Bcell_receptor_analysis

Sedreh

5/10/2019

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
library(seqinr)
```

```
##
## Attaching package: 'seqinr'
```

```
## The following object is masked from 'package:dplyr':
##
##     count
```

```
library(Biostrings)
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter,
##     Find, get, grep, grepl, intersect, is.unsorted, lapply, Map,
##     mapply, match, mget, order, paste, pmax, pmax.int, pmin,
##     pmin.int, Position, rank, rbind, Reduce, rownames, sapply,
##     setdiff, sort, table, tapply, union, unique, unsplit, which,
##     which.max, which.min
```

```
## Loading required package: S4Vectors
```

```
## Loading required package: stats4
```

```
##
## Attaching package: 'S4Vectors'
```

```
## The following objects are masked from 'package:data.table':
##
##     first, second
```

```
## The following object is masked from 'package:tidyr':
##
##     expand
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, rename
```

```
## The following object is masked from 'package:base':
##
##     expand.grid
```

```
## Loading required package: IRanges
```

```
##
## Attaching package: 'IRanges'
```

```
## The following object is masked from 'package:data.table':
##
##     shift
```

```
## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice
```

```
## Loading required package: XVector
```

```
##
## Attaching package: 'Biostrings'
```

```
## The following object is masked from 'package:seqinr':
##
##     translate
```

```
## The following object is masked from 'package:base':
##
##     strsplit
```

```
library(ggplot2)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
library(stringi)
library(bcRep)
```

```
########################################
#processing 10x genomics filtered dataset
########################################
```

```
#load the dataset
filtered_data = read.csv ('/home/sedreh/ITMO/semester2/Bcellsproject/final_Rcode/PBMC
s_of_a_healthy_donor/vdj_v1_hs_pbmc_b_filtered_contig_annotations.csv')
```

```r
#As we have several copies of each barcode in the file, in ordre to count the number
 of cells we need to count just one copy of unique barcoe

barcode_summary <- function(filtered_data)
{
  number_of_cells <- filtered_data %>%
    distinct(barcode) %>%
    dplyr::count()

number_of_cells$n
}
```

```r
#look at to the quality of chains

count_table <- function(filtered_data)
{
v_gene_and_j_gene <- filter(filtered_data, v_gene != 'None' & j_gene != 'None')
v_gene_and_j_gene <- count(v_gene_and_j_gene)$n

no_v_gene_and_no_j_gene <- filter(filtered_data, v_gene == 'None' & j_gene == 'None')
no_v_gene_and_no_j_gene <- count(no_v_gene_and_no_j_gene)$n

v_gene_or_j_gene <- filter(filtered_data, v_gene != 'None' | j_gene != 'None')
v_gene_or_j_gene <- count(v_gene_or_j_gene)$n

stats <- data.frame(v_gene_and_j_gene, v_gene_or_j_gene, no_v_gene_and_no_j_gene)
stats
}
```

```r
#How many IGK,IGH, IGL and Multi we have in the data? (cell distribution based on cha
in type)
Summary <- function(filtered_data)
{
  chain_summary <- group_by(filtered_data, chain)
  chain_summary <- summarize(chain_summary, count=n())
  chain_summary
}
```

```r
#Number of copies of each cell (we want to know how many copy of each cell we have)

barcode_summary <- function(filtered_data)
  {
  barcode_summary <- group_by(filtered_data, barcode)
  barcode_summary <- summarize(barcode_summary, count=n())
  barcode_summary
}
```

```
#B cells distribution by number of chains in a cell

barcode_summary <- function(filtered_data)
  {
  barcode_summary <- group_by(filtered_data, barcode)
  barcode_summary <- summarize(barcode_summary, count=n())
  barcode_summary <- group_by(barcode_summary, count)
  barcode_summary <- summarize(barcode_summary, count_total=n())
  barcode_summary
}
copy_barcode_filtered <- barcode_summary(filtered_data)
```

```
#B cells distribution by number of chains in a cell(percentage)
copy_barcode_filtered$count_total_pct = round((copy_barcode_filtered$count_total / su
m(copy_barcode_filtered$count_total)) * 100,2)
```

```
#we want to know each cell contain how many chain
occurance_of_each_chain <- function(filtered_data)
  {
  result <- filtered_data %>%
    group_by(barcode, chain) %>%
    filter(chain %in% c('IGK','IGH','IGL')) %>%
    summarize(count=n())
  result <- unite(result, 'result_chain', count, chain, remove=F, sep='')
  result <- summarize(result, type=paste(result_chain, collapse='_'), count=sum(coun
t))
  result
}
```

```
# occurance of each condition (we can add which condition that we want)
#the purpose of this calculation is that we want to know how many cell with one heavy
chain and one light chain there are(if there are too many, so it is good news) and al
so how many cells with 2 light chain we can find in the data! with this calculation w
e can study dual light chain effect

Condition <- function (result)
  {

  many_conditions = c(
  '1IGL', '1IGH', '1IGK', '1IGH_1IGK_1IGL' , '1IGH_1IGL' , '1IGH_1IGK', '1IGH_2IGK',
'IGH_2IGL')
  result %>%
  filter(type %in% many_conditions) %>%
  group_by(type) %>%
  summarise(total=n())
}
filtered_condition <- Condition(occurance_of_each_chain(filtered_data))
```

```
#beside above condition that we find,there are cells with other condition too.
get_total_without_condition <- function(result) {
  result %>%
    group_by(type) %>%
    summarise(total=n()) %>%
    pull(total) %>%
    sum
}
total_without_condition <- get_total_without_condition(occurance_of_each_chain(filter
ed_data))

other_data <- data.frame('Other', total_without_condition - sum(filtered_condition$to
tal))
names(other_data) <- names(filtered_condition)
filtered_condition <- rbind(filtered_condition, other_data)
```

```
#calculation with percentage
filtered_condition$total_pct = round((filtered_condition$total / sum(filtered_conditi
on$total)) * 100, 2)
```

```
#################################
#processing migmap output
#################################
```

```
#load the dataset
data_migmap <-read.csv ('/home/sedreh/ITMO/semester2/Bcellsproject/final_Rcode/PBMCs_
of_a_healthy_donor/migmap_result_healthy_donor.csv', sep="\t", header=TRUE)
```

```
# we need to have same columns names the same as filtered data for analysis(for examp
le:In #filtered data we have barcode as a first column name but in migmap the name is
read.heaer).So we need to split read and header to create barcode column.
#for this purpose, we need to do some steps:
preprocess_data <- function(data_migmap){
separate(data_migmap,
         col = "read.header",
         into = c("read", "header"),
         sep = "_")
}
data <- preprocess_data(data_migmap)
```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 2644 rows [1, 2,
## 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

```
#from the migmap output we will choose the column that we will work with them and cha
nge the columns' name (same as filtered data)
preprocess <- function(data) {
data <- data %>% select(barcode = read, v_gene = v.segment, d_gene = d.segment, j_gen
e= j.segment)
data
}
data <- preprocess(data)
```

```
#Number of cells (As we have several copies of each barcode,we need to count just one
copy of unique barcoe and calculate number of cells)

barcode_summary <- function(data)
{
  number_of_cells <- data %>%
    distinct(barcode) %>%
    dplyr::count()

number_of_cells$n
}
```

```
#look at to the quality of chains

count_table <- function(data)
{
v_gene_and_j_gene <- filter(data, v_gene != 'None' & j_gene != 'None')
v_gene_and_j_gene <- count(v_gene_and_j_gene)$n

no_v_gene_and_no_j_gene <- filter(data, v_gene == 'None' & j_gene == 'None')
no_v_gene_and_no_j_gene <- count(no_v_gene_and_no_j_gene)$n

v_gene_or_j_gene <- filter(data, v_gene != 'None' | j_gene != 'None')
v_gene_or_j_gene <- count(v_gene_or_j_gene)$n

stats <- data.frame(v_gene_and_j_gene, v_gene_or_j_gene, no_v_gene_and_no_j_gene)
stats
}
```

```
# In midmap data we do not have chain column that shows the type of chain in each cel
l. Therefore we need to creat chain column based on v_gene, d_gene and j_gene (the sa
me as filtered data)

matrix_gene_data <- as.matrix(data[,2:4])
 matrix_gene_data <- substr(matrix_gene_data, 1, 3) # get only first three characters
 data$chain <- apply(matrix_gene_data, 1, function(x) {
   x <- x[!(x %in% ".")] # removeing .
   x <- unique(x) # get unique value from row
   if(length(x) == 0) { # if all are . then return none
     "None"
   } else if (length(x) > 1) { # if more than 1 unique value then it's multi
     "Multi"
   } else { # otherwise just single chain value
     x
   }
 })
```

```r
#Number of copies of each cell

barcode_summary <- function(data)
  {
  barcode_summary <- group_by(data, barcode)
  barcode_summary <- summarize(barcode_summary, count=n())
  barcode_summary
}
```

```r
#B cells distribution by number of chains in a cell
barcode_summary <- function(data)
  {
  barcode_summary <- group_by(data, barcode)
  barcode_summary <- summarize(barcode_summary, count=n())
  barcode_summary <- group_by(barcode_summary, count)
  barcode_summary <- summarize(barcode_summary, count_total=n())
  barcode_summary
}
copy_barcode_migmap <- barcode_summary(data)
```

```r
#B cells distribution by number of chains in a cell(percentage)
copy_barcode_migmap$count_total_pct = round((copy_barcode_migmap$count_total / sum(co
py_barcode_migmap$count_total)) * 100, 2)
```

```r
#we want to know each cell how many chain has

occurance_of_each_chain <- function(data)
  {
  result <- data %>%
    group_by(barcode, chain) %>%
    filter(chain %in% c('IGK','IGH','IGL')) %>%
    summarize(count=n())
  result <- unite(result, 'result_chain', count, chain, remove=F, sep='')
  result <- summarize(result, type=paste(result_chain, collapse='_'), count=sum(coun
t))
  result
}
```

```r
# I added whichever condition that I want from data

Condition <- function (result)
  {

  many_conditions = c(
  '1IGL', '1IGH', '1IGK', '1IGH_1IGK_1IGL' , '1IGH_1IGL' , '1IGH_1IGK', '1IGH_2IGK',
'IGH_2IGL')
  result %>%
  filter(type %in% many_conditions) %>%
  group_by(type) %>%
  summarise(total=n())
}
migmap_condition <- Condition(occurance_of_each_chain(data))
```

```
#beside above condition that we find,there are cells with other condition too.
get_total_without_condition <- function(result) {
  result %>%
    group_by(type) %>%
    summarise(total=n()) %>%
    pull(total) %>%
    sum
}
total_without_condition <- get_total_without_condition(occurance_of_each_chain(data))

other_data <- data.frame('Other', total_without_condition - sum(migmap_condition$tota
l))
names(other_data) <- names(migmap_condition)
migmap_condition <- rbind(migmap_condition, other_data)
```

```
#calculation with percentage
migmap_condition$total_pct = round((migmap_condition$total / total_without_condition)
* 100, 2)
```

```
########################################
      #calculating chain distance
########################################
#for this purpos we need to have read.header in migmapdata instead of barcode because
it contains barcode and "contig"! here we need contig for recognizing the type of cha
in in "fasta file"! so we should not delete it like previous studies!
```

```
#load migmap output file
migmap <-read.csv ('/home/sedreh/ITMO/semester2/Bcellsproject/final_Rcode/PBMCs_of_a_
healthy_donor/migmap_result_healthy_donor.csv', sep="\t", header=TRUE)
#load fasta file for recognizing light chains in cells
d <- read.fasta('/home/sedreh/ITMO/semester2/Bcellsproject/final_Rcode/PBMCs_of_a_hea
lthy_donor/vdj_v1_hs_pbmc_b_filtered_contig.fasta')
```

```
#select needed columns from data and rename them like filtered data
preprocess_data <- function(migmap) {
migmap <- migmap %>% select(barcode = read.header, v_gene = v.segment, d_gene = d.seg
ment, j_gene= j.segment)
migmap
}
migmap <- preprocess_data(migmap)
```

```
#determine the chain type for each cell based on v, d and j gene and make "chain" col
umn
matrix_gene_data <- as.matrix(migmap[,2:4])
 matrix_gene_data <- substr(matrix_gene_data, 1, 3) # get only first three characters
 migmap$chain <- apply(matrix_gene_data, 1, function(x) {
   x <- x[!(x %in% ".")] # removeing .
   x <- unique(x) # get unique value from row
   if(length(x) == 0) { # if all are . then return none
     "None"
   } else if (length(x) > 1) { # if more than 1 unique value then it's multi
     "Multi"
   } else { # otherwise just single chain value
     x
   }
 })
```

```
occurance_of_each_chain <- function(migmap)
  {
  result <- migmap %>%
    group_by(barcode, chain) %>%
    filter(chain %in% c('IGK','IGH','IGL')) %>%
    summarize(count=n())
  result <- unite(result, 'result_chain', count, chain, remove=F, sep='')
  result <- summarize(result, type=paste(result_chain, collapse='_'), count=sum(coun
t))
  result
}
```

```
#In this step we need to recognize the type of each chain with contig to search the s
equence in fasta file
estimate_condition <- function(migmap) {
  migmap %>%
  separate(barcode, into=c('bc','contig'),sep = '1_',remove = F) %>%
    mutate(bc=substr(bc,2,18)) %>%
  filter(chain %in% c('IGK','IGH','IGL')) %>%
     group_by(bc, chain) %>%
  summarize(contig=paste(contig, collapse = ','), count=n()) %>%
  unite('result_chain', count, chain, remove=F, sep='') %>%
  unite('contig', result_chain, contig, sep = '-', remove = F) %>%
  group_by(bc) %>%
  summarize(type=paste(result_chain, collapse='_'), count=sum(count), contig=paste(co
ntig, collapse = '@'))
}
```

```
#calculate score of distance between 2 light chain in cells with dual light chain con
dition
get_score <- function(s, condition) {
  mat <- nucleotideSubstitutionMatrix(match = 0, mismatch = 1, baseOnly = TRUE)
scores <- estimate_condition(s) %>%
  filter(type %in%  c(condition)) %>%
  apply(1, function(x) {
    contig <- strsplit(x['contig'], '@')[[1]]

    if(length(contig) == 2) {
      contig <- strsplit(contig[2], '-')[[1]]
      contig <- strsplit(contig[2], ',')[[1]]
      barcodes <- c(
        paste(x['bc'], contig[1], sep = '1_'),
        paste(x['bc'], contig[2], sep = '1_'))
    } else {
      contig1 <- strsplit(contig[2], '-')[[1]]
      contig1 <- strsplit(contig1[2], ',')[[1]]

      contig2 <- strsplit(contig[3], '-')[[1]]
      contig2 <- strsplit(contig2[2], ',')[[1]]

       barcodes <- c(
        paste(x['bc'], contig1[1], sep = '1_'),
        paste(x['bc'], contig2[1], sep = '1_'))
    }

    s1 <- DNAString(
      toupper(
        paste(d[barcodes[1]][[1]], collapse = '')
      )
    )

    s2 <- DNAString(
      toupper(
        paste(d[barcodes[2]][[1]], collapse = '')
      )
    )

    globalAlign <-
      pairwiseAlignment(s1, s2, substitutionMatrix = mat,
                        gapOpening = 0, gapExtension = -1, scoreOnly=T)
    globalAlign
  }) %>% unlist
}
scores_1IGH_2IGL <- -get_score(migmap, '1IGH_2IGL')
scores_1IGH_2IGL
```

```
##  [1] 149 158 180 136 156 113 102 199 139 162 175 185 192  22 174 181 172
## [18] 226 110 202 194
```

```
scores_1IGH_1IGK_1IGL <- -get_score(migmap, '1IGH_1IGK_1IGL')
scores_1IGH_1IGK_1IGL
```

```
##  [1] 316 310 305 408 316 338 435 316 366 337 350 352 303 294 319 293 333
## [18] 298 285 335 309 350 282
```

```
scores_1IGH_2IGK <- -get_score(migmap, '1IGH_2IGK')
scores_1IGH_2IGK
```

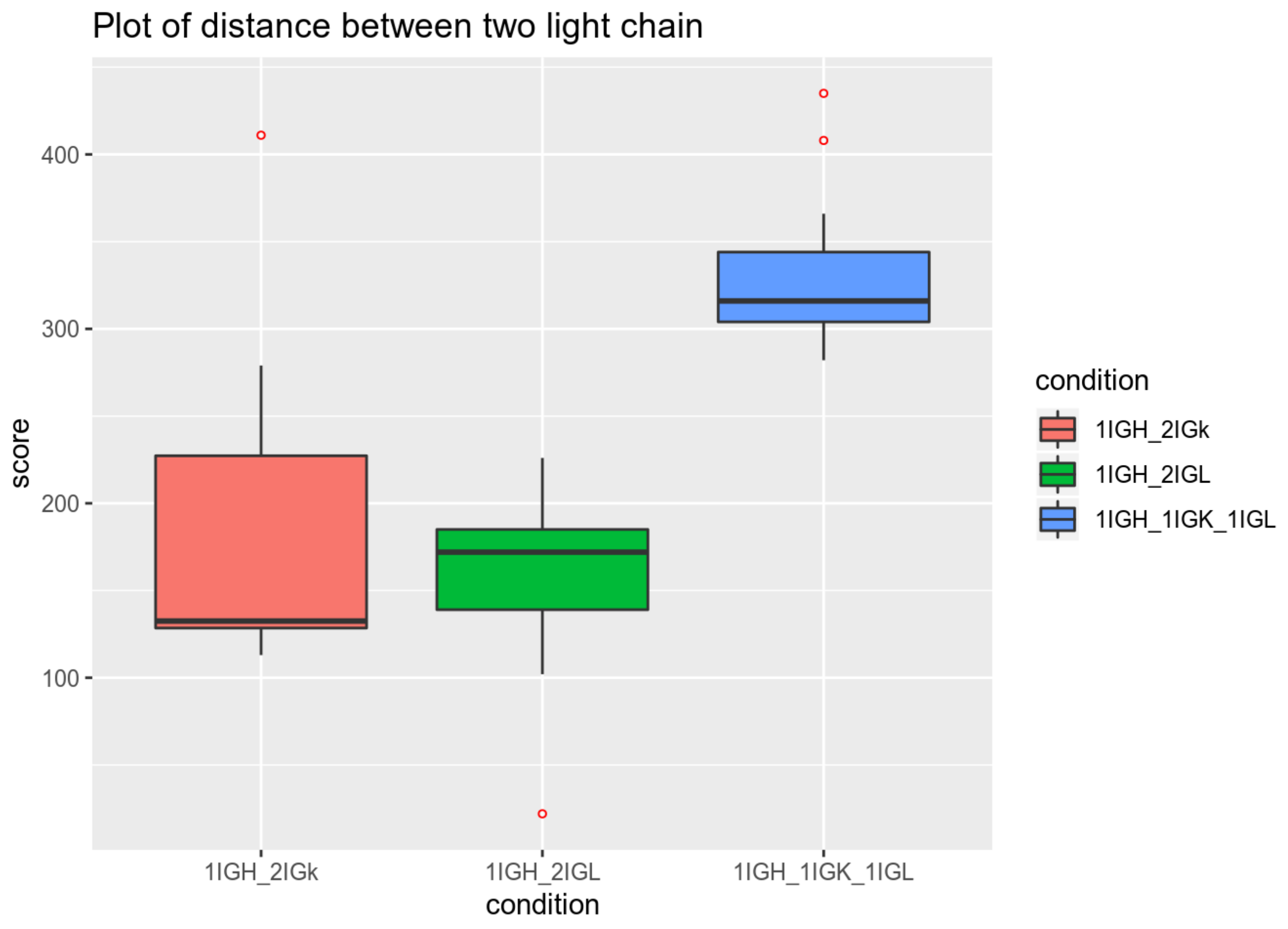
```
##  [1] 130 130 411 232 113 279 128 213 127 135
```

```
#calculatr light chains' lenght
get_chain_length <- function(s, condition) {
lengths <- estimate_condition(s) %>%
  filter(type %in%  c(condition)) %>%
  apply(1, function(x) {
    contig <- strsplit(x['contig'], '@')[[1]]

    if(length(contig) == 2) {
      contig <- strsplit(contig[2], '-')[[1]]
      contig <- strsplit(contig[2], ',')[[1]]
      barcodes <- c(
        paste(x['bc'], contig[1], sep = '1_'),
        paste(x['bc'], contig[2], sep = '1_'))
    } else {
      contig1 <- strsplit(contig[2], '-')[[1]]
      contig1 <- strsplit(contig1[2], ',')[[1]]

      contig2 <- strsplit(contig[3], '-')[[1]]
      contig2 <- strsplit(contig2[2], ',')[[1]]

       barcodes <- c(
        paste(x['bc'], contig1[1], sep = '1_'),
        paste(x['bc'], contig2[1], sep = '1_'))
    }

    lengths <- list()
    lengths[[barcodes[1]]] <- length(d[barcodes[1]][[1]])
    lengths[[barcodes[2]]] <- length(d[barcodes[2]][[1]])

    lengths
  })
  lengths %>% unlist
}
get_chain_length(migmap, '1IGH_2IGL')
```

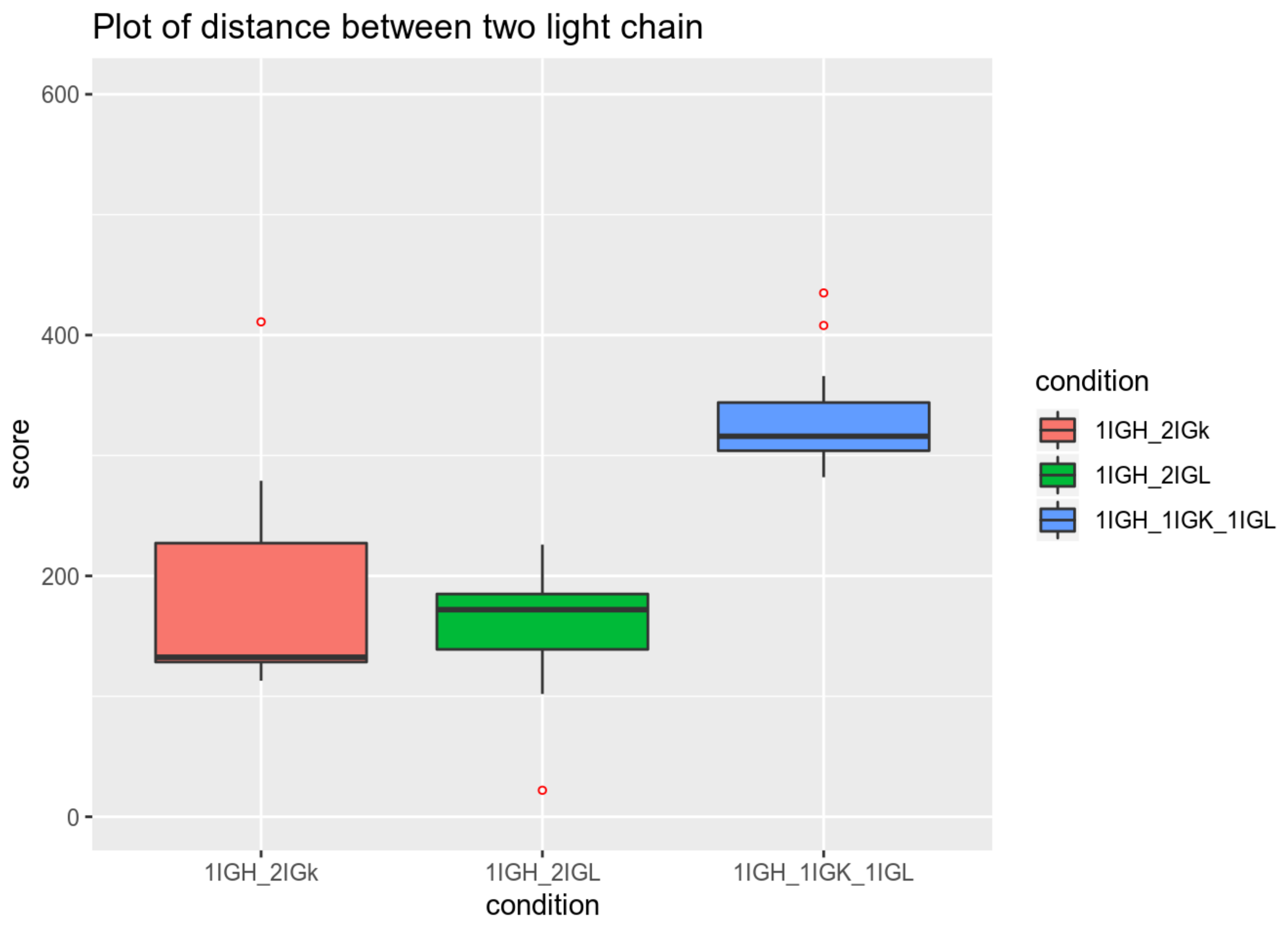
```
## ACCGTAAGTATCAGTC-1_contig_1 ACCGTAAGTATCAGTC-1_contig_3 
##                         647                         668 
## AGACGTTCATCTACGA-1_contig_1 AGACGTTCATCTACGA-1_contig_2 
##                         678                         709 
## CAAGAAAAGCTCCTCT-1_contig_1 CAAGAAAAGCTCCTCT-1_contig_2 
##                         664                         674 
## CAGCGACAGTACGTTC-1_contig_1 CAGCGACAGTACGTTC-1_contig_5 
##                         662                         732 
## CGATCGGGTTTGGGCC-1_contig_1 CGATCGGGTTTGGGCC-1_contig_3 
##                         683                         692 
## CGCTATCCACGACGAA-1_contig_1 CGCTATCCACGACGAA-1_contig_2 
##                         663                         635 
## CTACATTCACGTGAGA-1_contig_2 CTACATTCACGTGAGA-1_contig_3 
##                         678                         695 
## CTCGTACGTGAGTGAC-1_contig_1 CTCGTACGTGAGTGAC-1_contig_3 
##                         675                         689 
## GACAGAGTCCAGAAGG-1_contig_2 GACAGAGTCCAGAAGG-1_contig_3 
##                         705                         661 
## GACCAATTCCTTGCCA-1_contig_2 GACCAATTCCTTGCCA-1_contig_3 
##                         642                         673 
## GAGTCCGGTACTCGCG-1_contig_4 GAGTCCGGTACTCGCG-1_contig_5 
##                         688                         658 
## GCATGATTCGGAAACG-1_contig_4 GCATGATTCGGAAACG-1_contig_5 
##                         662                         726 
## GCGAGAACAGGAATCG-1_contig_1 GCGAGAACAGGAATCG-1_contig_2 
##                         704                         657 
## GCGCGATGTAAGTGTA-1_contig_1 GCGCGATGTAAGTGTA-1_contig_3 
##                         650                         665 
## GGGACCTAGCGTAATA-1_contig_1 GGGACCTAGCGTAATA-1_contig_2 
##                         683                         690 
## TACCTTATCAGCAACT-1_contig_1 TACCTTATCAGCAACT-1_contig_2 
##                         672                         654 
## TCATTACGTCTCTTTA-1_contig_1 TCATTACGTCTCTTTA-1_contig_3 
##                         653                         666 
## TCGTACCGTGTCCTCT-1_contig_2 TCGTACCGTGTCCTCT-1_contig_3 
##                         740                         674 
## TGACTTTTCTACTCAT-1_contig_1 TGACTTTTCTACTCAT-1_contig_2 
##                         653                         676 
## TGAGAGGAGCTGTTCA-1_contig_1 TGAGAGGAGCTGTTCA-1_contig_3 
##                         779                         790 
## TGGCCAGGTTATCACG-1_contig_1 TGGCCAGGTTATCACG-1_contig_2 
##                         759                         682
```

```
#Making dataframe from all conditions(with two light chain) with distance score

IGH_2IGk.data <- data.frame("1IGH_2IGk",scores_1IGH_2IGK) %>% select(condition = "X.1
IGH_2IGk.", score = "scores_1IGH_2IGK")
#names(IGH_2IGK.data)

IGH_2IGL.data <- data.frame('1IGH_2IGL', scores_1IGH_2IGL)  %>%
  data.frame("1IGH_2IGL",scores_1IGH_2IGL) %>% select(condition = "X.1IGH_2IGL.", sco
re = "scores_1IGH_2IGL")
#names(IGH_2IGL.data)

IGH_IGL_IGK.data <- data.frame("1IGH_1IGK_1IGL",scores_1IGH_1IGK_1IGL) %>%
  select(condition = "X.1IGH_1IGK_1IGL.", score = "scores_1IGH_1IGK_1IGL")
#names(IGH_IGL_IGK.data)

migmap <- rbind(IGH_2IGk.data, IGH_2IGL.data, IGH_IGL_IGK.data)
migmap$condition <- as.factor(migmap$condition)
```

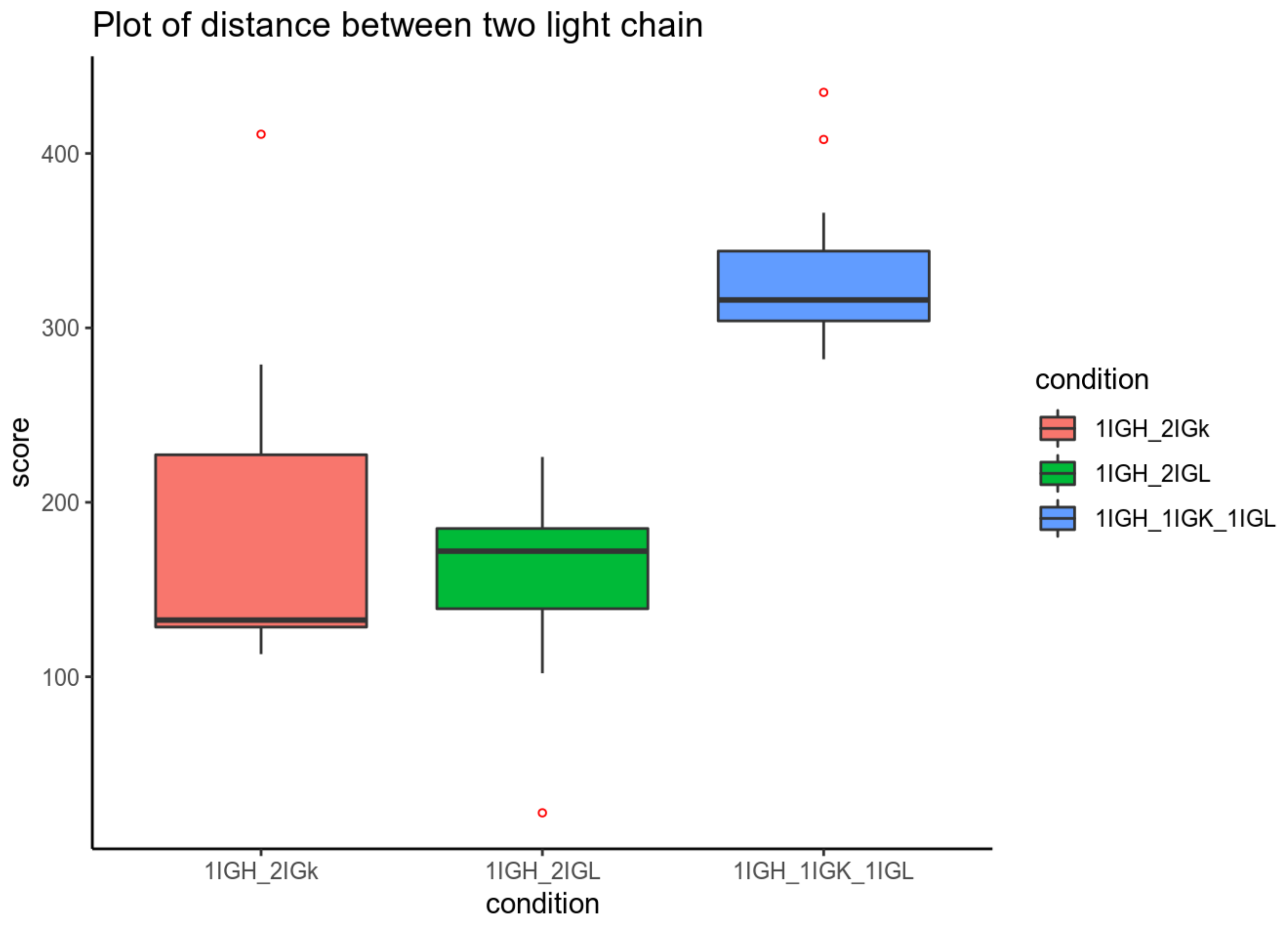
```
#plot distance between 2 light chain in cells with dual light chain condition
p <- ggplot(migmap, aes(x=condition, y=score, fill=condition)) +
  geom_boxplot(outlier.colour="red", outlier.shape=1,
              outlier.size=1)+
  labs(title="Plot of distance between two light chain",x="condition", y = "score")
p+scale_color_brewer(palette="Dark2")
```



```
p+coord_cartesian(ylim = c(2, 600))
```



```
p + theme_classic()
```


Plot of distance between two light chain
score
1IGH_2IGk
1IGH_2IGL
1IGH_1IGK_1IGL
condition
condition
1IGH_2IGk
1IGH_2IGL
1IGH_1IGK_1IGL