list(max.insertion.size = 50,
                                        legend.cols = 2),
             plotFreqHeatmap.args = list(
               legend.key.height = grid::unit(1, "lines")))
```

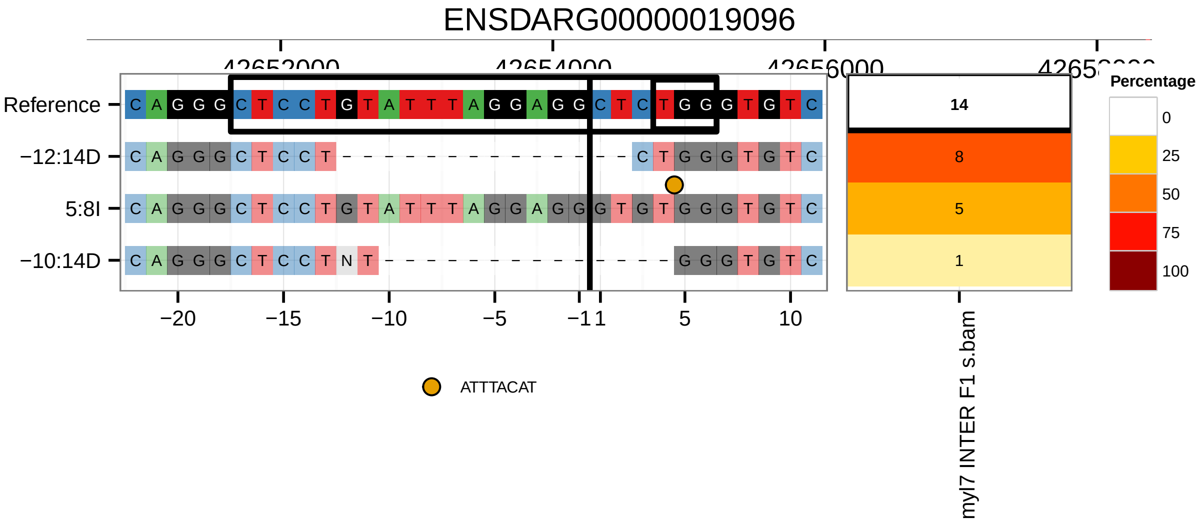


```
## TableGrob (2 x 1) "arrange": 2 grobs
##   z     cells    name              grob
## 1 1 (1-1,1-1) arrange rect[GRID.rect.11]
## 2 2 (2-2,1-1) arrange    gtable[arrange]
```

Alignment uncertainty in *myl7*

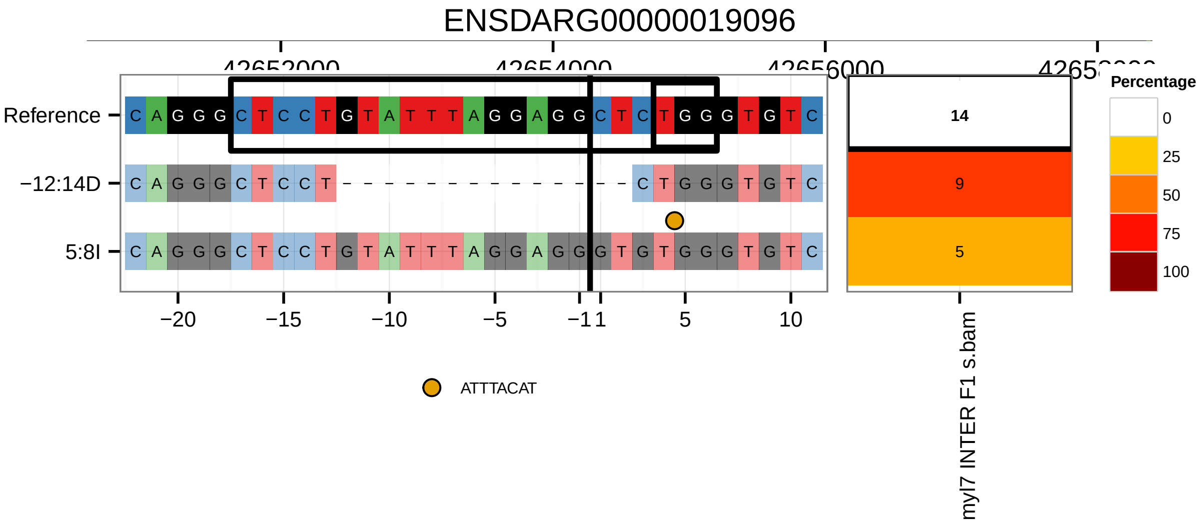
```
raw <- "../Burger_Sanger_data/myl7_INTER_F1/ab1"
fastq <- "../Burger_Sanger_data/myl7_INTER_F1/fastq"
bam <- "../Burger_Sanger_data/myl7_INTER_F1/bam"

guide <- gdl["myl7_5"]
ref <- refs["myl7_5"]

ab1ToPlot <- function(raw, fastq, bam, guide, ref, recall){
  fqs <- ab1ToFastq(raw, fastq, recall = recall)
  bms <- runMapping(fqs, bam, ind)
  nms <- gsub("_s.bam", "", gsub("_", " ", basename(bms)))

  cset <- readsToTarget(bms, target = guide, reference = ref,
           names = nms, target.loc = 22, verbose = FALSE)

  print(cset)
  plotVariants(cset, txdb = txdb)

  # Remove the fastq and bam files so the reads are
  # not duplicated at the next function call
  unlink(fastq, recursive = TRUE)
  unlink(bam, recursive = TRUE)
}

# First extract ab1 files without base recalibration
dummy <- ab1ToPlot(raw, fastq, bam, guide, ref, recall = FALSE)
```

```
## [1] "CH141"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH141.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH142"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH142.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH143"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH143.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH144"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH144.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH146"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH146.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH147"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH147.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH148"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH148.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH149"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH149.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH150"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH150.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH151"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH151.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH152"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH152.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH154"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH154.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH155"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH155.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH156"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH156.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## CrisprSet object containing 1 CrisprRun samples
## Target location:
## GRanges object with 1 range and 2 metadata columns:
##          seqnames               ranges strand |        name     score
##             <Rle>            <IRanges>  <Rle> | <character> <numeric>
##   myl7_5     chr8 [42658358, 42658390]      - |      myl7_5         0
##   -------
##   seqinfo: 2 sequences from an unspecified genome; no seqlengths
## [1] "Most frequent variants:"
##         myl7 INTER F1 s.bam
## -12:14D                   8
## 5:8I                      5
## -10:14D                   1
## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
```



```r
# Then with recalibration
dummy <- ab1ToPlot(raw, fastq, bam, guide, ref, recall = TRUE)
```

```
## [1] "CH141"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH141.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH142"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH142.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH143"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH143.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH144"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH144.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH146"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH146.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH147"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH147.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH148"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH148.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH149"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH149.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH150"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH150.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH151"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH151.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH152"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH152.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH154"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH154.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH155"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH155.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## [1] "CH156"
## [2] "../Burger_Sanger_data/myl7_INTER_F1/ab1/CH156.ab1"
## [3] "../Burger_Sanger_data/myl7_INTER_F1/fastq/myl7_INTER_F1.fastq"
## CrisprSet object containing 1 CrisprRun samples
## Target location:
## GRanges object with 1 range and 2 metadata columns:
##          seqnames               ranges strand |        name     score
##             <Rle>            <IRanges>  <Rle> | <character> <numeric>
##   myl7_5     chr8 [42658358, 42658390]      - |      myl7_5         0
##   -------
##   seqinfo: 2 sequences from an unspecified genome; no seqlengths
## [1] "Most frequent variants:"
##         myl7 INTER F1 s.bam
## -12:14D                   9
## 5:8I                      5

## 'select()' returned 1:many mapping between keys and columns
## 'select()' returned 1:many mapping between keys and columns
```



Analyses with Burger *et al* MiSeq data

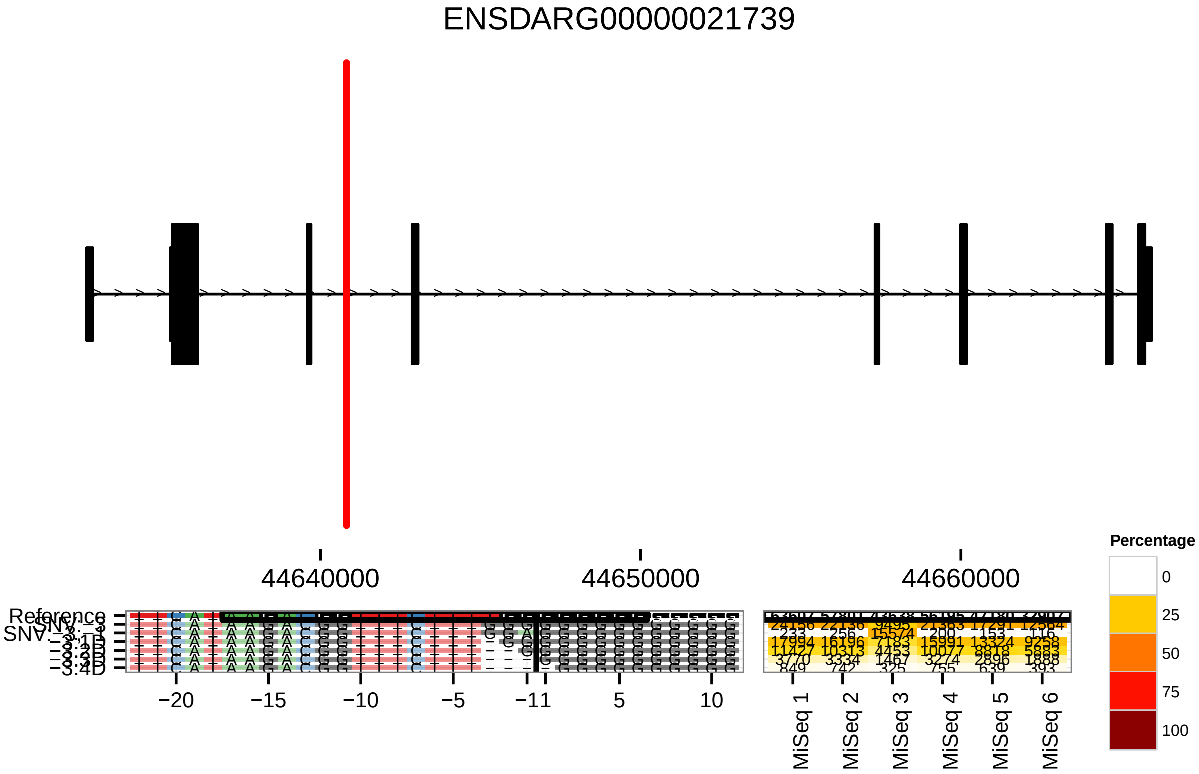
The Burger *et al* MiSeq data is available from ArrayExpress with accession number E-MTAB-4143. Here we assume that the bam files have been download into a directory named “Burger_MiSeq_data”. These are paired-end 250bp reads.

```
bam_dir <- "../Burger_MiSeq_data"
bams <- list.files(bam_dir, pattern = "*.bam$", full.names = TRUE)
target <- rtracklayer::import("../annotation/Burger_MiSeq_guides.bed")
target <- target[target$name == "xirp1_off2"]
target <- target + 5
reference <- getSeq(danRer7, target)
```

```
## Warning in .Seqinfo.mergexy(x, y): Each of the 2 combined objects has sequence levels not in the othe
##   - in 'x': chr4, chr7, chr10, chr15, chr16, chr21, chr23, chr24, chr25, chrM
##   - in 'y': EGFP_ORF
##   Make sure to always combine/compare objects based on the same reference
##   genome (use suppressWarnings() to suppress this warning).
```

```
# Note: getSeq returns a DNAStringSet, we must select the first
# sequence to transform it into a DNAString

sample_names <- paste("MiSeq", c(1:6))

# Set collapse.pairs = TRUE to count pairs correctly
cset <- readsToTarget(bams, target = target, reference = reference[[1]],
                      target.loc = 22, names = sample_names,
                      collapse.pairs = TRUE, verbose = FALSE)
plotVariants(cset, txdb = txdb, row.ht.ratio = c(3,1), col.wdth.ratio = c(1.5,1),
             plotAlignments.args = list(min.freq = 1),
             plotFreqHeatmap.args = list(min.freq = 1))
```

```
## Warning in .Seqinfo.mergexy(x, y): Each of the 2 combined objects has sequence levels not in the othe
##   - in 'x': EGFP_ORF
##   - in 'y': chr4, chr7, chr10, chr15, chr16, chr21, chr23, chr24, chr25, chrM, Zv9_NA1, Zv9_NA10, Zv
```



```
## TableGrob (2 x 1) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange  gtable[layout]
## 2 2 (2-2,1-1) arrange gtable[arrange]
```

Supplementary Note 8: Chimeric alignments include genuine variants

and sequencing errors

Here we use the data from Burger *et al* and Cho *et al*. The data from Cho *et al* is available from the DNA Data Bank of Japan under accession DRA001195. We assume the Cho off-target data (i.e. not exome data) has been downloaded, extracted to fastq and mapped, and that the mapped bam files are located in a directory named "Cho_data". The Cho data is for human cells. We have reformatted the metadata describing the Cho samples (Supplementary Table 2 intheir paper) and include this in the annotation folder.

Functions used in chimera analyses

read_alns

This is a wrapper for GenomicAlignments::readGAlignments setting some parameters

```
read_alns <- function(fname){
  readGAlignments(fname, param = ScanBamParam(what = c("seq", "flag")),
     use.names = TRUE)
}
```

get_chimeras

```
get_chimeras <- function(alns, guides, expand.guide = 20){
  # Get the chimeric reads
  ch_idxs <- CrispRVariants::findChimeras(alns, by.flag = TRUE)
  ch <- alns[ch_idxs]

  # Check that the chimeric region is near the guide
  ex_gd <- guides + expand.guide
  # Exclude reads where the guide + some flanking sequence is contained
  # within a chimeric segment
  guide_within <- subjectHits(findOverlaps(ex_gd, ch,
                              ignore.strand = TRUE, type = "within"))

  ordrd <- unique(guide_within[order(guide_within)])
  non_ch <- names(ch) %in% names(ch)[ordrd]
  ch_idxs <- ch_idxs[!non_ch]
  ch <- alns[ch_idxs]

  # Get combinations of cigar strings for chimeric alignment sets
  partition <- cumsum(rle(names(ch))$lengths)
  cigs <- relist(cigar(ch), PartitioningByEnd(partition))
  cigs <- lapply(cigs, paste, collapse = ",")
  uniq_combs <- ! duplicated(cigs)

  # Return one read set per chimera combination
  ch <- unlist(relist(ch, PartitioningByEnd(partition))[as.vector(uniq_combs)])
  ch
}
```

select_chimeras

This function filters candidate chimeric reads by maximum mapped range and number of chromosomes spanned (all chimeric segments must map to a single chromosome).

```
select_chimeras <- function(alns){
    partition <- cumsum(rle(names(alns))$lengths)
    alnsl <- relist(alns, PartitioningByEnd(partition))

    # Remove groups spanning multiple chromosomes.
    # This will exclude insertions mapped to another chr, these are rare
    sq_nms <- unique(seqnames(alnsl))
    keep <- elementLengths(sq_nms) == 1
    alnsl <- alnsl[keep]

    singlechr_len <- length(unlist(alnsl))
    sq_nms <- sq_nms[keep]
    # Get ranges
    gp_rngs <- GRanges(as.character(sq_nms), IRanges(min(start(alnsl)), max(end(alnsl))))

    # Remove alns spanning more than 1000, these are probably pcr errors
    keep <- width(gp_rngs) <= 1000
    alnsl <- alnsl[keep]
    gp_rngs <- gp_rngs[keep]
    result <- list(alns = alnsl, gp_rngs = gp_rngs)
    result
}
```

dist_to_cut

Finds the minimum distance between any member of a chimeric read set and a CRISPR cut site.

```
dist_to_cut <- function(alns_list, cut_sites){
    # Accepts a list of alignments split by chimeric group
    all_alns <- as(unlist(alns_list), "GRanges")
    dists <- distanceToNearest(all_alns, cut_sites, ignore.strand = TRUE)
    dists <- min(relist(mcols(dists)$distance, alns_list))
    as.vector(dists)
}
```

make_violin_plot

This makes violin plots showing the distibution of on- and off-target chimeric reads.

```
make_violin_plot <- function(dat){
    p <- ggplot(dat, aes(x = cond, y = dist)) + geom_violin(fill = "gray") +
        theme_bw() + ylab("Distance to nearest cut site (bases)") +
        theme(axis.title.x=element_blank())
    return(p)
}
```

mappedRangeByTarget

This function filters chimeric read alignment sets by the aligned range. The aligned range must be nearly identical to the PCR primers for the amplicon. This is done to try and avoid PCR chimeras. We only perform this step for the Burger data, as we do not have sufficient information to do the same for the Cho data. Returns a list of chimeras for each PCR range.

```
mappedRangeByTarget <- function(alns, targets, maxgap = 5){
  # Only keep reads where the range of the aligned segments is
  # nearly equal to the primer range.  Note this excludes segments
  # where an insertion maps outside of the amplicon, but endpoints will be retained

  temp <- select_chimeras(alns)
  alnsl <- temp$alns
  gp_rngs <- temp$gp_rngs

  hits <- findOverlaps(gp_rngs, targets + maxgap, type = "within")
  wdth_diff <- abs(width(targets[subjectHits(hits)]) - width(gp_rngs[queryHits(hits)]))
  hits <- hits[wdth_diff <= 2*maxgap]

  result <- vector("list", length(targets))
  splits <- split(queryHits(hits), subjectHits(hits))

  for (nm in names(splits)){
    result[as.integer(nm)] <- unlist(alnsl[splits[[nm]]])
  }
  if ("name" %in% names(mcols(targets))) names(result) <- targets$name

  return(result)
}
```

getChByTgt

Select chimeras matching each PCR primer range, only used for Burger data.

```
getChByTgt <- function(bams, guides, target_ranges){
    alns_by_tgt <- bplapply(bams, function(bam){
      alns <- read_alns(bam)
      # Cut down size by first selecting chimeric reads that overlap the primer ranges
      # (redundant with filtering in mappedRangeByTarget)
      ch_idxs <- CrispRVariants::findChimeras(alns, by.flag = TRUE)
      alns <- alns[ch_idxs]
      hits_pcr <- names(alns)[queryHits(findOverlaps(alns, target_ranges))]
      alns <- alns[names(alns) %in% hits_pcr]

      # Now exclude guides completely contained within one segment
      alns <- get_chimeras(alns, guides)

      result <- mappedRangeByTarget(alns, targets = target_ranges)
      names(result) <- target_ranges$name
      result
      }, BPPARAM = BiocParallel::MulticoreParam(6))
    alns_by_tgt
}
```

Chimera analysis with Cho data

```
library("BiocParallel")
library("BSgenome.Hsapiens.UCSC.hg19")
library("CrispRVariants")
library("gdata")
library("GenomicRanges")
library("GenomicAlignments")
library("ggplot2")
library("gridExtra")
library("Rsamtools")
library("rtracklayer")
```

```
hg19 <- BSgenome.Hsapiens.UCSC.hg19
cho <- gdata::read.xls("../annotation/Cho_Table2_reformatted.xls", header = TRUE)
guides <- GenomicRanges::GRanges(cho$chromosome,
                                 IRanges(cho$start + 1, width = 23), strand = cho$strand)
guide_seqs <- getSeq(hg19, guides)
cho_guides <- as(cho$guide, "DNAStringSet")
all.equal(as.character(guide_seqs), as.character(cho_guides))
```

```
## [1] TRUE
```

```
is_ontarget <- grepl("OnTarget", cho$X)
#cut_sites <- narrow(guides, start = 17, end = 17)
cut_sites <- resize(resize(guides, 6, fix = "end"), 1, fix = "start")

cho_bam_files <- list.files("../Cho_data", pattern = "*.bam", full.names = TRUE)
print(cho_bam_files)
```

```
## [1] "../Cho_data/DRR014240_s.bam" "../Cho_data/DRR014241_s.bam"
## [3] "../Cho_data/DRR014249_s.bam"
```

```
alns <- lapply(cho_bam_files, read_alns)
alns <- lapply(alns, get_chimeras, guides = guides)
alns <- do.call(c, alns)

temp <- select_chimeras(alns)
alnsl <- temp$alns
gp_rngs <- temp$gp_rngs


# Keep alignments that only overlap one guide, divide into on- and off-target
hits <- findOverlaps(gp_rngs, guides)
```

```
## Warning in .Seqinfo.mergexy(x, y): Each of the 2 combined objects has sequence levels not in the othe
##   - in 'x': chr17, chr19, chr18, chrX
##   - in 'y': chr16
##   Make sure to always combine/compare objects based on the same reference
##   genome (use suppressWarnings() to suppress this warning).
```

```
unq_hits <- ! duplicated(queryHits(hits)) & ! duplicated(queryHits(hits), fromLast = TRUE)

unq_on <- unq_hits & is_ontarget[subjectHits(hits)]
unq_off <- unq_hits & !is_ontarget[subjectHits(hits)]

cho_on <- alnsl[queryHits(hits)[unq_on]]
cho_off <- alnsl[queryHits(hits)[unq_off]]

cho_off_dists <- dist_to_cut(cho_off, cut_sites)
cho_on_dists <- dist_to_cut(cho_on, cut_sites)

cho_dat <- data.frame(dist = c(cho_on_dists, cho_off_dists),
  cond = rep(c("On target","Off target"),
             c(length(cho_on_dists), length(cho_off_dists))))

print(sprintf("On target %s \n Off target %s\n",
              length(cho_on_dists), length(cho_off_dists)))
```

```
## [1] "On target 1142 \n Off target 158\n"
```

```
p1 <- make_violin_plot(cho_dat)
p1 <- p1 + ggtitle("Cho")
```

Chimera analysis with Burger data

```
bam_dir <- "../Burger_MiSeq_data"
bams <- list.files(bam_dir, pattern = "*.bam$", full.names = TRUE)
guides <- rtracklayer::import("../annotation/Burger_MiSeq_guides.bed")
pcr_ranges <- import(file.path("../annotation/Burger_MiSeq_primer_ranges.bed"))
gd_to_primer <- read.table("../annotation/Burger_MiSeq_layout.txt", sep = "\t",
                           header = TRUE)

# In this experiment the lcr guides were multiplexed and the hand2 guides share
# PCR primers, so we'll remove these.
guides <- guides[!grepl("hand2_ccA$|hand2_ccB$|lcr", guides$name)]
pcr_ranges <- pcr_ranges[!grepl("hand2$|lcr", pcr_ranges$name)]
gd_to_primer <- gd_to_primer[!grepl("hand2$|lcr", gd_to_primer$primer), ]

cut_sites <- narrow(guides, start = 17, end = 17)
is_ontarget <- ! grepl("off", guides$name)

ch_by_primer <- getChByTgt(bams, guides, pcr_ranges)

ch_by_gd <- apply(gd_to_primer, 1, function(rw){
  fn <- rw["primer"]
  gd <- rw[["guide"]]
  gidx <- guides$name == gd
  guide <- guides[gidx]
  cut_site <- cut_sites[gidx]
  idxs <- as.numeric(strsplit(rw["samples"], ",")[[1]])
  # Pull out chimeras for this guide for the correct samples
  all_alns <- lapply(idxs, function(idx) ch_by_primer[[idx]][[gd]])
  all_dists <- lapply(all_alns, function(alns, guides){
    if (length(alns) == 0) return(list())
    ch <- get_chimeras(alns, guides)
    ch <- split(ch, names(ch))
    dists <- dist_to_cut(ch, cut_site)
    }, guides = guide)
  dists <- do.call(c, all_dists)
  dists
})

burger_on_target <- unlist(ch_by_gd[is_ontarget], use.names = FALSE)
burger_off_target <- unlist(ch_by_gd[!is_ontarget], use.names = FALSE)
```

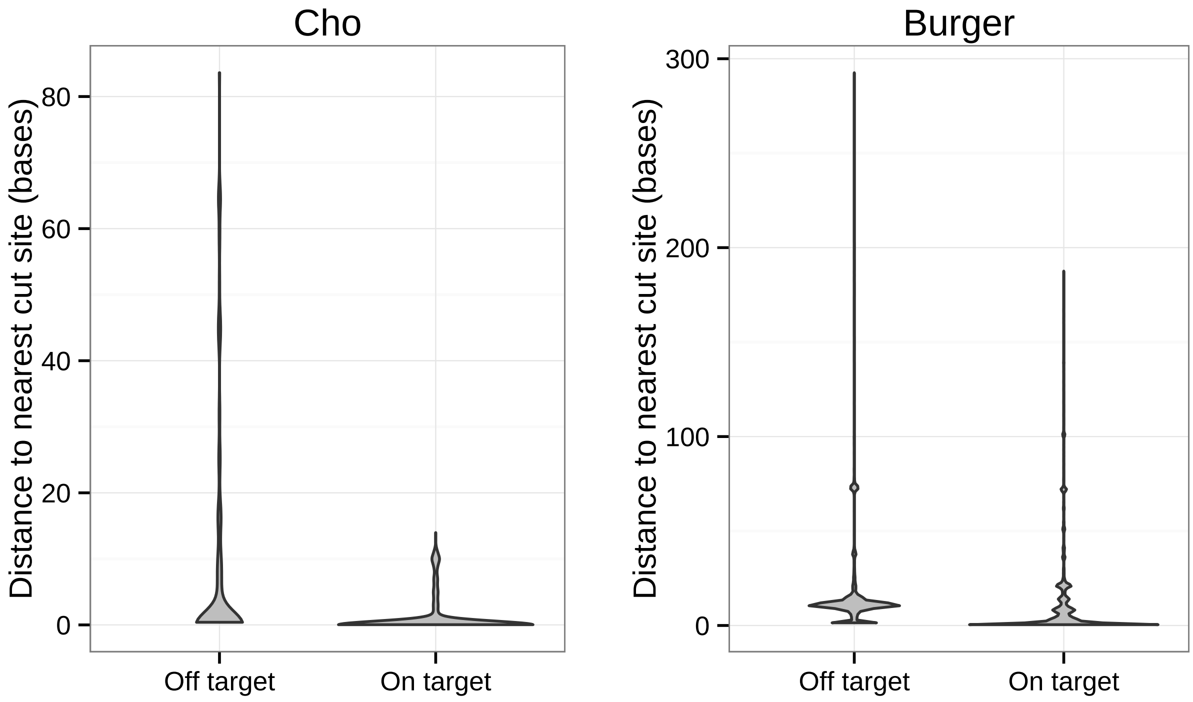
```
dat <- data.frame(dist = c(burger_on_target, burger_off_target),
cond = rep(c("On target","Off target"), c(length(burger_on_target),
           length(burger_off_target))))

print(sprintf("On target %s \n Off target %s\n", length(burger_on_target),
       length(burger_off_target)))
```

```
## [1] "On target 11386 \n Off target 18252\n"
```

```
p2 <- make_violin_plot(dat)
p2 <- p2 + ggtitle("Burger")
```

Combine chimera plots

```
gridExtra::grid.arrange(p1,p2, ncol=2)
```


Cho
Distance to nearest cut site (bases)
Off target
On target
Burger
Distance to nearest cut site (bases)
Off target
On target