Agrupamento de clonótipos para TCR e BCR

Contexto

A etapa de agrupamento de sequências de BCRs e TCRs é comum na análise de repertório desses receptores. Em células B, esta etapa é necessária afim de produzir linhagens clonais. Permitindo o avanço no entendimento dos níveis de hipermutação somática que alteram a afinidade de ligação.

Quanto aos TCR, já foi demonstrado que células T capazes de reconhecer epítopos idênticos têm TCR altemente semelhante. Assim, o agrupamento de TCR semelhantes, permite encontrar repertórios semelhantes que provavelmente reconhecem os mesmos antígenos.

O agrupamento envolve duas etapas:

1. cálculo da distância entre as sequências
2. agrupar as sequências usando as informações das distâncias

Um clonótipo é um conjunto de células que supostamente compartilham um ancestral comum ou propriedades comuns. Em repertórios de TCR e BCR, existem algumas maneiras em definir clonótipos.

Cálculo da distância e agrupamento

```
# -- pacotes necessarios
library(immunarch)
library(dplyr)
```

Inciei preparando o ambiente de trabalho, importando e tratando os dados necessários.

```
# -- preparando o ambiente de trabalho
dir_data <-
  "../../../../Projetos/Bigdata/BigData/BigData/repertorio_tcrbcr_acc/data/"
```

```
immdata <- repLoad(paste(dir_data, "outputTrust4_report", sep = ""))
load(paste(dir_data, "metadata.RData", sep = ""))

immdata$meta <- metadata[metadata$sample_id %in%
                           gsub("_report", "", immdata$meta$Sample),]
```

Separei os repertórios de TCR e BCR, pois a definição de clonótipos para estes receptores é diferente.

```
# -- separacao dos repertorios de TCR e BCR
immdata$meta$Sample <- paste(immdata$meta$sample_id,"_report",sep = "")

immdata$meta$Sample %in% names(immdata$data)

immdata_bcr <- repFilter(immdata,
                         .method = "by.clonotype",
                         .query = list(V.name = include("IG")),
                         .match = "substring")

immdata_tcr <- repFilter(immdata,
                         .method = "by.clonotype",
                         .query = list(V.name = include("TR")),
                         .match = "substring")
```

```
# -- funcao para remover linhas com NA na coluna J.name
remove_NA <- function(df){
  df %>%
    filter(!is.na(J.name))
}

# -- funcao para separar os NA e acrescentar a tabela final depois
capturar_NA <- function(df){
  df %>%
    filter(is.na(J.name))
}

# -- NAs em J.name para acrescentar
TCR_NAs_acrescentar <- lapply(immdata_tcr$data, capturar_NA)
BCR_NAs_acrescentar <- lapply(immdata_bcr$data, capturar_NA)

# -- removendo linhas com valores NA na coluna J.name
```

```
immdata_tcr$data <- lapply(immdata_tcr$data, remove_NA)
immdata_bcr$data <- lapply(immdata_bcr$data, remove_NA)
```

Para os clonótipos de BCR estou considerando sequências com o mesmo tamanho da região CDR3 (em pb), mesmo gene V e J e 90% de similaridade entre as sequências de nucleotídeos para a região CDR3. Paro os clonótipos de TCR consirei o mesmo tamanho da região CDR3 (em pb), mesmo gene V e J e 95% de similaridade entre as sequências de nucleotídeos para a região CDR3.

```
# -- calculo da distancia entre as sequencias
dist_BCR <- seqDist(immdata_bcr$data,
                    .group_by = c("V.name", "J.name"),
                    .group_by_seqLength = TRUE,
                    .col = "CDR3.nt")

clust_BCR <- seqCluster(immdata_bcr$data, dist_BCR, .perc_similarity = 0.9)


dist_TCR <- seqDist(immdata_tcr$data[-12],
                    .group_by = c("V.name", "J.name"),
                    .group_by_seqLength = TRUE,
                    .col = "CDR3.nt")

clust_TCR <- seqCluster(immdata_tcr$data[-12], dist_TCR, .perc_similarity = 0.95)
```

Precisei remover a amostra "12" porque o cálculo da distância e geração dos clauster geravam um erro (fiz a clusterização dessa amostra na "mão").

```
temp <- immdata_tcr$data[12]

temp$`130723_UNC9-SN296_0386_BC2E4WACXX_TAGCTT_L006_report`$J.name
```

```
[1] "TRAJ17*01" "TRAJ17*01"
```

```
nchar(temp$`130723_UNC9-SN296_0386_BC2E4WACXX_TAGCTT_L006_report`$CDR3.nt)
```

```
[1] 39 39
```

```
temp$`130723_UNC9-SN296_0386_BC2E4WACXX_TAGCTT_L006_report`$Cluster <-
  c("TRAV9-2/TRAJ17_length_39","TRAV9-2/TRAJ17_length_39")

names(temp)
```

```
[1] "130723_UNC9-SN296_0386_BC2E4WACXX_TAGCTT_L006_report"
```

```
clust_TCR$`130723_UNC9-SN296_0386_BC2E4WACXX_TAGCTT_L006_report` <-
  as.data.frame(temp$`130723_UNC9-SN296_0386_BC2E4WACXX_TAGCTT_L006_report`)


clust_TCR$`130723_UNC9-SN296_0386_BC2E4WACXX_TAGCTT_L006_report`
```

```
  Clones Proportion                                    CDR3.nt       CDR3.aa
1      1  0.1428571 TGTGCTCTGAGGAAAGCTGCAGGCAACAAGCTAATTTTT CALRKAAGNKLIF
2      1  0.1428571 TGTGCTCTGAGGAAAGCTGCAGGCAACAAGCTAACTTTT CALRKAAGNKLTF
     V.name D.name    J.name V.end D.start D.end J.start VJ.ins VD.ins DJ.ins
1 TRAV9-2*01   <NA> TRAJ17*01    NA      NA    NA      NA     NA     NA     NA
2 TRAV9-2*01   <NA> TRAJ17*01    NA      NA    NA      NA     NA     NA     NA
                  Cluster
1 TRAV9-2/TRAJ17_length_39
2 TRAV9-2/TRAJ17_length_39
```

```
clust_TCR$`130723_UNC9-SN296_0386_BC2E4WACXX_ACTTGA_L003_report`
```

```
  Clones Proportion                                        CDR3.nt
1      3 0.10000000 TGTGCCAGCAGCGTAGACCGGACAGGAGGGGACTATGGCTACACCTTC
2      1 0.03333333          TGTGCCAGCAGTTACTCCGGTACGGACGAGCAGTACTTC
3      1 0.03333333       TGCAGCGTCTTACTAGCGGGGAGCACAGATACGCAGTATTTT
          CDR3.aa      V.name   D.name     J.name V.end D.start D.end J.start
1 CASSVDRTGGDYGYTF    TRBV9*01 TRBD1*01 TRBJ1-2*01    NA      NA    NA      NA
2    CASSYSGTDEQYF  TRBV6-5*01 TRBD2*01 TRBJ2-7*01    NA      NA    NA      NA
3   CSVLLAGSTDTQYF TRBV29-1*01 TRBD2*01 TRBJ2-3*01    NA      NA    NA      NA
  VJ.ins VD.ins DJ.ins                   Cluster
1     NA     NA     NA    TRBV9/TRBJ1-2_length_48
2     NA     NA     NA  TRBV6-5/TRBJ2-7_length_39
3     NA     NA     NA TRBV29-1/TRBJ2-3_length_42
```

Após gerar os clusteres para as sequências de TCR e BCR, precisei agrupá-las com base nos clusteres, ou seja, as sequências que foram condizentes com os critérios de clonótipos foram agrupadas e assim, tratadas como clonótipos.

```
# funcao para capturar as linhas de cada dataframe que contem NAs em Cluster
capturar_NA_cluster <- function(df){
  df %>%
    filter(is.na(Cluster))
}

# -- NAs em cluster para acrescentar
TCR_NAs_acrescentar_cluster <- lapply(clust_TCR, capturar_NA_cluster)
BCR_NAs_acrescentar_cluster <- lapply(clust_BCR, capturar_NA_cluster)

# -- funcao para remover linhas com NA na coluna Cluster
remove_NA_cluster <- function(df){
  df %>%
    filter(!is.na(Cluster))
}

# removendo NAs para acrescentar depois
clust_TCR <- lapply(clust_TCR, remove_NA_cluster)
clust_BCR <- lapply(clust_BCR, remove_NA_cluster)
```

```
# funcao para agrupar os dataframes
agrupar_por_cluster <- function(df){
  df %>%
    group_by(Cluster) %>%
    mutate(Clones = sum(Clones),
           Proportion = sum(Proportion)) %>%
    ungroup() %>%
    distinct(Cluster, .keep_all = TRUE) %>%
    as.data.frame()
}

# aplicando a funcao na lista de dataframes
clust_TCR_agrupados <- lapply(clust_TCR, agrupar_por_cluster)
clust_BCR_agrupados <- lapply(clust_BCR, agrupar_por_cluster)
```

```r
# adicionando coluna cluster
add_col <- function(lista){
  for (i in seq_along(lista)) {
    lista[[i]]$Cluster <- NA
  }
  return(lista)
}

BCR_NAs_acrescentar <- add_col(BCR_NAs_acrescentar)
TCR_NAs_acrescentar <- add_col(TCR_NAs_acrescentar)
```

```r
juntar_df <- function(df1, df2, df3){
  return(rbind(df1,df2,df3))
}

clust_TCR_agrupados <- Map(juntar_df, clust_TCR_agrupados,
                           TCR_NAs_acrescentar, TCR_NAs_acrescentar_cluster)

clust_BCR_agrupados <- Map(juntar_df, clust_BCR_agrupados,
                           BCR_NAs_acrescentar, BCR_NAs_acrescentar_cluster)
```

Formatação

Afim de gerar tabelas com a estrutura das tabelas originais (report) agrupei os dataframe de TCR com os dataframes de BCR.

```r
# funcao para juntar listas
juntar_listas <- function(lista1, lista2){
  nomes_comuns <- intersect(names(lista1), names(lista2))
  lista_final <- list()

  for (nome in nomes_comuns) {
    df1 <- lista1[[nome]]
    df2 <- lista2[[nome]]

    lista_final[[nome]] <- rbind(df1,df2)
  }

  nomes_lista1 <- setdiff(names(lista1), nomes_comuns)
```

```
  for (nome in nomes_lista1) {
    lista_final[[nome]] <- lista1[[nome]]
  }

  nomes_lista2 <- setdiff(names(lista2), nomes_comuns)
  for (nome in nomes_lista2) {
    lista_final[[nome]] <- lista2[[nome]]
  }

  return(lista_final)
}

lista_tcr_bcr <- juntar_listas(clust_TCR_agrupados, clust_BCR_agrupados)
```

Aqui está o exemplo de dataframe, após essa primeira formatação:

```
lista_tcr_bcr$`130723_UNC9-SN296_0386_BC2E4WACXX_ACTTGA_L003_report`
```

```
   Clones Proportion                                          CDR3.nt
1       3 0.10000000 TGTGCCAGCAGCGTAGACCGGACAGGAGGGGACTATGGCTACACCTTC
2       1 0.03333333          TGTGCCAGCAGTTACTCCGGTACGGACGAGCAGTACTTC
3       1 0.03333333       TGCAGCGTCTTACTAGCGGGGAGCACAGATACGCAGTATTTT
4       7 0.23333333          TGCTGCTCATATGCAGGTAGTACCACTTTCGCGGTATTT
5       3 0.10000000                TGTCAACAGGCTTACAGTCCCCCTGAGACGTTC
6       3 0.10000000                TGTCAGCAGTATGGTACCTCACCTGAAATGTTC
7       2 0.06666667                TGTCAAAAGTATGACAGTGTCCCGCTCACTTTC
8       2 0.06666667                   TGCATGCAAACTCTACAAAGGGAGACGTTC
9       2 0.06666667                TGTCAACAGGCTCACAGTTTCCCCTTCACTTTC
10      1 0.03333333             TGCGGCTCATATACAAGCAGCAGCACTCGGGTGTTC
11      1 0.03333333                TGTCAACAATTTAATAATTTCCCGCTCACTTTC
12      1 0.03333333                TGCTTGCTCTCCTATAATGGTCCTTGGGTGTTC
13      1 0.03333333                TGTCAGCAGTATAAAAACTGGCCTCCCACTTTC
14      1 0.03333333                TGTCAGCAATCTTATAGTGCTCCGATCACCTTC
15      1 0.03333333                TGCATGCAAGCTACACAGTGGCCGTACACTTTT
           CDR3.aa      V.name   D.name     J.name V.end D.start D.end J.start
1  CASSVDRTGGDYGYTF    TRBV9*01 TRBD1*01 TRBJ1-2*01    NA      NA    NA      NA
2     CASSYSGTDEQYF  TRBV6-5*01 TRBD2*01 TRBJ2-7*01    NA      NA    NA      NA
3    CSVLLAGSTDTQYF TRBV29-1*01 TRBD2*01 TRBJ2-3*01    NA      NA    NA      NA
4     CCSYAGSTTFAVF IGLV2-23*02     <NA>   IGLJ2*01    NA      NA    NA      NA
5       CQQAYSPPETF IGKV1-12*01     <NA>   IGKJ1*01    NA      NA    NA      NA
6       CQQYGTSPEMF IGKV3-20*01     <NA>   IGKJ1*01    NA      NA    NA      NA
```

```
7       CQKYDSVPLTF IGKV1-37*01     <NA>   IGKJ4*01    NA      NA    NA      NA
8        CMQTLQRETF IGKV2-28*01     <NA>   IGKJ1*01    NA      NA    NA      NA
9       CQQAHSFPFTF IGKV1-12*01     <NA>   IGKJ3*01    NA      NA    NA      NA
10     CGSYTSSSTRVF IGLV2-14*01     <NA>   IGLJ3*02    NA      NA    NA      NA
11      CQQFNNFPLTF IGKV1-13*01     <NA>   IGKJ4*01    NA      NA    NA      NA
12      CLLSYNGPWVF IGLV7-46*01     <NA>   IGLJ3*02    NA      NA    NA      NA
13      CQQYKNWPPTF IGKV3-15*01     <NA>   IGKJ4*01    NA      NA    NA      NA
14      CQQSYSAPITF  IGKV4-1*01     <NA>   IGKJ5*01    NA      NA    NA      NA
15      CMQATQWPYTF IGKV2-18*01     <NA>   IGKJ2*01    NA      NA    NA      NA
   VJ.ins VD.ins DJ.ins                      Cluster
1      NA     NA     NA    TRBV9/TRBJ1-2_length_48
2      NA     NA     NA  TRBV6-5/TRBJ2-7_length_39
3      NA     NA     NA TRBV29-1/TRBJ2-3_length_42
4      NA     NA     NA   IGLV2-23/IGLJ2_length_39
5      NA     NA     NA   IGKV1-12/IGKJ1_length_33
6      NA     NA     NA   IGKV3-20/IGKJ1_length_33
7      NA     NA     NA   IGKV1-37/IGKJ4_length_33
8      NA     NA     NA   IGKV2-28/IGKJ1_length_30
9      NA     NA     NA   IGKV1-12/IGKJ3_length_33
10     NA     NA     NA   IGLV2-14/IGLJ3_length_36
11     NA     NA     NA   IGKV1-13/IGKJ4_length_33
12     NA     NA     NA   IGLV7-46/IGLJ3_length_33
13     NA     NA     NA   IGKV3-15/IGKJ4_length_33
14     NA     NA     NA    IGKV4-1/IGKJ5_length_33
15     NA     NA     NA   IGKV2-18/IGKJ2_length_33
```

Substitui os NAs por ".", visto que, nas tabelas originais, os valores faltantes são representados por pontos (.).

```r
# funcao para substituir NA por "."
substituir_na <- function(df){
  df[is.na(df)] <- "."
  return(df)
}

for (i in seq_along(lista_tcr_bcr)) {
  lista_tcr_bcr[[i]] <- substituir_na(lista_tcr_bcr[[i]])
}
```

Aqui está o exemplo de como ficou após essa segunda formatação:

```r
lista_tcr_bcr$`130723_UNC9-SN296_0386_BC2E4WACXX_ACTTGA_L003_report`
```

	Clones	Proportion	CDR3.nt
1	3	0.10000000	TGTGCCAGCAGCGTAGACCGGACAGGAGGGGACTATGGCTACACCTTC
2	1	0.03333333	TGTGCCAGCAGTTACTCCGGTACGGACGAGCAGTACTTC
3	1	0.03333333	TGCAGCGTCTTACTAGCGGGGAGCACAGATACGCAGTATTTT
4	7	0.23333333	TGCTGCTCATATGCAGGTAGTACCACTTTCGCGGTATTT
5	3	0.10000000	TGTCAACAGGCTTACAGTCCCCCTGAGACGTTC
6	3	0.10000000	TGTCAGCAGTATGGTACCTCACCTGAAATGTTC
7	2	0.06666667	TGTCAAAAGTATGACAGTGTCCCGCTCACTTTC
8	2	0.06666667	TGCATGCAAACTCTACAAAGGGAGACGTTC
9	2	0.06666667	TGTCAACAGGCTCACAGTTTCCCCTTCACTTTC
10	1	0.03333333	TGCGGCTCATATACAAGCAGCAGCACTCGGGTGTTC
11	1	0.03333333	TGTCAACAATTTAATAATTTCCCGCTCACTTTC
12	1	0.03333333	TGCTTGCTCTCCTATAATGGTCCTTGGGTGTTC
13	1	0.03333333	TGTCAGCAGTATAAAAACTGGCCTCCCACTTTC
14	1	0.03333333	TGTCAGCAATCTTATAGTGCTCCGATCACCTTC
15	1	0.03333333	TGCATGCAAGCTACACAGTGGCCGTACACTTTT

	CDR3.aa	V.name	D.name	J.name	V.end	D.start	D.end	J.start
1	CASSVDRTGGDYGYTF	TRBV9*01	TRBD1*01	TRBJ1-2*01	.	.	.	.
2	CASSYSGTDEQYF	TRBV6-5*01	TRBD2*01	TRBJ2-7*01	.	.	.	.
3	CSVLLAGSTDTQYF	TRBV29-1*01	TRBD2*01	TRBJ2-3*01	.	.	.	.
4	CCSYAGSTTFAVF	IGLV2-23*02	.	IGLJ2*01	.	.	.	.
5	CQQAYSPPETF	IGKV1-12*01	.	IGKJ1*01	.	.	.	.
6	CQQYGTSPEMF	IGKV3-20*01	.	IGKJ1*01	.	.	.	.
7	CQKYDSVPLTF	IGKV1-37*01	.	IGKJ4*01	.	.	.	.
8	CMQTLQRETF	IGKV2-28*01	.	IGKJ1*01	.	.	.	.
9	CQQAHSFPFTF	IGKV1-12*01	.	IGKJ3*01	.	.	.	.
10	CGSYTSSSTRVF	IGLV2-14*01	.	IGLJ3*02	.	.	.	.
11	CQQFNNFPLTF	IGKV1-13*01	.	IGKJ4*01	.	.	.	.
12	CLLSYNGPWVF	IGLV7-46*01	.	IGLJ3*02	.	.	.	.
13	CQQYKNWPPTF	IGKV3-15*01	.	IGKJ4*01	.	.	.	.
14	CQQSYSAPITF	IGKV4-1*01	.	IGKJ5*01	.	.	.	.
15	CMQATQWPYTF	IGKV2-18*01	.	IGKJ2*01	.	.	.	.

	VJ.ins	VD.ins	DJ.ins	Cluster
1	.	.	.	TRBV9/TRBJ1-2_length_48
2	.	.	.	TRBV6-5/TRBJ2-7_length_39
3	.	.	.	TRBV29-1/TRBJ2-3_length_42
4	.	.	.	IGLV2-23/IGLJ2_length_39
5	.	.	.	IGKV1-12/IGKJ1_length_33
6	.	.	.	IGKV3-20/IGKJ1_length_33
7	.	.	.	IGKV1-37/IGKJ4_length_33
8	.	.	.	IGKV2-28/IGKJ1_length_30
9	.	.	.	IGKV1-12/IGKJ3_length_33
10	.	.	.	IGLV2-14/IGLJ3_length_36

```
11      .      .      .   IGKV1-13/IGKJ4_length_33
12      .      .      .   IGLV7-46/IGLJ3_length_33
13      .      .      .   IGKV3-15/IGKJ4_length_33
14      .      .      .    IGKV4-1/IGKJ5_length_33
15      .      .      .   IGKV2-18/IGKJ2_length_33
```

Removi algumas colunas que não tinham nas colunas das tabelas originais. Substitui os nomes das colunas, afim de ficar com os nomes das colunas das tabelas originais.

```r
# funcao para processar o dataframe
processar_dataframe <- function(df){

  # remover as colunas
  df <- df %>%
    select(-c(V.end, D.start, D.end, J.start, VJ.ins, VD.ins, DJ.ins, Cluster))

  # alterar o nome das colunas
  names(df) <- c("count", "frequency", "CDR3nt", "CDR3aa", "V", "D", "J")

  # adicionar a coluna "C"
  df$C <- ifelse(substr(df$V, 1,3) == "IGL","IGLC",
                 ifelse(substr(df$V, 1,3) == "IGK","IGKC",
                        ifelse(substr(df$V, 1,3) == "IGH","IGHC",
                               ifelse(substr(df$V, 1,3) == "TRA","TRAC",
                                      ifelse(substr(df$V, 1,3) == "TRB","TRBC",
                                             ifelse(substr(df$V, 1,3) == "TRG","TRGC","TRD
  return(df)
}

# funcao para processar cada dataframe
processar_lista <- function(lista){
  nomes <- names(lista)
  for (i in seq_along(lista)) {
    lista[[i]] <- processar_dataframe(lista[[i]])
  }
  names(lista) <- nomes
  return(lista)
}

# aplicando as operacoes na lista de dataframes
lista_tcr_bcr_formatada <- processar_lista(lista_tcr_bcr)
```

Aqui está um exemplo de como ficou após essa terceira formatação:

```
lista_tcr_bcr_formatada$`130723_UNC9-SN296_0386_BC2E4WACXX_ACTTGA_L003_report`
```

	count	frequency	CDR3nt	CDR3aa	V	D	J	C
1	3	0.10000000	TGTGCCAGCAGCGTAGACCGGACAGGAGGGGACTATGGCTACACCTTC	CASSVDRTGGDYGYTF	TRBV9*01	TRBD1*01	TRBJ1-2*01	TRBC
2	1	0.03333333	TGTGCCAGCAGTTACTCCGGTACGGACGAGCAGTACTTC	CASSYSGTDEQYF	TRBV6-5*01	TRBD2*01	TRBJ2-7*01	TRBC
3	1	0.03333333	TGCAGCGTCTTACTAGCGGGGAGCACAGATACGCAGTATTTT	CSVLLAGSTDTQYF	TRBV29-1*01	TRBD2*01	TRBJ2-3*01	TRBC
4	7	0.23333333	TGCTGCTCATATGCAGGTAGTACCACTTTCGCGGTATTT	CCSYAGSTTFAVF	IGLV2-23*02	.	IGLJ2*01	IGLC
5	3	0.10000000	TGTCAACAGGCTTACAGTCCCCCTGAGACGTTC	CQQAYSPPETF	IGKV1-12*01	.	IGKJ1*01	IGKC
6	3	0.10000000	TGTCAGCAGTATGGTACCTCACCTGAAATGTTC	CQQYGTSPEMF	IGKV3-20*01	.	IGKJ1*01	IGKC
7	2	0.06666667	TGTCAAAAGTATGACAGTGTCCCGCTCACTTTC	CQKYDSVPLTF	IGKV1-37*01	.	IGKJ4*01	IGKC
8	2	0.06666667	TGCATGCAAACTCTACAAAGGGAGACGTTC	CMQTLQRETF	IGKV2-28*01	.	IGKJ1*01	IGKC
9	2	0.06666667	TGTCAACAGGCTCACAGTTTCCCCTTCACTTTC	CQQAHSFPFTF	IGKV1-12*01	.	IGKJ3*01	IGKC
10	1	0.03333333	TGCGGCTCATATACAAGCAGCAGCACTCGGGTGTTC	CGSYTSSSTRVF	IGLV2-14*01	.	IGLJ3*02	IGLC
11	1	0.03333333	TGTCAACAATTTAATAATTTCCCGCTCACTTTC	CQQFNNFPLTF	IGKV1-13*01	.	IGKJ4*01	IGKC
12	1	0.03333333	TGCTTGCTCTCCTATAATGGTCCTTGGGTGTTC	CLLSYNGPWVF	IGLV7-46*01	.	IGLJ3*02	IGLC
13	1	0.03333333	TGTCAGCAGTATAAAAACTGGCCTCCCACTTTC	CQQYKNWPPTF	IGKV3-15*01	.	IGKJ4*01	IGKC
14	1	0.03333333	TGTCAGCAATCTTATAGTGCTCCGATCACCTTC	CQQSYSAPITF	IGKV4-1*01	.	IGKJ5*01	IGKC
15	1	0.03333333	TGCATGCAAGCTACACAGTGGCCGTACACTTTT	CMQATQWPYTF	IGKV2-18*01	.	IGKJ2*01	IGKC

Precisei refazer o cálculo da frequência, pois como foi alterado as contagens, isso influencia na frequência.

```r
# funcao para atualizar os valores da coluna frequency

atualizar_frequency<- function(df){
  df %>%
    mutate(
      frequency = ifelse(substr(V,1,2) == "IG",
                         `count` / sum(filter(df, substr(V,1,2) == "IG")$`count`),
                         `count` / sum(filter(df, substr(V,1,2) == "TR")$`count`))
    )
}


# funcao para atualizar cada dataframe na lista
atualizar_lista <- function(lista){
  for (nome_df in names(lista)) {
    lista[[nome_df]] <- atualizar_frequency(lista[[nome_df]])
  }

  return(lista)
}


# aplica as operacoes na lista de dataframes
lista_tcr_bcr_formatada_frequency <- atualizar_lista(lista_tcr_bcr_formatada)
```

Por fim, salvei as tabelas no diretório "data" em "04_analytics/00_clones"

```r
# funcao para salvar cada dataframe individualmente
salvar_dataframes <- function(lista, dir_destino){
  for (nome_df in names(lista)) {
    arquivo <- paste0(dir_destino, "/", nome_df, ".tsv")
    write.table(lista[[nome_df]], arquivo, sep = "\t", row.names = FALSE)
  }
}

# salvar os dataframes como .tsv
salvar_dataframes(lista_tcr_bcr_formatada_frequency, dir_destino = "data")
```

```r
save(lista_tcr_bcr_formatada_frequency, file = "lista_tcrbcr_formatadaFrequency.RData")
```