My Bioinformatics Internship Report: Part I

Universitätsmedizin Mannheim, Institut für Kliniche Chemie

Author: Frano Malinarich G. Supervisors: Prof.Dr. Neumaier and Dr. Ast

2024-09-27

Table of contents

PART I

Introduction

This report summarizes my internship in Bioinformatics called "Single-cell transcriptomic analysis in the identification and characterization of VIREM cells applying knowledge of R, Python and Linux", conducted at the laboratory of Prof. Dr. Neumaier and Dr. Ast (Bioinformatician) at the Universitätsmedizin Mannheim, Institut für kliniche Chemie. The goal of my internship was to identify and characterize a novel class of monocytic cells, capable of recombining and expressing antibody genes and alpha and beta chains, a function previously thought to be a unique feature of B and T cells, respectively. These innate cells are called "variable immunoreceptor–expressing myeloid cells" (VIREMs) Those expressing B cell receptor chains (heavy and light) are called B-VIREMs and those alpha and beta chains, T-VIREMs.

In brief, my internship consisted in identifying and characterizing VIREMs peripheral blood mononuclear cells (PBMCs) from healthy humans, starting from T and B cell clonotypic analysis using the V(D)J library, next, cell clustering by analyzing the gene expression library, followed by data integration of both libraries and, finally, finding possible VIREMs.

For the VDJ and Gene expression analyses, the dataset* was obtained from 10x Genomics dataset repository:

https://www.10xgenomics.com/datasets/human-pbmc-from-a-healthy-donor-10-k-cells-v-2-2-standard-5-0-0

*PBMCs of a healthy male donor aged 27.

To learn how to do VDJ clonotipic analysis using scRepertoire package, go to this website:

https://www.bioconductor.org/packages/release/bioc/vignettes/scRepertoire/inst/doc/vignette.html

To learn how to do Gene expression analyses on human PBMCs using Seurat package, go to this website:

https://satijalab.org/seurat/articles/pbmc3k_tutorial.

In this "Part I", the analysis starts with the V(D)J library, firt with the analysis of TCR chains and, finally, BCR chains, performing the following:

- Combination of contigs
- Quantification of chains alleles per clone
- Frequencies of alleles
- Clonal visualizations
- Quantification/frequency of clones

Packages

```
# Load packages
if (!requireNamespace("pacman")) {
  install.packages("pacman")
}

pacman::p_load(conflicted, dplyr, tidyr, Seurat, scRepertoire, patchwork,
               ggplot2, ggrepel, pROC, flextable, ggraph, stringr)

conflict_prefer("filter", "dplyr")
conflict_prefer("select", "dplyr")
conflict_prefer("mutate", "dplyr")
conflict_prefer("slice", "dplyr")

# Activate libraries
suppressMessages(library(scRepertoire))
# install.packages("conflicted")
```

Setup

```
knitr::opts_chunk$set(fig.width = 7, fig.height = 4)  # Adjust figure size if applicable
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
set_flextable_defaults(font.size = 6, font.color = "black", border.color = "black", border.width
 ↪  = 1, padding = 3)
```

V(D)J clonotypic analysis

T cell receptor (TCR)

Load data and QC

The "VDJ_TCR_filtered_contig_annotation_3_10x.csv" file can be directly downloaded from the 10x Genomics website:

https://www.10xgenomics.com/datasets/human-pbmc-from-a-healthy-donor-10-k-cells-v-2-2-standard-5-0-0 Click on "VDJ TCR - Filtered contig annotations (CSV)"

```
# Load the sample 'filtered_contig_annotation_3_10x.csv' file using read.csv'
TCR_contigs <- read.csv(
  "Data_2024_Internship/VDJ_TCR_filtered_contig_annotation_3_10x.csv")

# Check the list of contigs
# View(TCR_contigs)
cat("Amount of barcodes and columns of .csv file:", dim(TCR_contigs))
```

```
Amount of barcodes and columns of .csv file: 10817 18
```

```
# QC: Identify false confidence, false productive and NAs
false_confidence <- TCR_contigs[TCR_contigs$high_confidence == FALSE, ]
false_productive <- TCR_contigs[TCR_contigs$productive == FALSE, ]
false_full_length <- TCR_contigs[TCR_contigs$full_length == FALSE, ]

# Counting the number of false confidence rows
false_confidence_count <- nrow(false_confidence)
false_productive_count <- nrow(false_productive)
false_full_length_count <- nrow(false_full_length)
cat("Amount of 'false confidence':", false_confidence_count, "\n")
```

```
Amount of 'false confidence': 0
```

```
cat("Amount of 'false productive':", false_productive_count, "\n")
```

```
Amount of 'false productive': 0
```

```
cat("Amount of 'false full length':", false_full_length_count, "\n")
```

```
Amount of 'false full length': 0
```

```
# Amount of NAs in the whole .csv file
cat("Amount of NAs:", sum(is.na(TCR_contigs)))
```

```
Amount of NAs: 0
```

Pairing contigs

```
# Combine contigs per each barcode to define clonotypes
S1_combTCR_allcontigs <- combineTCR(TCR_contigs,
                           removeNA = FALSE,
                           removeMulti = FALSE,
                           filterMulti = FALSE)

# View(S1_combTCR_allcontigs$S1)
cat("Amount of barcodes and columns of the combined contigs:", dim(S1_combTCR_allcontigs$S1))
```

```
Amount of barcodes and columns of the combined contigs: 5328 28
```

```
# Extract the TCR data for sample 'S1'
tcr_data <- S1_combTCR_allcontigs$S1
```

Table: Quantification of the amount of TCR alpha and beta chains per clonotype

```
# Define a function to identify and then count TCR chains
count_TCR_chains <- function(tcr1, tcr2) {
  TRA_count <- 0
  TRB_count <- 0

  # Count TRA chains in TCR1
  if (!is.na(tcr1)) {
    if (grepl("TRA", tcr1)) {
      TRA_count <- TRA_count + 1
    }
    if (grepl(";", tcr1)) {
      TRA_count <- TRA_count + 1
    }
  }

  # Count TRB chains in TCR2
  if (!is.na(tcr2)) {
    if (grepl("TRB", tcr2)) {
      TRB_count <- TRB_count + 1
    }
    if (grepl(";", tcr2)) {
      TRB_count <- TRB_count + 1
    }
  }

  return(c(TRA_count, TRB_count))
}

###### Make a Table: Pairing of alpha and beta chains per single barcode ########

# Apply the function to the TCR data
```

```r
tcr_chain_counts <- tcr_data %>%
  rowwise() %>%
  mutate(
    counts = list(count_TCR_chains(TCR1, TCR2)),
    TRA_count = counts[1],
    TRB_count = counts[2]
  ) %>%
  ungroup() %>%
  select(barcode, TRA_count, TRB_count, raw_consensus_id, TCR1, TCR2,
         CTgene, high_confidence, productive)

# Create a a first column called "Barcode_Number"
tcr_chain_counts <- tcr_chain_counts %>%
  mutate(Barcode_Number = row_number()) %>%
  relocate(Barcode_Number, .before = everything())

set.seed(1978)
# Sample 10 random rows
random_sample <- tcr_chain_counts[sample(nrow(tcr_chain_counts), 10), ]

# Select rows 15 to 40
# selected_rows <- tcr_chain_counts[15:40, ]
# Select specific columns (assuming you want to skip certain columns)
selected_columns <- random_sample[, c("Barcode_Number", "barcode", "TRA_count", "TRB_count",
 ↪  "TCR1", "TCR2")]

# Flextable
simple_table1 <- flextable(selected_columns) |>
  set_header_labels(Barcode_Number = "Barcode\n Number",
                    barcode = "Barcodes",
                    TRA_count = "Alpha\n count",
                    TRB_count = "Beta\n count",
                    TCR1 = "Alpha chain (TRA)",
                    TCR2 = "Beta chain (TRB)") |>
  add_header_lines(values = "Pairing of alpha and beta chains per single barcode in sample 3
   ↪  from 10x Genomics*") |>
  set_table_properties(layout = "autofit") |>
  theme_vanilla() |>
  colformat_char(na_str = "NA") |>
  width(j = ~ Barcode_Number + barcode + TRA_count + TRB_count + TCR1 + TCR2,
        width = c(0.8, 1.2, 0.8, 0.8, 2.0, 2.0)) |>  # Adjusting column widths manually
  fontsize(size = 7, part = "all") |>  # Reducing font size
  align(align = "center", part = "all") |>
  add_footer_lines(values = "*PBMC --> T-cells from healthy human, sequenced on Illumina NovaSeq
   ↪  6000.\n*10 randomly selected barcodes out of 5328 total.") |>
  fontsize(size = 6, part = "footer")

simple_table1
```

Pairing of alpha and beta chains per single barcode in sample 3 from 10x Genomics*					
Barcode Number	**Barcodes**	**Alpha count**	**Beta count**	**Alpha chain (TRA)**	**Beta chain (TRB)**
1,841	CCTAAAGAGCGCCTCA-1	0	1	NA	TRBV9.NA.TRBJ2-7.TRBC2
1,858	CCTACACAGATCACGG-1	1	1	TRAV9-2.TRAJ22.TRAC	TRBV6-5.NA.TRBJ1-2.TRBC1
3,560	GGACGTCTCTCGTATT-1	1	1	TRAV8-2.TRAJ3.TRAC	TRBV19.NA.TRBJ2-7.TRBC2
2,451	CTACATTGTGACAAAT-1	1	1	TRAV1-1.TRAJ13.TRAC	TRBV27.TRBD1.TRBJ2-7.TRBC2
1,018	ATAGACCTCACCAGGC-1	2	1	TRAV17.TRAJ53.TRAC;TRAV41.TRAJ32.TRAC	TRBV7-2.TRBD1.TRBJ1-5.TRBC1
1,840	CCTAAAGAGCAGATCG-1	1	1	TRAV12-2.TRAJ13.TRAC	TRBV4-3.TRBD2.TRBJ2-2.TRBC2
3,704	GGGAGATTCTAAGCCA-1	1	1	TRAV38-1.TRAJ4.TRAC	TRBV30.NA.TRBJ1-5.TRBC1
4,769	TGACTAGCACGGTGTC-1	1	1	TRAV8-4.TRAJ12.TRAC	TRBV9.NA.TRBJ2-7.TRBC2
1,422	CAGATCACATTACCTT-1	1	1	TRAV38-1.TRAJ17.TRAC	TRBV5-4.NA.TRBJ2-5.TRBC2
2,550	CTCAGAATCCTTTACA-1	1	1	TRAV10.TRAJ10.TRAC	TRBV9.NA.TRBJ2-3.TRBC2

*PBMC –> T-cells from healthy human, sequenced on Illumina NovaSeq 6000.*10 randomly selected barcodes out of 5328 total.

Table: Frequencies of different alpha and beta chains pairs

```
# Summarize the counts for barcodes with 1 and 2 alpha chains
summarize_alpha_chains <- tcr_chain_counts %>%
  summarize(
    total_barcodes = n(),
    zero_alpha_chain = sum(TRA_count == 0),
    one_alpha_chain = sum(TRA_count == 1),
    two_alpha_chains = sum(TRA_count == 2),
    zero_beta_chain = sum(TRB_count == 0),
    one_beta_chain = sum(TRB_count == 1),
    two_beta_chains = sum(TRB_count == 2)
  )

# Define the factor levels for all possible combinations
levels_alpha_beta <- c("1 / 0", "0 / 1", "1 / 1", "2 / 0", "0 / 2", "2 / 1", "1 / 2", "2 / 2")

# Create the new column with all possible levels
plot_tcr_chains <- tcr_chain_counts |>
  mutate(`alpha / beta chains` = factor(case_when(
    TRA_count == 1 & TRB_count == 0 ~ "1 / 0",
    TRA_count == 0 & TRB_count == 1 ~ "0 / 1",
    TRA_count == 1 & TRB_count == 1 ~ "1 / 1",
    TRA_count == 2 & TRB_count == 0 ~ "2 / 0",
    TRA_count == 0 & TRB_count == 2 ~ "0 / 2",
    TRA_count == 2 & TRB_count == 1 ~ "2 / 1",
    TRA_count == 1 & TRB_count == 2 ~ "1 / 2",
    TRA_count == 2 & TRB_count == 2 ~ "2 / 2"),
    levels = levels_alpha_beta))

# Calculate the counts and percentages for each category
```

```r
counts <- plot_tcr_chains %>%
  group_by(`alpha / beta chains`) %>%
  summarize(count = n(), .groups = 'drop') %>%
  mutate(percentage = round(count / sum(count),5) * 100) |>
  complete(`alpha / beta chains` = levels_alpha_beta,  # Add missing levels with zero counts
           fill = list(count = 0, percentage = 0)) |>
  arrange(desc(count))

########### Make a TABLE: Frequencies of alpha ad beta chain pairs ###############

# Add a total row
total_row <- counts %>%
  summarise(`alpha / beta chains` = "Total",
            count = sum(count),
            percentage = round(sum(percentage),2))

# Combine the counts and total row
counts_with_total <- bind_rows(counts, total_row)

# Create the flextable
ft <- flextable(counts_with_total) %>%
  set_header_labels(`alpha / beta chains` = "Alpha / Beta chains",
                    count = "Count",
                    percentage = "% of total Barcodes") %>%
  add_header_lines(values = "Frequencies of alpha and beta chain pairs in sample 3 10x
   ↪  Genomics") |>
  set_table_properties(layout = "autofit") |>
  bold(i = nrow(counts_with_total)) %>%  # bold the last row (total row)
  theme_vanilla() %>%
  fontsize(size = 8, part = "all")

# Format the count and percentage columns using colformat_num()
ft <- colformat_num(ft,
                    j = c("count", "percentage"),  # Column names to format
                    big.mark = ".",                 # Thousands separator
                    decimal.mark = ",")             # Decimal mark

ft <- align(ft, align = "center", part = "header")
ft <- align(ft, align = "center", part = "body")

ft
```

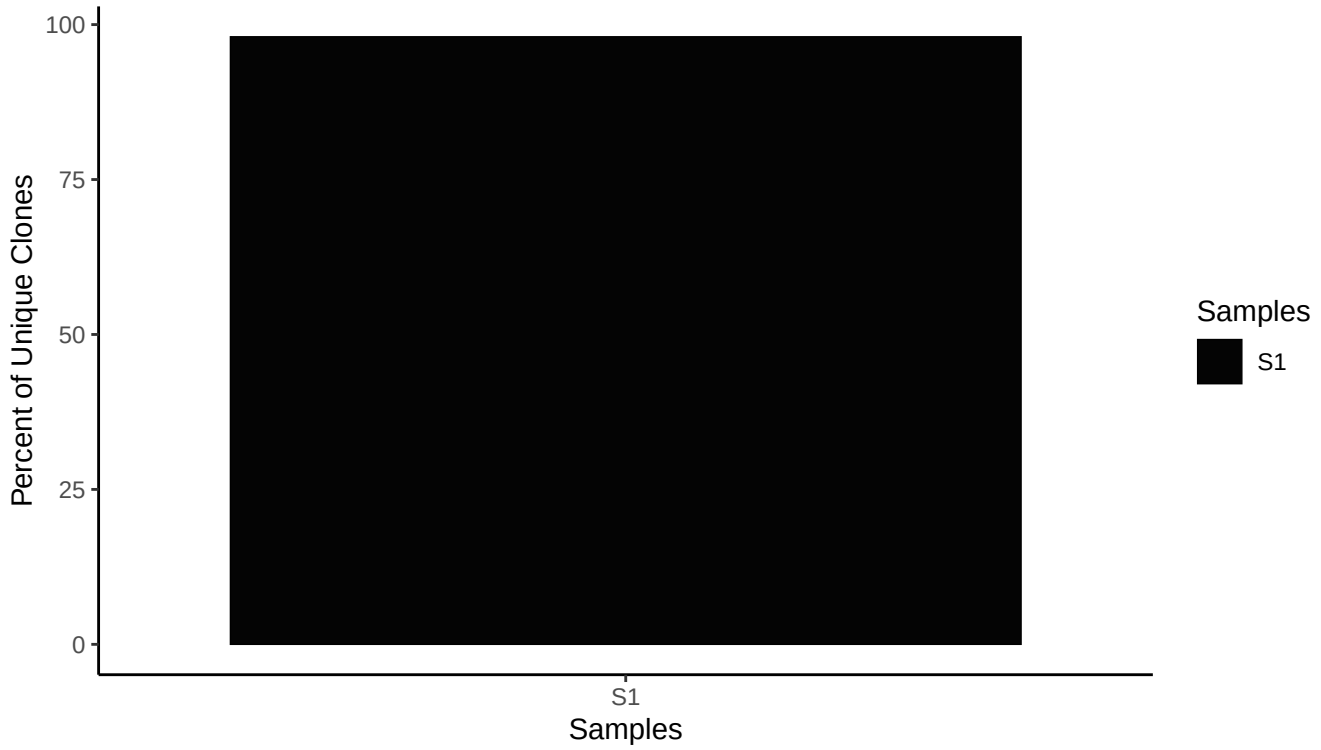
Frequencies of alpha and beta chain pairs in sample 3 10x Genomics

Alpha / Beta chains	Count	% of total Barcodes
1 / 1	4.070	76,389
0 / 1	498	9,347
2 / 1	466	8,746
1 / 2	116	2,177
1 / 0	93	1,745

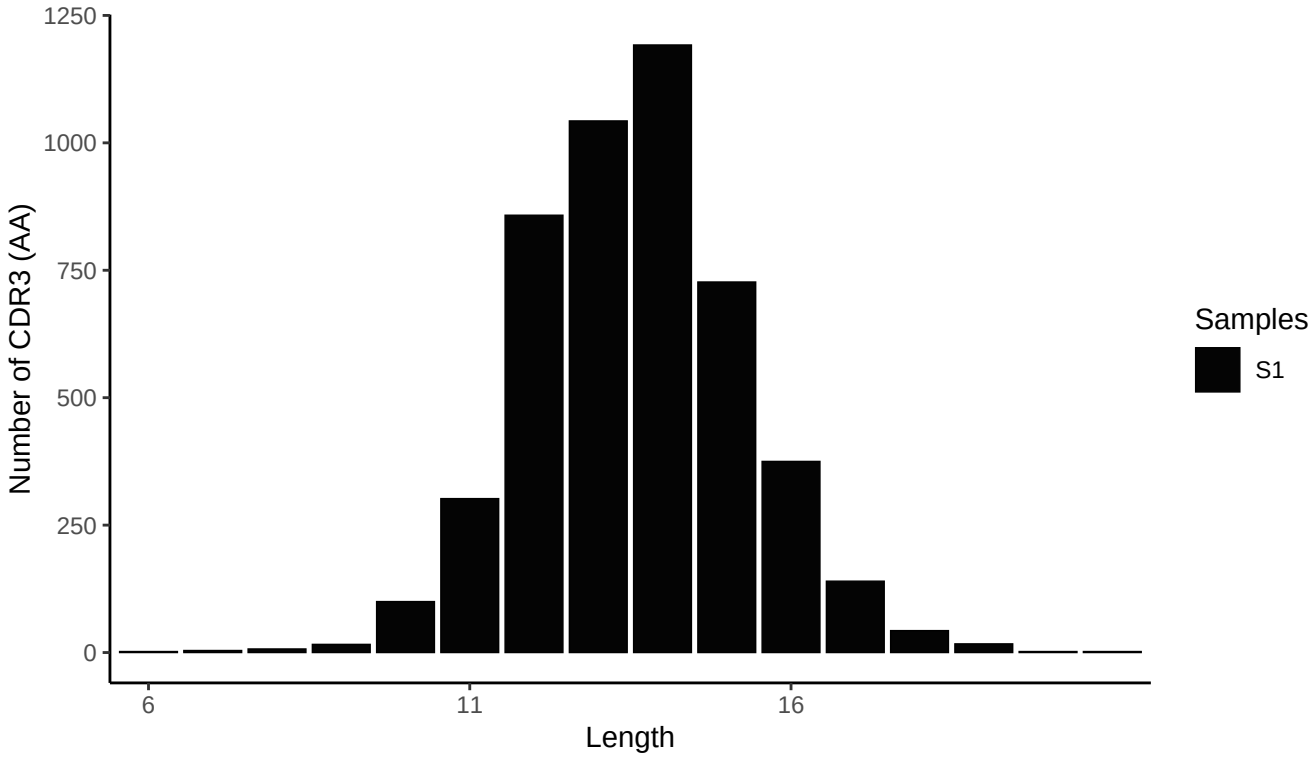
Frequencies of alpha and beta chain pairs in sample 3 10x Genomics		
Alpha / Beta chains	**Count**	**% of total Barcodes**
2 / 2	85	1,595
0 / 2	0	0,000
2 / 0	0	0,000
Total	**5.328**	**100,000**

Clonal visualizations: Unique clones, CDR3, alpha/beta chain compositions

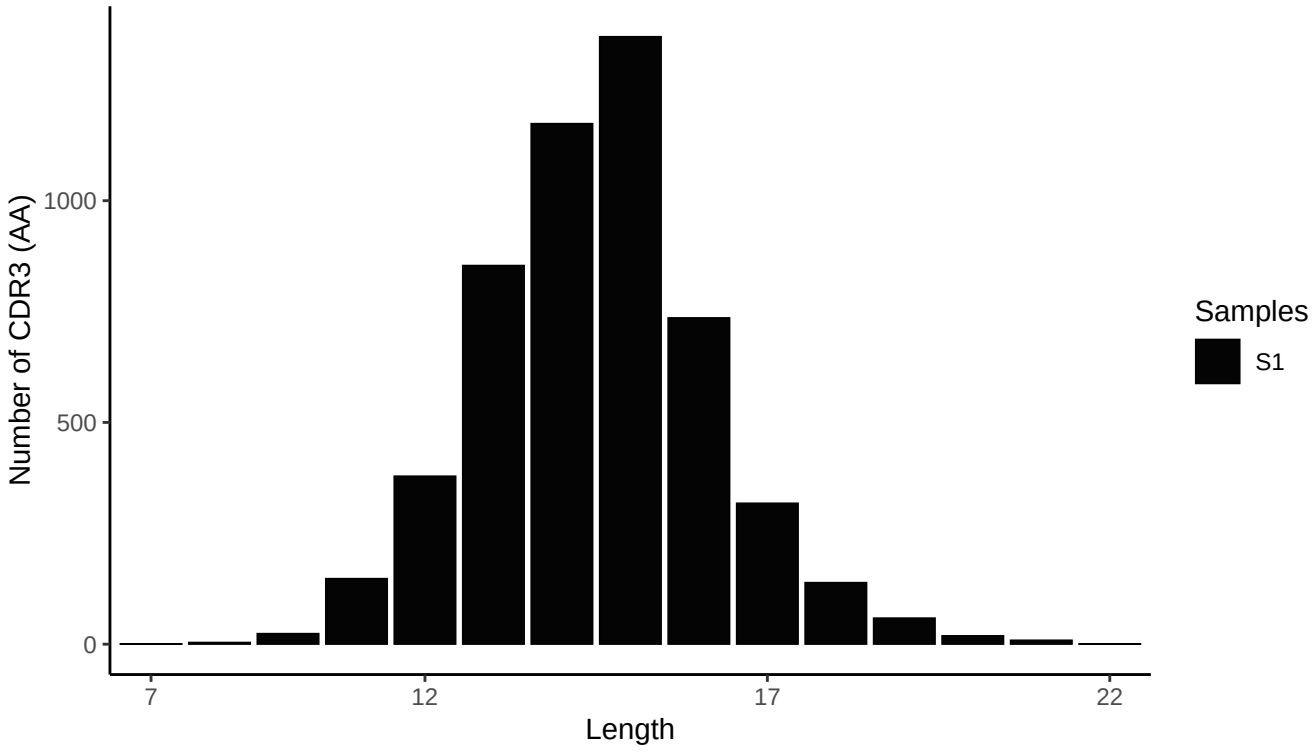
```
# Total number of unique clones
clonalQuant(S1_combTCR_allcontigs,
            cloneCall="strict",
            chain = "TRB",
            scale = TRUE) # Around 5000 unique clones out of 5328
```



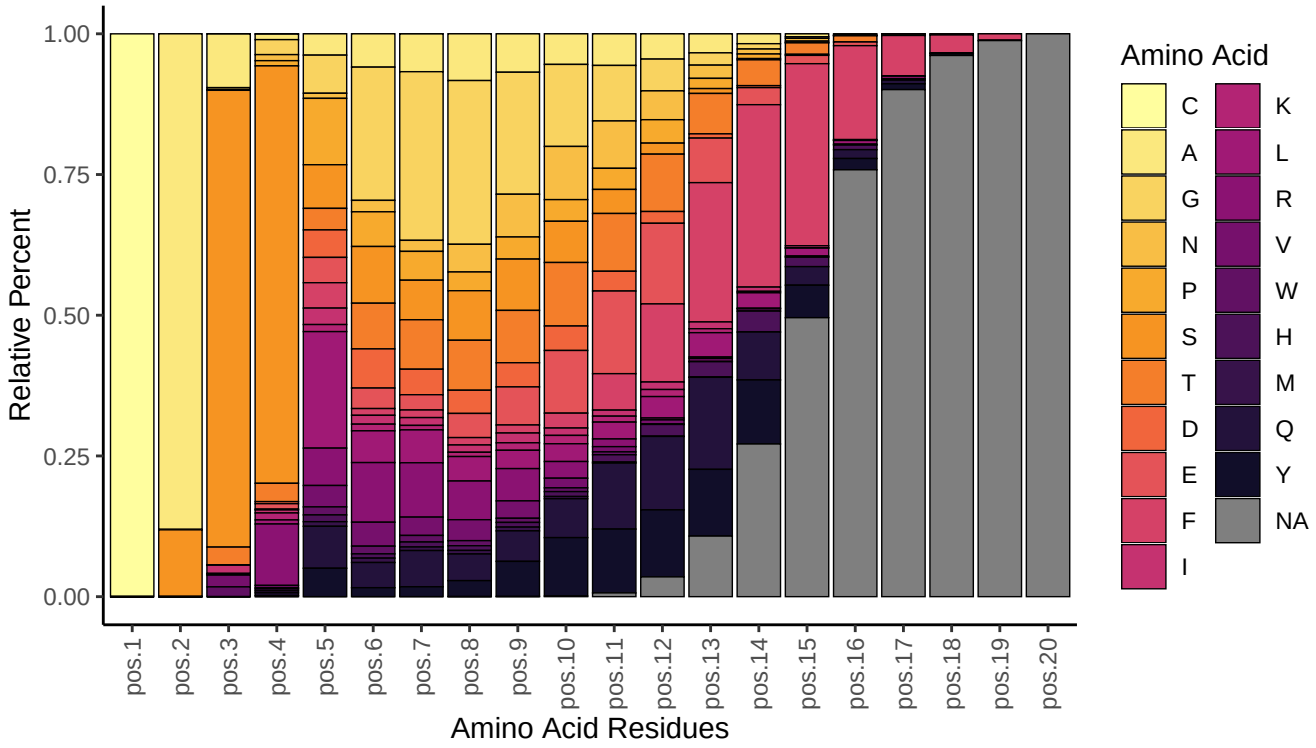

```
# Checking the length distribution and "aa" composition of CDR3 sequences
clonalLength(S1_combTCR_allcontigs,
             cloneCall="aa",
             chain = "TRA")
```



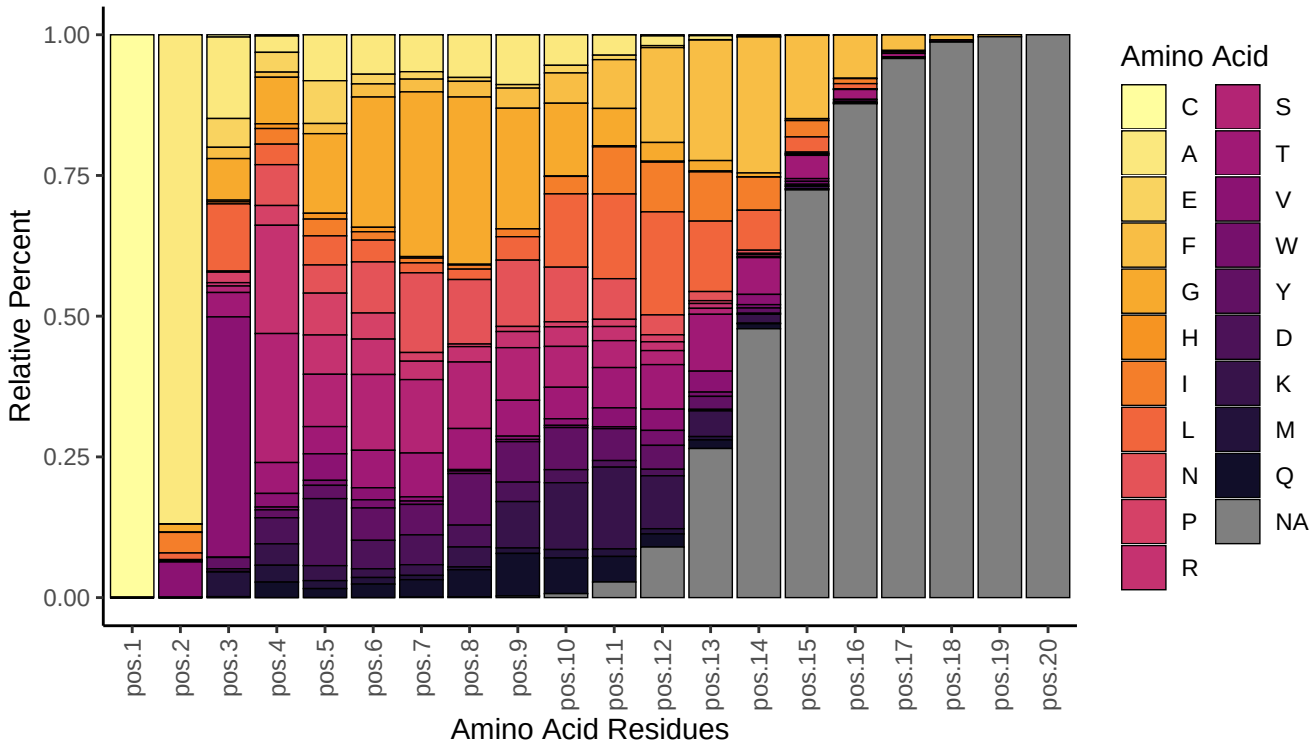

```
clonalLength(S1_combTCR_allcontigs,
             cloneCall="aa",
             chain = "TRB")
```



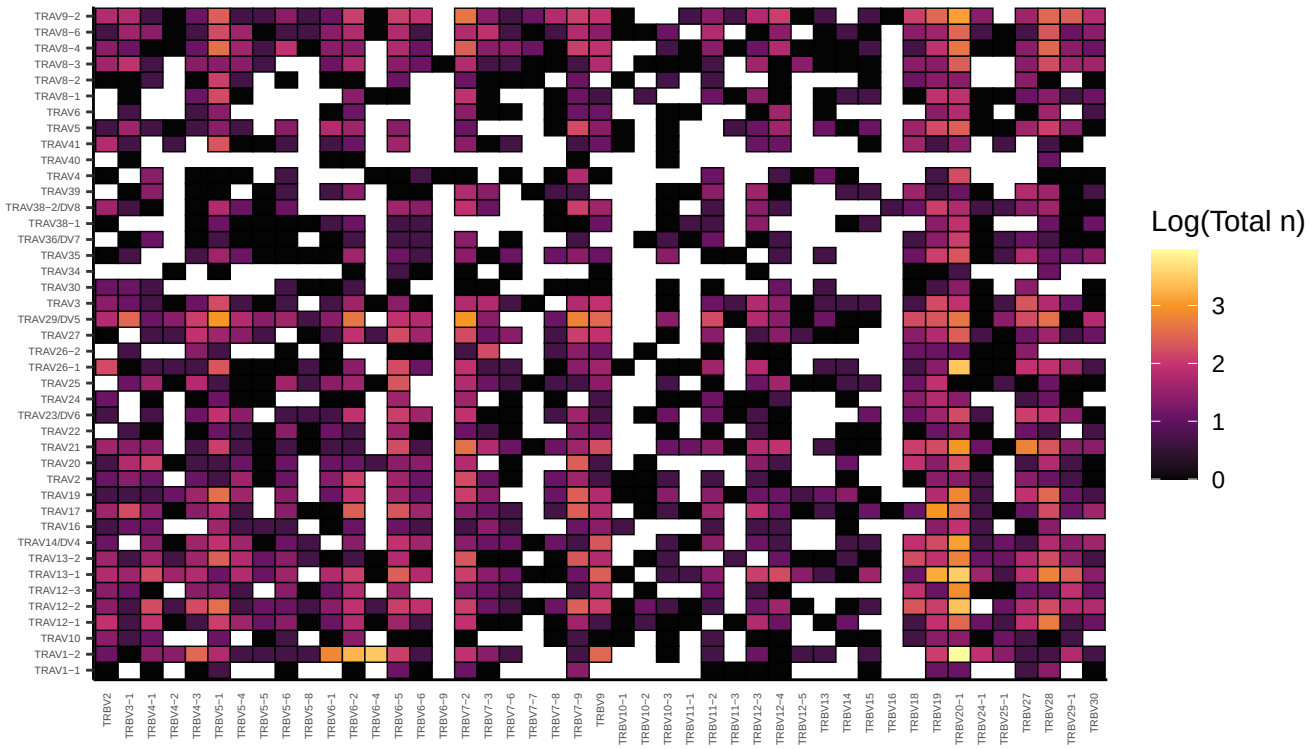
```
percentAA(S1_combTCR_allcontigs,
          chain = "TRB",
          aa.length = 20)
```



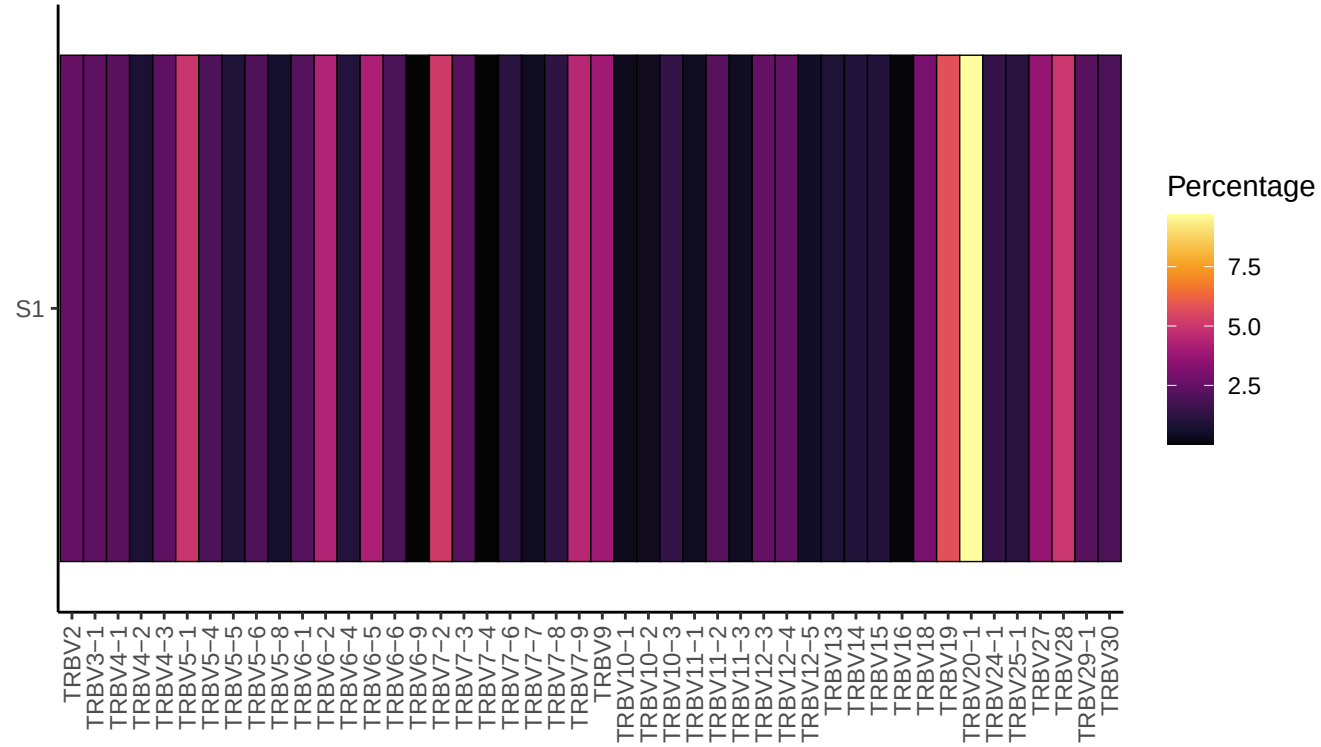

```
percentAA(S1_combTCR_allcontigs,
          chain = "TRA",
          aa.length = 20)
```



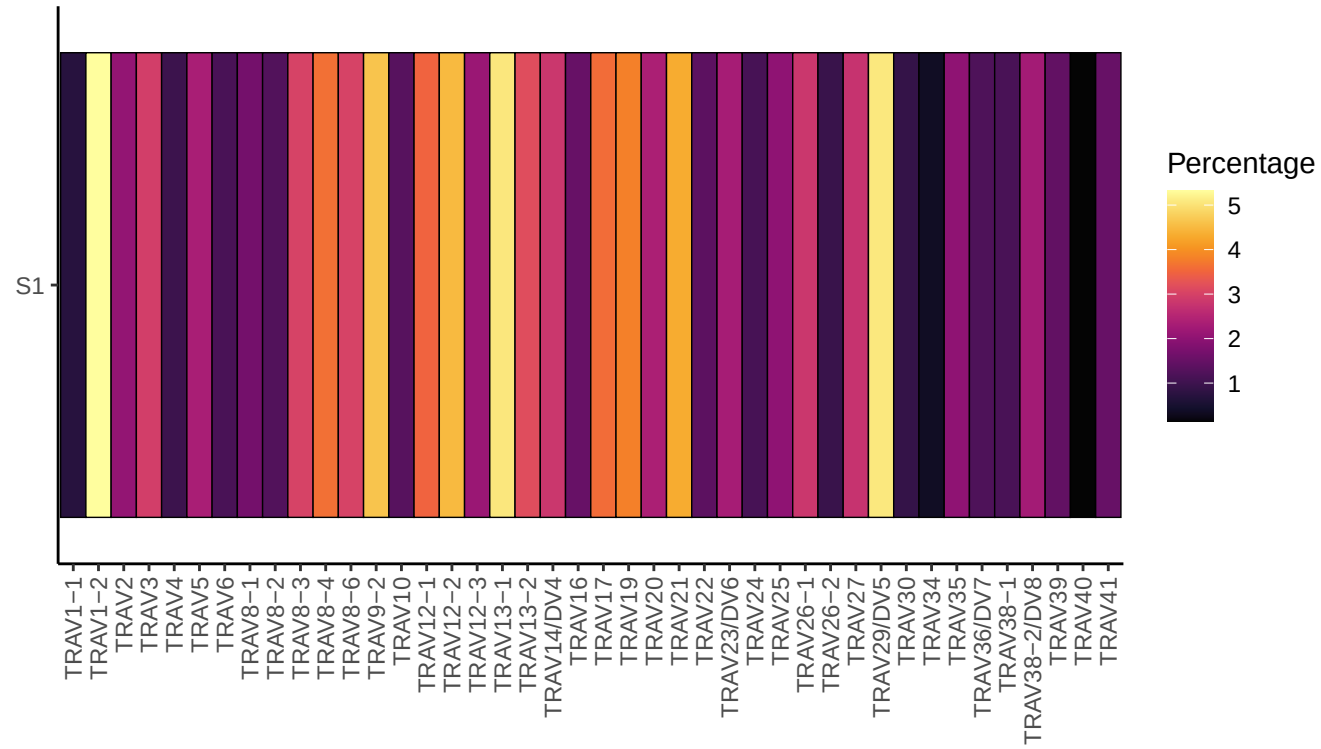

```
# Heatmat of frequencies of TRAV and TRBV pairing
viz2 <- vizGenes(tcr_data, # It can be also 'tcr_data'
         x.axis = "TRBV",
         y.axis = "TRAV",
         plot = "heatmap",
         scale = FALSE)
viz2
```



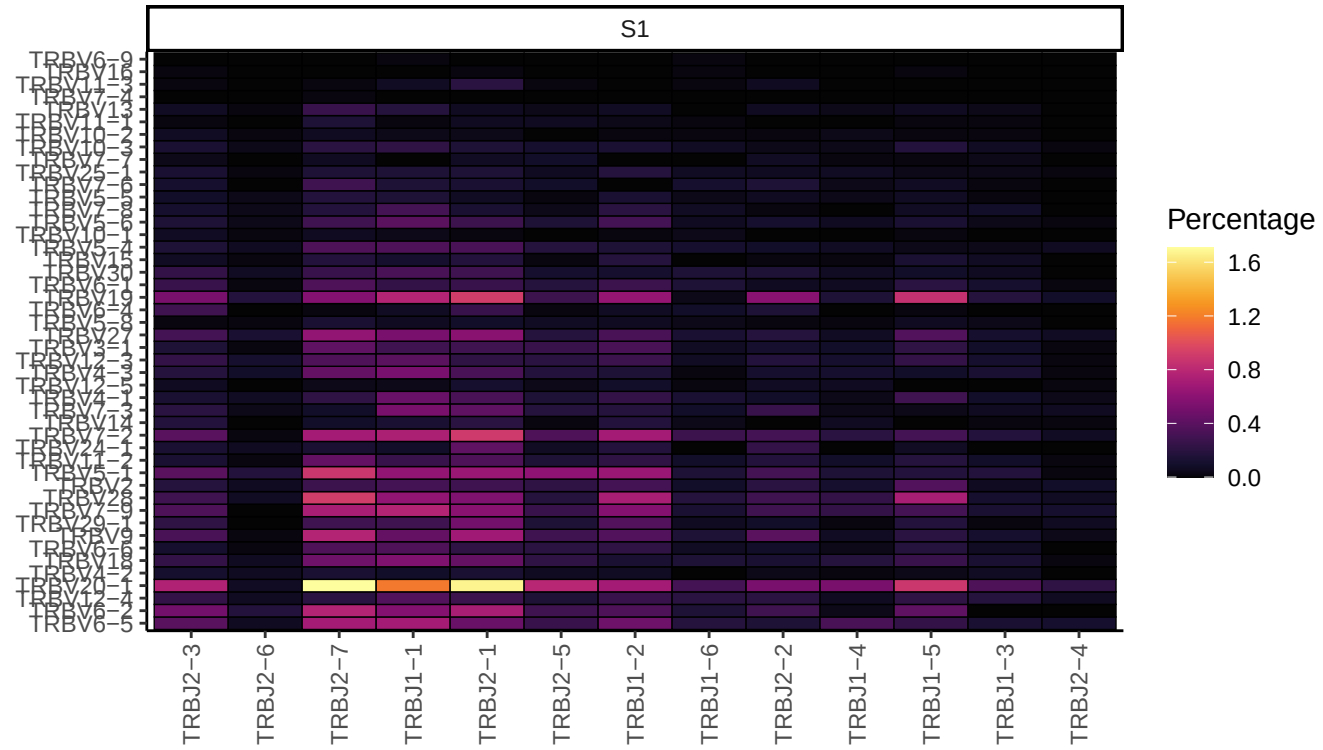

```
# Percentage of TRBV, TRAV and pairing TRBV with TRBJ in T cells
percentGenes(S1_combTCR_allcontigs,
             chain = "TRB",
             gene = "Vgene")
```




```
percentGenes(S1_combTCR_allcontigs,
             chain = "TRA",
             gene = "Vgene")
```



```
percentVJ(S1_combTCR_allcontigs,
          chain = "TRB")
```



Most frequent T cell clones: pairing of alpha and beta chains and cell frequencies

```
# This way of counting T cell clones includes those that do not have
# a chain and appear with NA, for example, those with TRA and TRB pairings like
# 1/0 and 0/1

# Count the occurrences of each TCR1 (TRA) and TCR2 (TRB) pair
pair_counts <- tcr_data %>%
  group_by(TCR1, TCR2) %>%
  summarize(count = n(), .groups = "drop") %>%
  arrange(desc(count))

# Calculate frequencies and percentages
total_counts <- sum(pair_counts$count)
pair_counts <- pair_counts %>%
  mutate(frequency = count / total_counts,
         log2_frequency = log2(frequency),
         Percentage = (count / total_counts) * 100)

nrow(pair_counts) # 4972 are the rows, but the total T cell clones are 5328
```

```
[1] 4972
```

```r
# Combine TCR1 and TCR2 into a single column
pair_counts <- pair_counts %>%
  dplyr::mutate(T_cell_clones = paste(TCR1, TCR2, sep = "_"))

# Function to count the number of alpha and beta chains
count_alpha_beta <- function(clone) {
  alpha_count <- str_count(clone, "TRAV")
  beta_count <- str_count(clone, "TRBV")
  return(paste(alpha_count, beta_count, sep = "/"))
}

# # Combine TCR1 and TCR2 into a single column and add the number of chains
# pair_counts <- pair_counts %>%
#   mutate(T_cell_clones = paste(TCR1, TCR2, sep = "_")) %>%
#   mutate(Alpha_Beta_chains = sapply(T_cell_clones, count_alpha_beta))

# Add the 'Alpha/Beta chains' column
pair_counts <- pair_counts %>%
  mutate(Alpha_Beta_chains = sapply(T_cell_clones, count_alpha_beta)) %>%
  select(`T cell clones` = T_cell_clones,
         `Alpha/Beta chains` = Alpha_Beta_chains,
         count,
         Percentage)

# Select the top N combinations for lollipop plot or to display
N <- 10 # Change this value to any number you like
topN_pairs_all <- pair_counts %>%
  slice(1:N)

# Select columns for the flextable
topN_pairs_all <- topN_pairs_all %>%
  select(`T cell clones`, `Alpha/Beta chains`, count, Percentage)

# Create a new flextable including the Alpha/Beta chains column
ft3 <- flextable(topN_pairs_all)

# Format the table
ft3 <- ft3 %>%
  set_header_labels(T_cell_clones = "T cell clones",
                    `Alpha/Beta chains` = "Alpha/Beta chains",
                    count = "Count",
                    Percentage = "Percentage") %>%
  add_header_lines(values = "Top 10 most frequent T cell clones, alpha/beta chains pairing and
   ↪  frequencies\n") |>
  set_table_properties(layout = "autofit") |>
  theme_vanilla() |>
  fontsize(size = 8, part = "all")

# Align the table
ft3 <- align(ft3, align = "center", part = "header")
ft3 <- align(ft3, align = "center", part = "body")

# Display
```

```
ft3
```

Top 10 most frequent T cell clones, alpha/beta chains pairing and frequencies

T cell clones	Alpha/Beta chains	Count	Percentage
TRAV1-2.TRAJ33.TRAC_TRBV20-1.NA.TRBJ2-1.TRBC2	1/1	13	0.24399399
TRAV26-2.TRAJ24.TRAC_TRBV7-3.NA.TRBJ1-1.TRBC1	1/1	9	0.16891892
TRAV1-2.TRAJ33.TRAC_TRBV20-1.NA.TRBJ2-7.TRBC2	1/1	8	0.15015015
TRAV1-2.TRAJ33.TRAC_NA	1/0	8	0.15015015
TRAV17.TRAJ16.TRAC_TRBV19.NA.TRBJ2-1.TRBC2	1/1	8	0.15015015
TRAV1-2.TRAJ33.TRAC_TRBV6-4.NA.TRBJ2-2.TRBC2	1/1	7	0.13138138
NA_TRBV20-1.NA.TRBJ1-1.TRBC1	0/1	7	0.13138138
NA_TRBV20-1.NA.TRBJ2-1.TRBC2	0/1	7	0.13138138
NA_TRBV9.NA.TRBJ2-7.TRBC2	0/1	6	0.11261261
TRAV1-2.TRAJ33.TRAC_TRBV6-4.NA.TRBJ2-1.TRBC2	1/1	5	0.09384384

```
# Write to CSV file
# write.csv(pair_counts, "Data_2024_Internship/Alpha_Beta_Tcellclones_3_10x.csv", row.names =
 ↪  FALSE)

# ft3 <- bg(ft3, bg = "white", part = "all")
# save_as_image(x = ft3, path =
 ↪  "Data_2024_Internship/Top_H_L_pairs_constant_regions_table_all_3_10x.png")
```

B cell receptor (BCR)

Load data and QC

The "VDJ_TCR_filtered_contig_annotation_3_10x.csv" file can be directly downloaded from the 10x Genomics website:

https://www.10xgenomics.com/datasets/human-pbmc-from-a-healthy-donor-10-k-cells-v-2-2-standard-5-0-0 Click on "VDJ Ig - Filtered contig annotations (CSV)"

```
# Load the sample 'filtered_contig_annotation_3_10x.csv' file using read.csv'
BCR_contigs <- read.csv(
  "Data_2024_Internship/VDJ_IG_filtered contig annotations_3_10x.csv")

# Check the list of contigs
# View(BCR_contigs)
cat("Amount of barcodes and columns of .csv file:", dim(BCR_contigs))
```

```
Amount of barcodes and columns of .csv file: 2066 18
```

```
# QC: Identify false confidence, false productive and NAs
false_confidence <- BCR_contigs[BCR_contigs$high_confidence == FALSE, ]
false_productive <- BCR_contigs[BCR_contigs$productive == FALSE, ]
false_full_length <- BCR_contigs[BCR_contigs$full_length == FALSE, ]

# Counting the number of false confidence rows
false_confidence_count <- nrow(false_confidence)
false_productive_count <- nrow(false_productive)
false_full_length_count <- nrow(false_full_length)
cat("Amount of 'false confidence':", false_confidence_count, "\n")
```

```
Amount of 'false confidence': 0
```

```
cat("Amount of 'false productive':", false_productive_count, "\n")
```

```
Amount of 'false productive': 0
```

```
cat("Amount of 'false full length':", false_full_length_count, "\n")
```

```
Amount of 'false full length': 0
```

```
# Amount of NAs in the whole .csv file
cat("Amount of NAs:", sum(is.na(BCR_contigs)))
```

```
Amount of NAs: 0
```

Pairing contigs

```
# Combine contigs per each barcode to define clonotypes
S1_combBCR_allcontigs <- combineBCR(BCR_contigs,
                                    samples = "S1",
                                    threshold = 0.85,
                                    removeNA = FALSE,
                                    removeMulti = FALSE,
                                    filterMulti = FALSE)

# View(S1_combBCR_allcontigs$S1)
cat("Amount of barcodes and columns of the combined contigs:", dim(S1_combBCR_allcontigs$S1)) #
 ↪  982   28
```

```
Amount of barcodes and columns of the combined contigs: 982 12
```

```
# Extract the BCR data for sample 'S1'
bcr_data <- S1_combBCR_allcontigs$S1
```

Table: Quantification of the amount of BCR heavy and light chains per clonotype

```
# Define a function to identify and then count IGH and IGL chains
count_BCR_chains <- function(igh, igl) {
  IGH_count <- 0
  IGL_count <- 0

  # Count igh chains in IGH
  if (!is.na(igh)) {
    if (grepl("IGH", igh)) {
      IGH_count <- IGH_count + 1
    }
    if (grepl(";", igh)) {
      IGH_count <- IGH_count + 1
    }
  }

  # Count igl chains in IGLC
  if (!is.na(igl)) {
    if (grepl("IGK", igl) || grepl("IGL", igl)) {
      IGL_count <- IGL_count + 1
    }
    if (grepl(";", igl)) {
      IGL_count <- IGL_count + 1
    }
  }

  return(c(IGH_count, IGL_count))
}

###### Make a Table: Pairing of igh and igl chains per single barcode ########

# Apply the function to the BCR data
```

```r
bcr_chain_counts <- bcr_data %>%
  rowwise() %>%
  mutate(
    counts = list(count_BCR_chains(IGH, IGLC)),
    IGH_count = counts[1],
    IGL_count = counts[2]
  ) %>%
  ungroup() %>%
  select(barcode, IGH_count, IGL_count, IGH, IGLC,
         CTgene)

# View(bcr_chain_counts)
dim(bcr_chain_counts)
```

```
[1] 982   6
```

```r
# Create a a first column called "Barcode_Number"
bcr_chain_counts <- bcr_chain_counts %>%
  mutate(Barcode_Number = row_number()) %>%
  relocate(Barcode_Number, .before = everything())

set.seed(2000)
# Sample 10 random rows
random_sample <- bcr_chain_counts[sample(nrow(bcr_chain_counts), 10), ]

# Select rows 15 to 40
# selected_rows <- bcr_chain_counts[15:40, ]
# Select specific columns (assuming you want to skip certain columns)
selected_columns <- random_sample[, c("Barcode_Number", "barcode", "IGH_count", "IGL_count",
 ↪  "IGH", "IGLC")]#, "CTgene")]

# Flextable
simple_table2 <- flextable(selected_columns) |>
  # separate_header(split = "_") |>
  set_header_labels(Barcode_Number = "Barcode\n Number",
                    barcode = "Barcodes",
                    IGH_count = "IGH count",
                    IGL_count = "IGL count",
                    IGH = "Heavy chain (IGH)",
                    IGLC = "Light chain (IGK/L)") |>
  add_header_lines(values = "Pairing of heavy and light chains per single barcode in sample 3
   ↪  10x Genomics*") |>
  set_table_properties(layout = "autofit") |>
  theme_vanilla() |>
  colformat_char(na_str = "NA") |>
   width(j = ~ Barcode_Number + barcode + IGH_count + IGL_count + IGH + IGLC,
         width = c(0.8, 1.2, 0.8, 0.8, 2.0, 2.0)) |>  # Adjusting column widths manually
  fontsize(size = 7, part = "all") |>  # Reducing font size
  align(align = "center", part = "all") |>
  add_footer_lines(values = "*PBMC --> B-cells from healthy human, sequenced on Illumina NovaSeq
   ↪  6000.\n*10 randomly selected barcodes out of 982 total.") |>
  fontsize(size = 6, part = "footer")
```

```
simple_table2
```

Pairing of heavy and light chains per single barcode in sample 3 10x Genomics*					
Barcode Number	**Barcodes**	**IGH count**	**IGL count**	**Heavy chain (IGH)**	**Light chain (IGK/L)**
597	S1_ACACCCTTCAGAAATG-1	1	1	IGHV3-30.NA.IGHJ4.IGHM	IGKV3-15.IGKJ1.IGKC
872	S1_GAATAAGAGGGTCTCC-1	1	1	IGHV2-26.NA.IGHJ5.IGHM	IGLV3-21.IGLJ1.IGLC1
178	S1_CATCCACTCAAGAAGT-1	1	1	IGHV1-24.NA.IGHJ4.IGHM	IGKV1-5.IGKJ1.IGKC
30	S1_ACGGGTCCAAGTCATC-1	1	1	IGHV1-8.NA.IGHJ4.IGHM	IGLV2-14.IGLJ2.IGLC2
53	S1_CTAGAGTTCATCTGTT-1	1	1	IGHV1-2.NA.IGHJ5.IGHM	IGLV2-14.IGLJ2.IGLC2
399	S1_TTCTCCTTCGACGGAA-1	1	1	IGHV3-23.NA.IGHJ4.IGHM	IGKV1D-39.IGKJ2.IGKC
398	S1_GGGTCTGCATCCCATC-1	1	1	IGHV4-31.NA.IGHJ4.IGHM	IGKV1D-39.IGKJ2.IGKC
527	S1_GGACGTCCAAGTTAAG-1	1	1	IGHV4-39.NA.IGHJ5.IGHM	IGKV1-8.IGKJ1.IGKC
504	S1_ACTATCTCATAAGACA-1	0	1	NA	IGKV1-33.IGKJ3.IGKC
197	S1_CAGATCACAGCCACCA-1	1	1	IGHV1-8.IGHD6-13.IGHJ6.IGHM	IGLV1-40.IGLJ2.IGLC2

*PBMC –> B-cells from healthy human, sequenced on Illumina NovaSeq 6000.*10 randomly selected barcodes out of 982 total.

```
# simple_table2 <- bg(simple_table2, bg = "white", part = "all")
# save_as_image(x = simple_table2, path = "Pairing_H_L_pairs_table_3_10x.png") # ggsave doesn't
 ↪  work with flextable, only ggplot
```

Table: Frequencies of different heavy and light chains pairs

```
# Summarize the counts for barcodes with 1 and 2 heavy or light chains
summarize_igh_chains <- bcr_chain_counts %>%
  summarize(
    total_barcodes = n(),
    zero_igh_chain = sum(IGH_count == 0),
    one_igh_chain = sum(IGH_count == 1),
    two_igh_chains = sum(IGH_count == 2),
    zero_igl_chain = sum(IGL_count == 0),
    one_igl_chain = sum(IGL_count == 1),
    two_igl_chains = sum(IGL_count == 2)
  )
# summarize_igh_chains

############ Make a PLOT: Frequencies of heavy ad light chain pairs ##############

# Define the factor levels for all possible combinations
levels_igh_igl <- c("1 / 0", "0 / 1", "1 / 1", "2 / 0", "0 / 2", "2 / 1", "1 / 2", "2 / 2")

# Create the new column with all possible levels
plot_bcr_chains <- bcr_chain_counts |>
```

```r
  mutate(`IgH / IgL chains` = factor(case_when(
    IGH_count == 1 & IGL_count == 0 ~ "1 / 0",
    IGH_count == 0 & IGL_count == 1 ~ "0 / 1",
    IGH_count == 1 & IGL_count == 1 ~ "1 / 1",
    IGH_count == 2 & IGL_count == 0 ~ "2 / 0",
    IGH_count == 0 & IGL_count == 2 ~ "0 / 2",
    IGH_count == 2 & IGL_count == 1 ~ "2 / 1",
    IGH_count == 1 & IGL_count == 2 ~ "1 / 2",
    IGH_count == 2 & IGL_count == 2 ~ "2 / 2"),
    levels = levels_igh_igl))

# View(plot_bcr_chains)

# Calculate the counts and percentages for each category
counts <- plot_bcr_chains %>%
  group_by(`IgH / IgL chains`) %>%
  summarize(count = n(), .groups = 'drop') %>%
  mutate(percentage = round(count / sum(count),5) * 100) |>
  complete(`IgH / IgL chains` = levels_igh_igl,  # Add missing levels with zero counts
           fill = list(count = 0, percentage = 0)) |>
  arrange(desc(count))

############ Make a TABLE: Frequencies of heavy ad light chain pairs ###############

# Add a total row
total_row <- counts %>%
  summarise(`IgH / IgL chains` = "Total",
            count = sum(count),
            percentage = round(sum(percentage),2))

# Combine the counts and total row
counts_with_total <- bind_rows(counts, total_row)

# Create the flextable
ft4 <- flextable(counts_with_total) %>%
  set_header_labels(`IgH / IgL chains` = "Heavy / Light chains",
                    count = "Count",
                    percentage = "% of total Barcodes") %>%
  add_header_lines(values = "Frequencies of heavy and light chain pairs in sample 3 10x
   ↪  Genomics") |>
  set_table_properties(layout = "autofit") |>
  bold(i = nrow(counts_with_total)) %>%  # bold the last row (total row)
  theme_vanilla() %>%
  fontsize(size = 8, part = "all")

# Format the count and percentage columns using colformat_num()
ft4 <- colformat_num(ft4,
                     j = c("count", "percentage"),  # Column names to format
                     big.mark = ".",                 # Thousands separator
                     decimal.mark = ",")             # Decimal mark

ft4 <- align(ft4, align = "center", part = "header")
ft4 <- align(ft4, align = "center", part = "body")
```

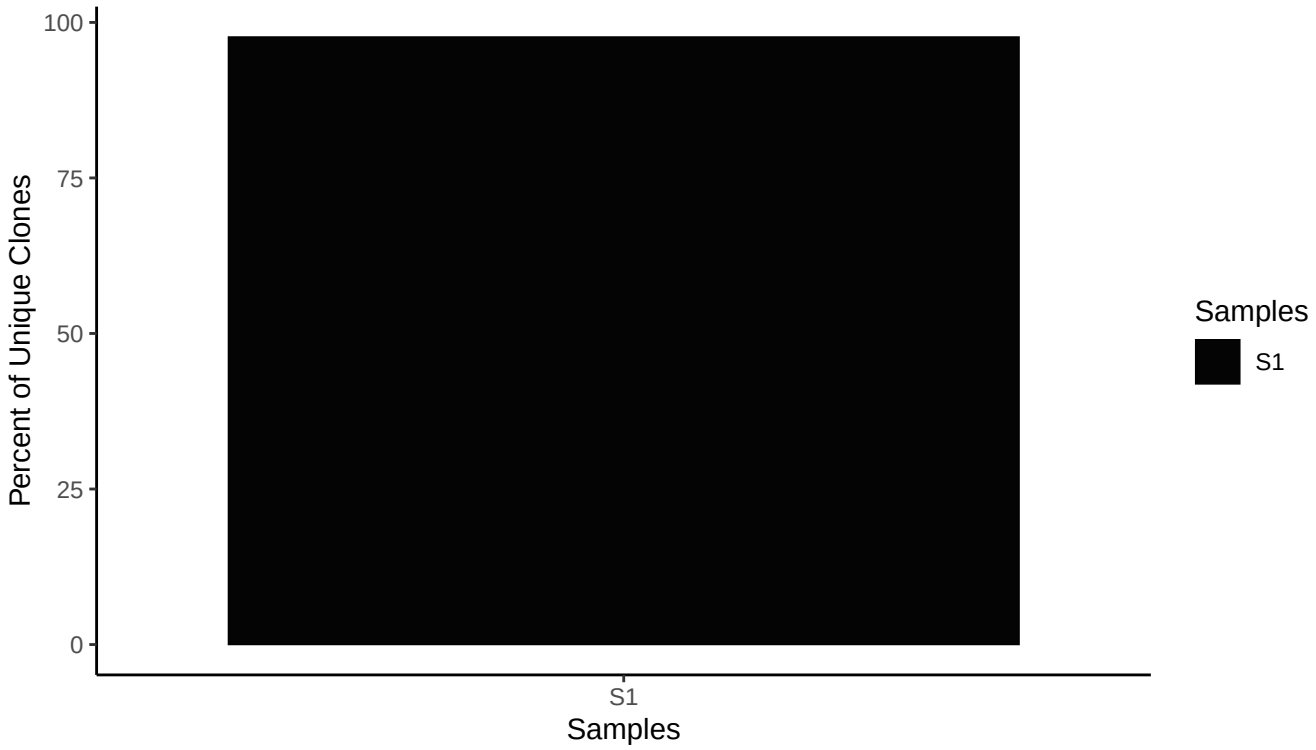
```
ft4
```

Frequencies of heavy and light chain pairs in sample 3 10x Genomics		
Heavy / Light chains	**Count**	**% of total Barcodes**
1 / 1	883	89,919
2 / 2	45	4,582
1 / 2	31	3,157
0 / 1	16	1,629
1 / 0	5	0,509
2 / 1	2	0,204
0 / 2	0	0,000
2 / 0	0	0,000
Total	**982**	**100,000**

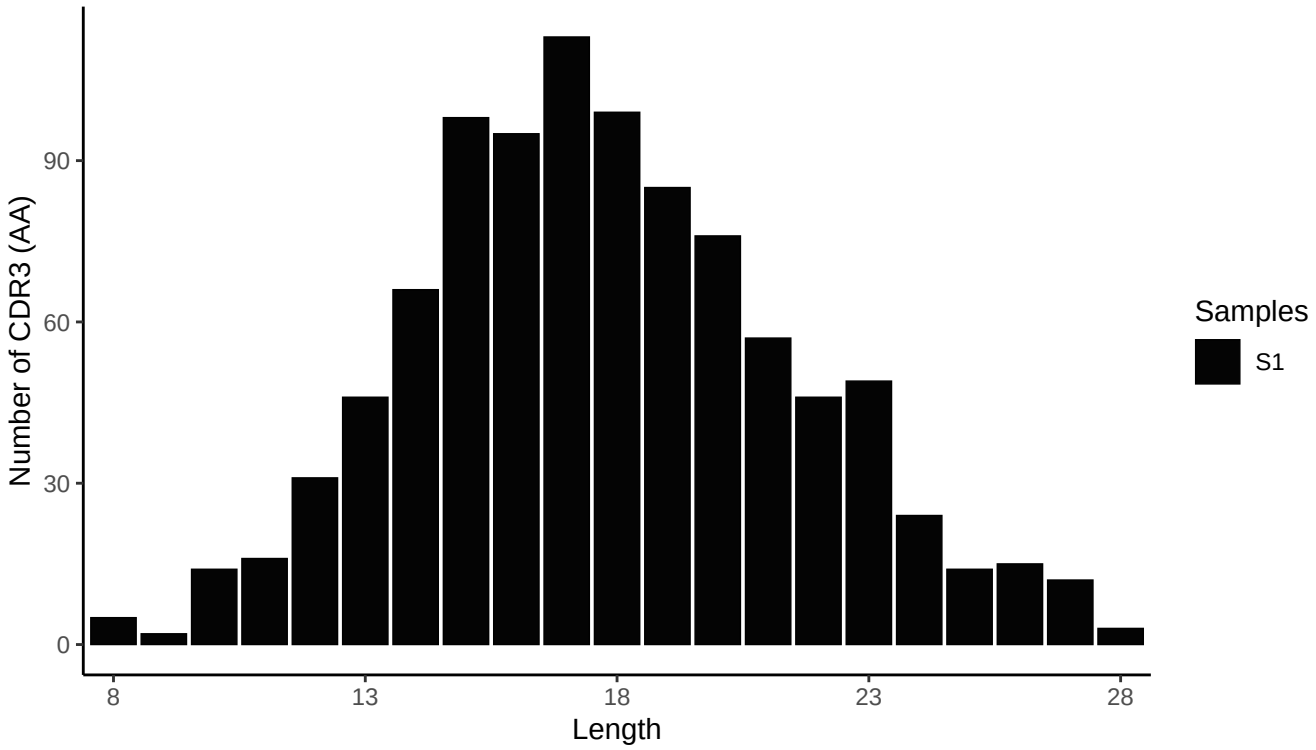
```
# ft4 <- bg(ft4, bg = "white", part = "all")
# save_as_image(x = ft4, path = "Frequencies_heavy_light_3_10x_table.png")
```

Clonal visualizations: Unique clones, CDR3, heavy/light chain compositions

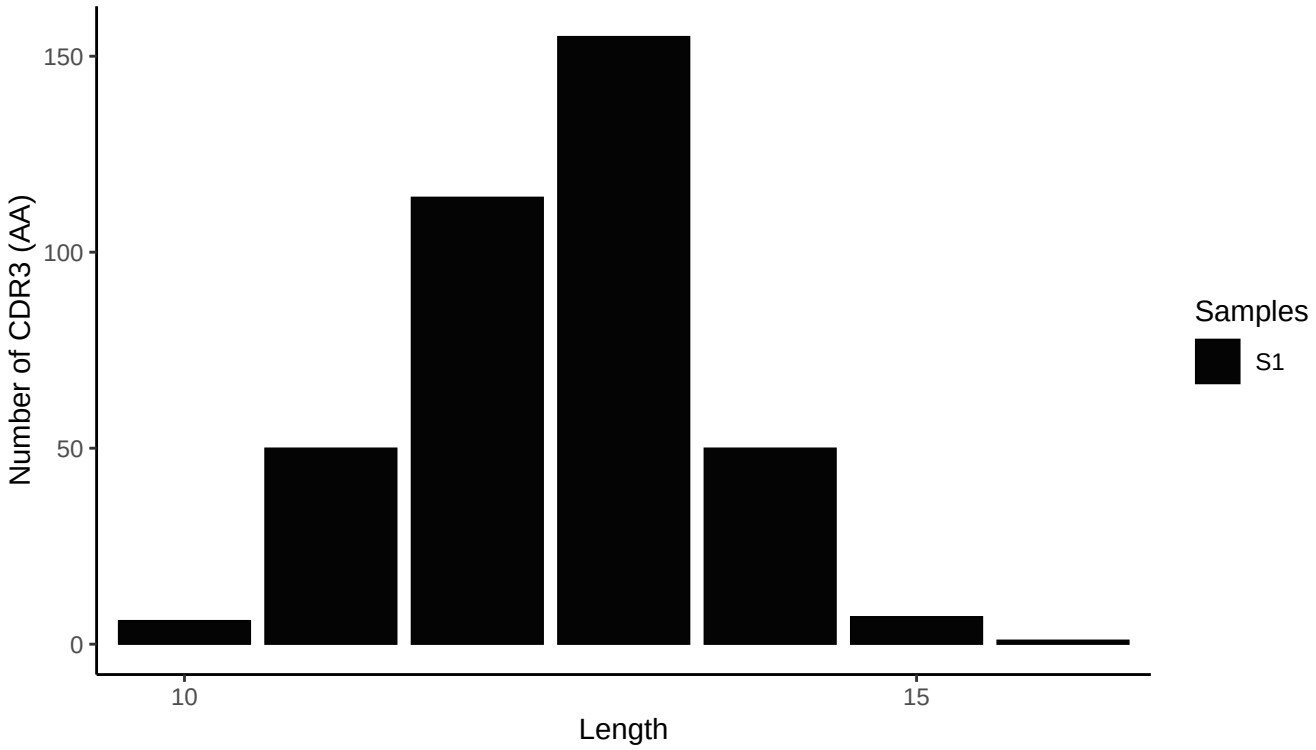
```
# Total number of unique clones
clonalQuant(S1_combBCR_allcontigs,
            cloneCall="strict",
            chain = "IGH",
            scale = TRUE) # Almost 100% of unique clones
```



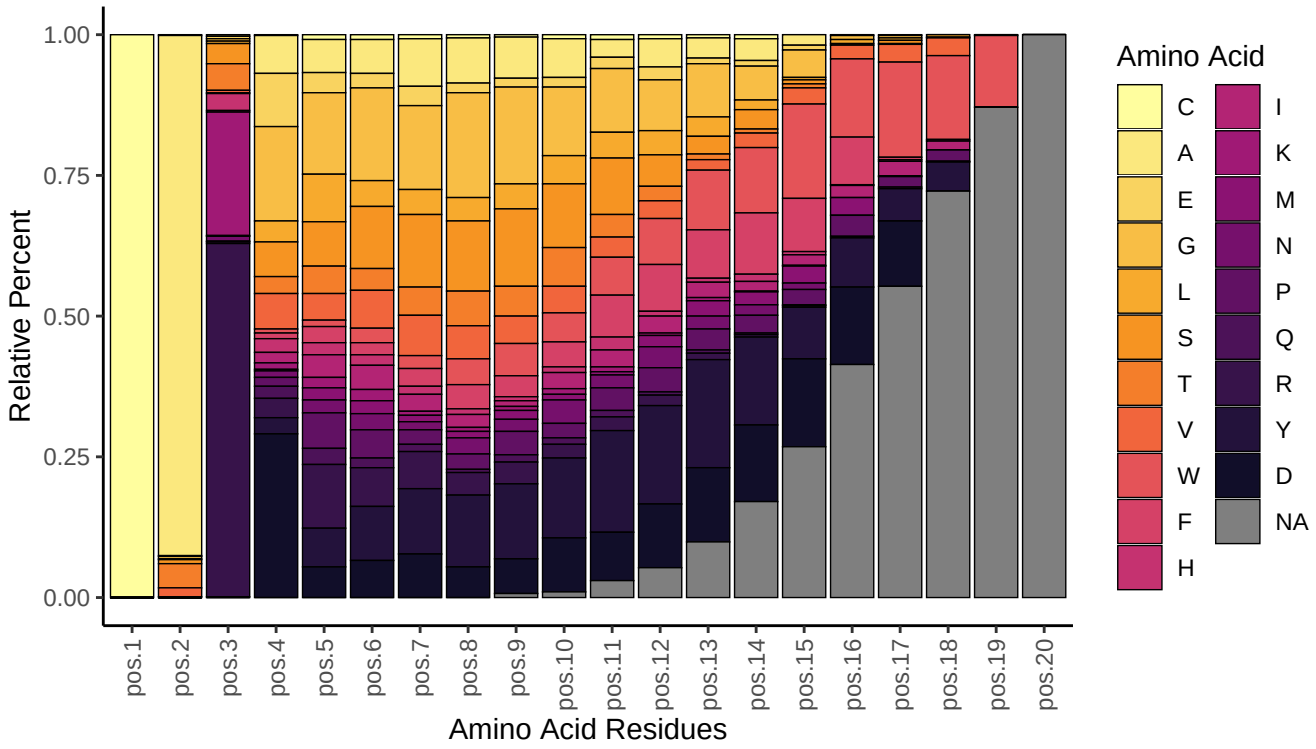

```
# Checking the length distribution of the CDR3 sequences
clonalLength(S1_combBCR_allcontigs,
             cloneCall="aa",
             chain = "IGH")
```



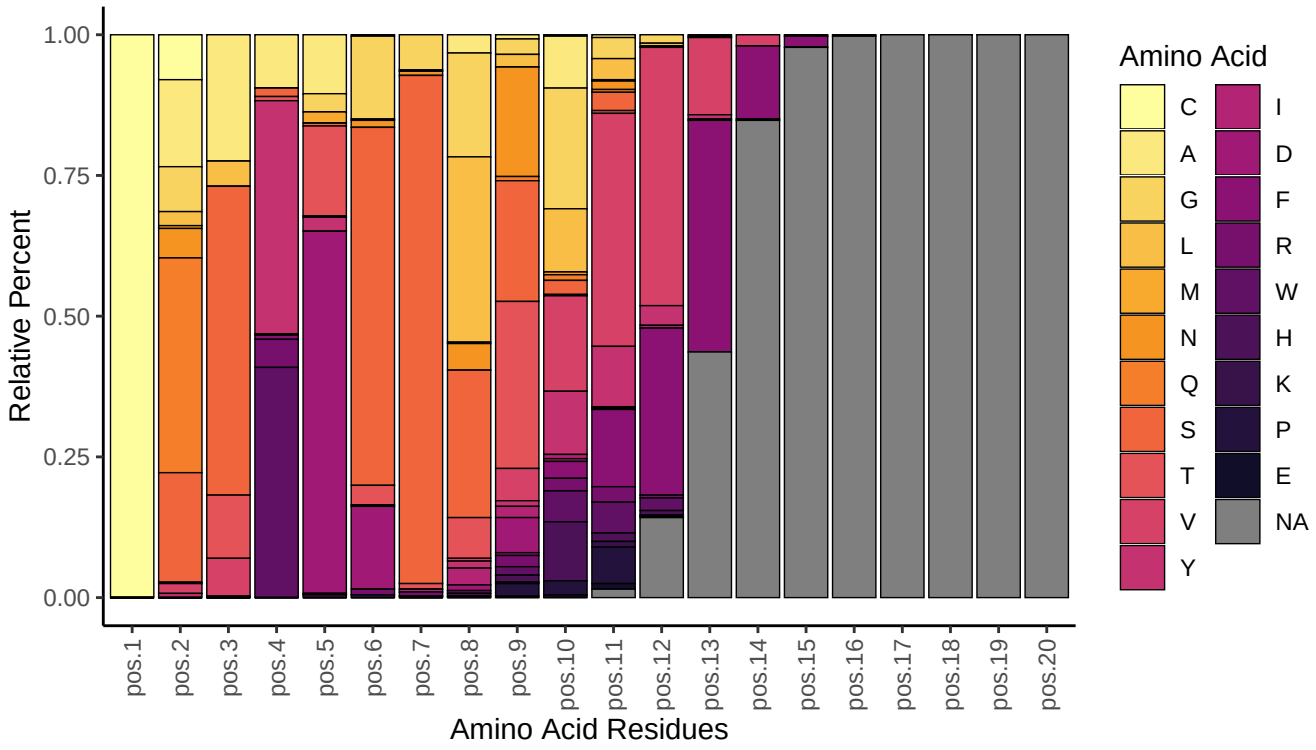

```
clonalLength(S1_combBCR_allcontigs,
             cloneCall="aa",
             chain = "IGL")
```



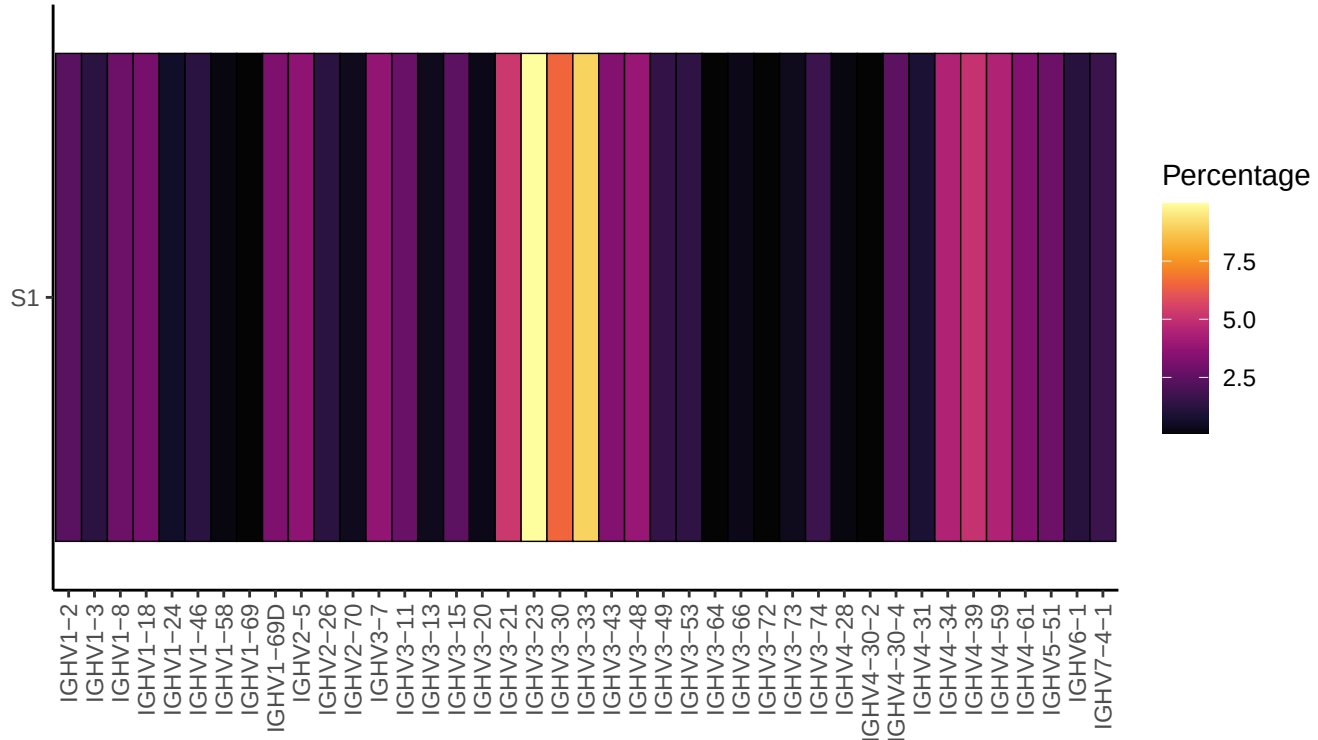
```
# Quantify the proportion of amino acids along the cdr3 sequence
percentAA(S1_combBCR_allcontigs,
          chain = "IGH",
          aa.length = 20)
```



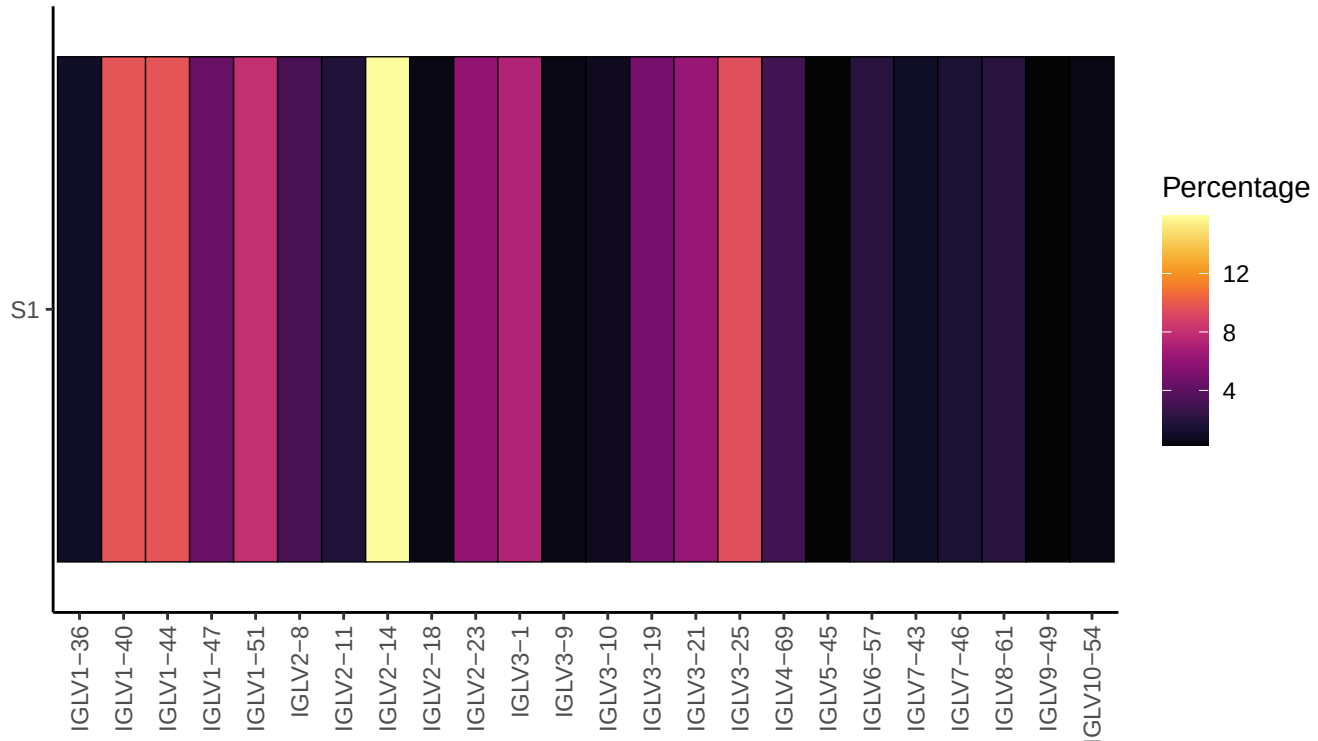

```
percentAA(S1_combBCR_allcontigs,
          chain = "IGL",
          aa.length = 20)
```



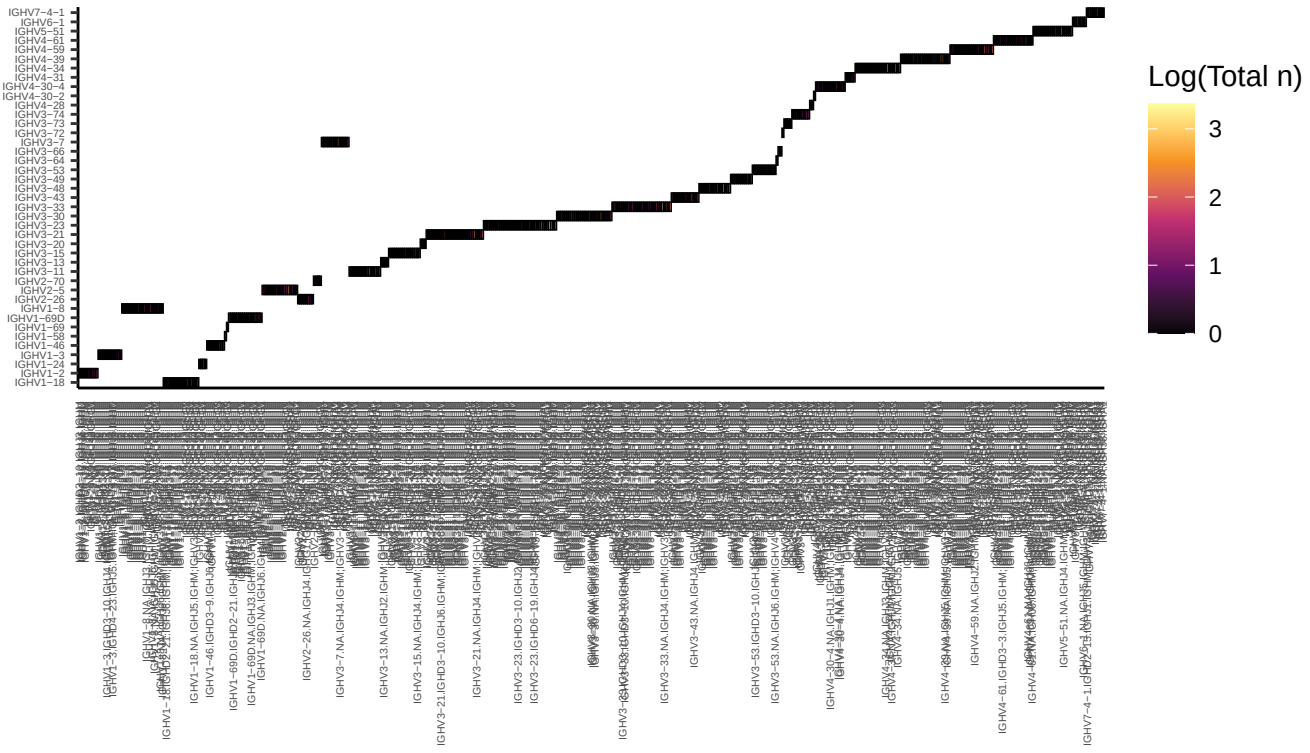

```
# Quantify the proportion of V or J gene usage
percentGenes(S1_combBCR_allcontigs,
             chain = "IGH",
             gene = "Vgene")
```




```
percentGenes(S1_combBCR_allcontigs,
             chain = "IGL",
             gene = "Vgene")
```



```
# visualization of the chain pairings
viz2 <- vizGenes(S1_combBCR_allcontigs,
         x.axis = "IGH",
         y.axis = "IGLV", # or "IGLV" or "IGKV" depending on the data
         plot = "heatmap",
         scale = FALSE)
print(viz2)
```




```
viz4 <- vizGenes(S1_combBCR_allcontigs,
         x.axis = "IGH",
         y.axis = "IGKJ",
         plot = "heatmap",
         scale = FALSE)
```

Most frequent B cell clones: pairing of heavy and light chains and cell frequencies

```
# This way of counting B cell clones includes those that do not have
# a chain and, thus, appear with NA, for example, those with IGH and IGL pairings like
# 1/0 and 0/1

# Count the occurrences of each IGH and IGLC pair
pair_counts_b <- bcr_data %>%
  group_by(IGH, IGLC) %>%
  summarize(count = n(), .groups = "drop") %>%
  arrange(desc(count))
```

```r
# Compare when NAs are removed
nrow(pair_counts_b) # 949
```

```
[1] 949
```

```r
pair_counts_nonas_b <- drop_na(pair_counts_b)
nrow(pair_counts_nonas_b) # 929
```

```
[1] 929
```

```r
# Calculate frequencies and percentages
total_counts_b <- sum(pair_counts_b$count) # 982
pair_counts_b <- pair_counts_b %>%
  mutate(frequency = count / total_counts_b,
                log2_frequency = log2(frequency),
                Percentage = (count / total_counts_b) * 100) # 949

nrow(pair_counts_b) # 949 are the rows, but the total B cell clones are 982
```

```
[1] 949
```

```r
# Function to count the number of heavy and light chains
count_heavy_light <- function(clone) {
  heavy_count <- str_count(clone, "IGHV")
  light_count <- str_count(clone, "IGKV") + str_count(clone, "IGLV")
  return(paste(heavy_count, light_count, sep = "/"))
}

# Function to extract and count the Ig constants
extract_ig_constant <- function(clone) {
  # Extract heavy chain constants based on specific delimiters
  heavy_constants <- str_extract_all(clone,
 ↪  "\\.IGHM_|\\.IGHD_|\\.IGHG[0-9]*_|\\.IGHA[0-9]*_|\\.IGHM;|\\.IGHD;|\\.IGHG[0-9]*;|\\.IGHA[0-9]*;")[[1]]
  light_constants_igk <- str_extract_all(clone, "IGKC")[[1]]
  light_constants_igl <- str_extract_all(clone, "IGLC[0-9]*")[[1]]

  # Clean up the constants to remove delimiters and IGH prefix
  heavy_constants <- gsub("[._;]", "", heavy_constants)
  heavy_constants <- gsub("IGH", "", heavy_constants)

  # Count the occurrences of each constant region
  heavy_counts <- table(heavy_constants)
  light_counts_igk <- table(light_constants_igk)
  light_counts_igl <- table(light_constants_igl)

  # Format the heavy chain constants
  format_heavy_constants <- function(counts) {
    formatted <- paste(ifelse(counts > 1, paste0(counts, names(counts)), names(counts)),
 ↪  collapse = "")
    formatted <- gsub("1", "", formatted) # Remove "1" to match the required format
```

```
    return(formatted)
  }

  # Format the light chain constants
  format_light_constants <- function(counts, type) {
    formatted <- paste(ifelse(counts > 1, paste0(counts, type), type), collapse = "")
    formatted <- gsub("1", "", formatted) # Remove "1" to match the required format
    return(formatted)
  }

  heavy_formatted <- format_heavy_constants(heavy_counts)
  light_formatted_igk <- format_light_constants(light_counts_igk, "K")
  light_formatted_igl <- format_light_constants(light_counts_igl, "L")

  # Combine the formatted light chains
  light_formatted <- paste(c(light_formatted_igk, light_formatted_igl), collapse = "")

  # Combine the formatted heavy and light chains
  return(paste(heavy_formatted, light_formatted, sep = " / "))
}

# Combine IGH and IGLC into a single column ("nonas")
pair_counts_b <- pair_counts_b %>%
  dplyr::mutate(B_cell_clones = paste(IGH, IGLC, sep = "_"))

# Calculate frequencies and percentages
total_counts_b <- sum(pair_counts_b$count) # 982
pair_counts_b <- pair_counts_b %>%
  dplyr::mutate(frequency = count / total_counts_b,
                log2_frequency = log2(frequency),
                Percentage = (count / total_counts_b) * 100) # 949

# Add the `Heavy/Light chains` and `Ig constant` columns
pair_counts_b <- pair_counts_b %>%
  dplyr::mutate(Heavy_Light_chains = sapply(B_cell_clones, count_heavy_light),
                `Ig constant` = sapply(B_cell_clones, extract_ig_constant)) %>%
  dplyr::select(`B cell clones` = B_cell_clones, `Heavy/Light chains` = Heavy_Light_chains, `Ig
 ↪  constant`, count, Percentage)

# pair_counts_b

# Write to CSV file
# write.csv(pair_counts_b, "Data_2024_Internship/Heavy_Light_Bcellclones_all_3_10x.csv",
 ↪  row.names = FALSE)

# Select the top N combinations for lollipop plot
N <- 10 # Change this value to any number you like
topN_pairs_b <- pair_counts_b %>%
  dplyr::slice(1:N)

# Create a flextable
ft6 <- flextable(topN_pairs_b)
```

```
# Format the table
ft6 <- ft6 %>%
  set_header_labels(`B cell clones` = "B cell clones",
                    `Heavy/Light chains` = "Heavy/Light chains",
                    `Ig constant` = "Ig constant\n region",
                    count = "Count",
                    Percentage = "Percentage") %>%
  add_header_lines(values = paste("Top", N, "most frequent B cell clones, Ig chains pairing, Ig
   ↪  constant regions and frequencies\n")) %>%
  set_table_properties(layout = "autofit") %>%
  theme_vanilla() %>%
  fontsize(size = 8, part = "all")

# Center align the text in header and body
ft6 <- align(ft6, align = "center", part = "header")
ft6 <- align(ft6, align = "center", part = "body")

# Display the flextable
ft6
```

Top 10 most frequent B cell clones, Ig chains pairing, Ig constant regions and frequencies

B cell clones	Heavy/Light chains	Ig constant region	Count	Percentage
IGHV3-23.NA.IGHJ4.IGHM_IGKV1D-39.IGKJ2.IGKC	1/1	M / K	4	0.407332
IGHV3-43.NA.IGHJ4.IGHM_IGLV2-14.IGLJ2.IGLC2	1/1	M / L	3	0.305499
IGHV1-8.IGHD4-17.IGHJ5.IGHM_IGKV1D-39.IGKJ2.IGKC	1/1	M / K	2	0.203666
IGHV2-5.IGHD2-2.IGHJ5.IGHM_IGLV1-44.IGLJ2.IGLC2	1/1	M / L	2	0.203666
IGHV2-5.NA.IGHJ4.IGHG1_IGLV1-44.IGLJ1.IGLC1	1/1	G / L	2	0.203666
IGHV3-23.NA.IGHJ4.IGHM_IGKV3-11.IGKJ4.IGKC	1/1	M / K	2	0.203666
IGHV3-23.NA.IGHJ4.IGHM_IGLV3-1.IGLJ2.IGLC2	1/1	M / L	2	0.203666
IGHV3-23.NA.IGHJ5.IGHM_IGKV3-11.IGKJ5.IGKC	1/1	M / K	2	0.203666
IGHV3-30.IGHD3-10.IGHJ6.IGHM_IGKV3-11.IGKJ4.IGKC	1/1	M / K	2	0.203666
IGHV3-30.NA.IGHJ4.IGHM_IGLV3-25.IGLJ3.IGLC2	1/1	M / L	2	0.203666

```
# simple_table4 <- bg(ft6, bg = "white", part = "all")
# save_as_image(x = simple_table4, path =
 ↪  "Data_2024_Internship/Top_H_L_pairs_constant_regions_table_all_3_10x.png")
```

After finishing the V(D)J analysis, the following "Part II" will continue with the analysis of the gene-expression library.