VDJ_assignment_scRepertoire_UNPRODUCTIVE_NEW

Javier Molina Valenzuela

2024-10-10

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

```
library(Seurat)
```

```
## Loading required package: SeuratObject
```

```
## Loading required package: sp
```

```
##
## Attaching package: 'SeuratObject'
```

```
## The following objects are masked from 'package:base':
##
##     intersect, t
```

```
library(ggplot2)
library(SingleR)
```

```
## Loading required package: SummarizedExperiment
```

```
## Loading required package: MatrixGenerics
```

```
## Loading required package: matrixStats
```

```
##
## Attaching package: 'MatrixGenerics'
```

```
## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars
```

```
## Loading required package: GenomicRanges
```

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

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

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following object is masked from 'package:SeuratObject':
##
##     intersect
```

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

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, 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.max, which.min
```

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

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

```
## The following object is masked from 'package:utils':
##
##     findMatches
```

```
## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname
```

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

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

```
## The following object is masked from 'package:sp':
##
##     %over%
```

```
## Loading required package: GenomeInfoDb
```

```
## Loading required package: Biobase
```

```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
##
## Attaching package: 'Biobase'
```

```
## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians
```

```
## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians
```

```
##
## Attaching package: 'SummarizedExperiment'
```

```
## The following object is masked from 'package:Seurat':
##
##     Assays
```

```
## The following object is masked from 'package:SeuratObject':
##
##     Assays
```

```
library(dplyr)
```

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

```
## The following object is masked from 'package:Biobase':
##
##     combine
```

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

```
## The following object is masked from 'package:GenomeInfoDb':
##
##     intersect
```

```
## The following objects are masked from 'package:IRanges':
##
##     collapse, desc, intersect, setdiff, slice, union
```

```
## The following objects are masked from 'package:S4Vectors':
##
##     first, intersect, rename, setdiff, setequal, union
```

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

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

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

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

```
library(celldex)
```

```
##
## Attaching package: 'celldex'
```

```
## The following objects are masked from 'package:SingleR':
##
##     BlueprintEncodeData, DatabaseImmuneCellExpressionData,
##     HumanPrimaryCellAtlasData, ImmGenData, MonacoImmuneData,
##     MouseRNAseqData, NovershternHematopoieticData
```

```
library(RColorBrewer)
library(SingleCellExperiment)
library(DropletUtils)
library(Matrix)
```

```
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:S4Vectors':
##
##     expand
```

```
library(knitr)
library(reshape2)
library(dittoSeq)
library(viridis)
```

```
## Loading required package: viridisLite
```

```
library(RColorBrewer)
library(wesanderson)
library(DoubletFinder)
library(SeuratDisk)
```

```
## Registered S3 method overwritten by 'SeuratDisk':
##   method            from
##   as.sparse.H5Group Seurat
```

```
library(httr)
```

```
##
## Attaching package: 'httr'
```

```
## The following object is masked from 'package:Biobase':
##
##     content
```

```
library(scRepertoire)
library(ggalluvial)
library(circlize)
```

```
## ========================================
## circlize version 0.4.16
## CRAN page: https://cran.r-project.org/package=circlize
## Github page: https://github.com/jokergoo/circlize
## Documentation: https://jokergoo.github.io/circlize_book/book/
```

```
##
## If you use it in published research, please cite:
## Gu, Z. circlize implements and enhances circular visualization
##   in R. Bioinformatics 2014.
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(circlize))
## ========================================
```

```
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:viridis':
##
##     viridis_pal
```

```
library(jsonlite)
library(Biostrings)
```

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

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

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

```
library(ggseqlogo)
```

Read Seurat information

```
Seurat_BM <- readRDS("../../3.Seurat/Results/BM_final_Seurat_TFM_simp.RDS")
Seurat_SP <- readRDS("../../3.Seurat/Results/SP_final_Seurat_TFM_simp.RDS")
```

Loading Data from multiplexed experiment

We need the filtered_contig_annotations.csv where each BCR contigs are stored and a Seurat object that will provide the information to determine which barcode belongs to each sample.

```
contigs_BM <- read.csv("../Data/all_contig_annotations_BM_TFM.csv")
contig_list_BM <- createHTOContigList(contigs_BM, Seurat_BM, group.by = "mice")

contigs_SP <- read.csv("../Data/all_contig_annotations_SP_TFM.csv")
contig_list_SP <- createHTOContigList(contigs_SP, Seurat_SP, group.by = "mice")
```

Check the data For each sample we have a tabular structure that contains the barcode

```
#head(contig_list_BM)
#lapply(contig_list_BM, dim)
```

We check that the A12 VDJ is present in the VDJ repertoire of A12 mice.

We confirmed that there are more barcodes with IghvA12 in BM than in SP (there were also more cells), and that there are barely any IghvA12 in the WT.

It seems that our modified VDJ reference has worked.

```
# In A12 mice
print("A12 mice")
```

```
## [1] "A12 mice"
```

```
length(which(contig_list_BM$A12_1$v_gene == "IGHV5-6(A12)"))
```

```
## [1] 299
```

```
length(which(contig_list_BM$A12_2$v_gene == "IGHV5-6(A12)"))
```

```
## [1] 333
```

```
length(which(contig_list_BM$A12_3$v_gene == "IGHV5-6(A12)"))
```

```
## [1] 327
```

```
length(which(contig_list_SP$A12_1$v_gene == "IGHV5-6(A12)"))
```

```
## [1] 90
```

```
length(which(contig_list_SP$A12_2$v_gene == "IGHV5-6(A12)"))
```

```
## [1] 97
```

```
length(which(contig_list_SP$A12_3$v_gene == "IGHV5-6(A12)"))
```

```
## [1] 65
```

```
# In WT mice
print("WT mice")
```

```
## [1] "WT mice"
```

```
length(which(contig_list_BM$WT_1$v_gene == "IGHV5-6(A12)"))
```

```
## [1] 18
```

```
length(which(contig_list_BM$WT_2$v_gene == "IGHV5-6(A12)"))
```

```
## [1] 20
```

```
length(which(contig_list_BM$WT_3$v_gene == "IGHV5-6(A12)"))
```

```
## [1] 24
```

```
length(which(contig_list_SP$WT_1$v_gene == "IGHV5-6(A12)"))
```

```
## [1] 9
```

```
length(which(contig_list_SP$WT_2$v_gene == "IGHV5-6(A12)"))
```

```
## [1] 11
```

```
length(which(contig_list_SP$WT_3$v_gene == "IGHV5-6(A12)"))
```

```
## [1] 14
```

```
contig_list_BM_IghA12 <- lapply(contig_list_BM, function(x) subset(x, v_gene == "IGHV5-6(A12)"))
```

Combining contigs into clones

1. Each barcode can only have a maximum of 2 sequences, if greater exists, the 2 with the highest reads are selected; 2) The strict definition of a clone is based on the normalized Levenshtein edit distance of CDR3 nucleotide sequences and V-gene usage.

Citation: https://pubmed.ncbi.nlm.nih.gov/34161770/

This definition allows for the grouping of BCRs derived from the same progenitor that have undergone mutation as part of somatic hypermutation and affinity maturation.

\[\text{threshold}(s, t) = 1 - \frac{\text{Levenshtein}(s, t)}{\frac{\text{length}(s) + \text{length}(t)}{2}} \]

Change Sample order

```
names(contig_list_BM)
```

```
## [1] "A12_2" "WT_2"  "WT_1"  "A12_1" "A12_3" "WT_3"
```

```
new_order_BM <- c(3, 2, 6, 4, 1, 5)
contig_list_BM <- contig_list_BM[new_order_BM]

names(contig_list_SP)
```

```
## [1] "WT_2"  "WT_3"  "WT_1"  "A12_3" "A12_2" "A12_1"
```

```
new_order_SP <- c(3, 2, 1, 6, 5, 4)
contig_list_SP <- contig_list_SP[new_order_SP]
```

```
combined_BCR_BM <- combineBCR(
                      contig_list_BM,
                      samples = names(contig_list_BM),
                      ID = NULL,
                      threshold = 0.85,
                      removeNA = FALSE,
                      removeMulti = FALSE,
                      filterMulti = FALSE,
                      filterNonproductive = FALSE
                    )
```

```
combined_BCR_SP <- combineBCR(
                      contig_list_SP,
                      samples = names(contig_list_SP),
                      ID = NULL,
                      threshold = 0.85,
                      removeNA = FALSE,
                      removeMulti = FALSE,
                      filterMulti = FALSE,
                      filterNonproductive = FALSE
                    )
```

Number of identified cells per mouse

```
lapply(contig_list_BM, function(x) length(x$productive[x$productive == "false"]))
```

```
## $WT_1
## [1] 2884
##
## $WT_2
## [1] 3241
##
## $WT_3
## [1] 3629
##
## $A12_1
## [1] 3732
##
## $A12_2
## [1] 3890
##
## $A12_3
## [1] 3691
```

```
lapply(contig_list_SP, function(x) length(x$productive[x$productive == "false"]))
```

```
## $WT_1
## [1] 882
##
## $WT_3
## [1] 846
##
## $WT_2
## [1] 930
##
## $A12_1
## [1] 966
##
## $A12_2
## [1] 1031
##
## $A12_3
## [1] 963
```

Add A12 information in the combined_BCR dataframe

Genotype information

```
# BM

combined_BCR_BM <- lapply(combined_BCR_BM, function(df) {
  df$genotype <- ifelse(grepl("A12", df$sample), "A12", "WT")
  return(df)
})

#SP
combined_BCR_SP <- lapply(combined_BCR_SP, function(df) {
  df$genotype <- ifelse(grepl("A12", df$sample), "A12", "WT")
  return(df)
})
```

Mark cells with heavy chain A12

```
# BM
```

```
combined_BCR_BM <- lapply(combined_BCR_BM, function(df) {
  df$A12_Igh_expressing <- grepl("IGHV5-6\\(A12\\)\\.IGHDA12\\.IGHJ1\\(A12\\)", df$IGH)
  return(df)
})

#SP
combined_BCR_SP <- lapply(combined_BCR_SP, function(df) {
  df$A12_Igh_expressing <- grepl("IGHV5-6\\(A12\\)\\.IGHDA12\\.IGHJ1\\(A12\\)", df$IGH)
  return(df)
})
```

Calculate length CDR3 Igh

```
# BM
combined_BCR_BM <- lapply(combined_BCR_BM, function(df) {
  df$len_aa_CDR3 <- nchar(df$cdr3_aa1)
  return(df)
})

#SP
combined_BCR_SP <- lapply(combined_BCR_SP, function(df) {
  df$len_aa_CDR3 <- nchar(df$cdr3_aa1)
  return(df)
})
```

Subset of IghA12 expressing cells

```
combined_BCR_BM_A12_IghA12 <- lapply(combined_BCR_BM, function(x) subset(x, A12_Igh_expressing == "TRUE" & genotyp
e == "A12"))
combined_BCR_SP_A12_IghA12 <- lapply(combined_BCR_SP, function(x) subset(x, A12_Igh_expressing == "TRUE" & genotyp
e == "A12"))
combined_BCR_BM_A12 <- lapply(combined_BCR_BM, function(x) subset(x, genotype == "A12"))
combined_BCR_SP_A12 <- lapply(combined_BCR_SP, function(x) subset(x, genotype == "A12"))
```

Number of B cells with multiple Igl or Igh

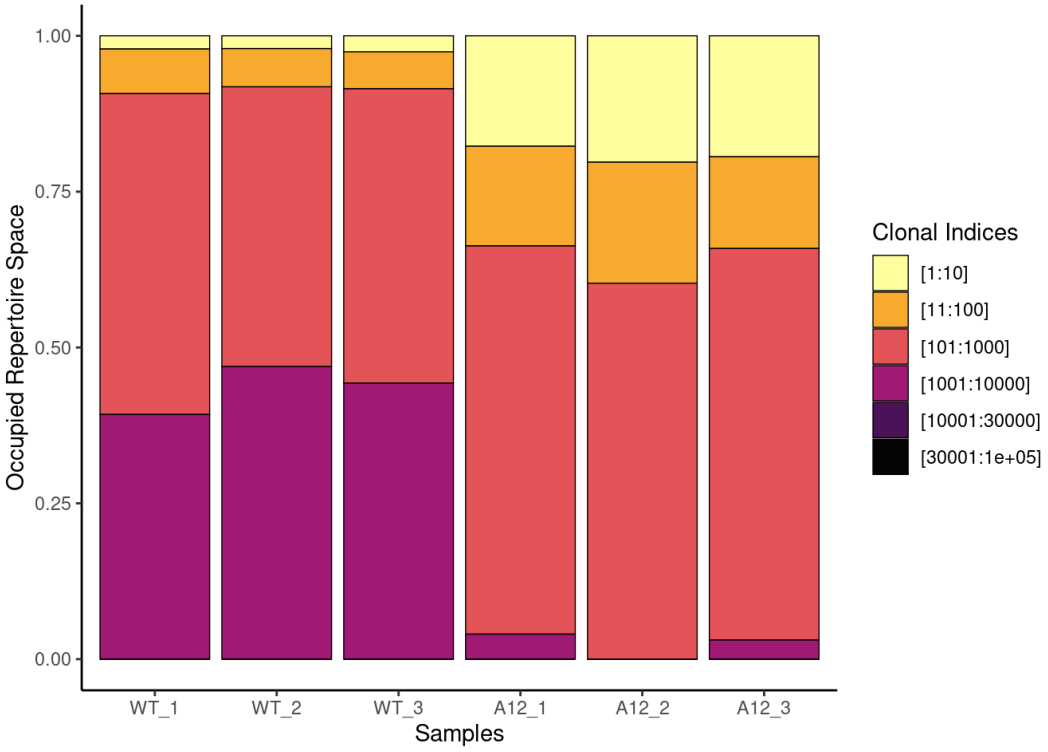
Export clones

```
df_combined_BM <- do.call(rbind, combined_BCR_BM)
write.csv(df_combined_BM, file = "../Results/combined_bcr_data_BM.csv", row.names = FALSE)

df_combined_SP <- do.call(rbind, combined_BCR_SP)
write.csv(df_combined_SP, file = "../Results/combined_bcr_data_SP.csv", row.names = FALSE)
```

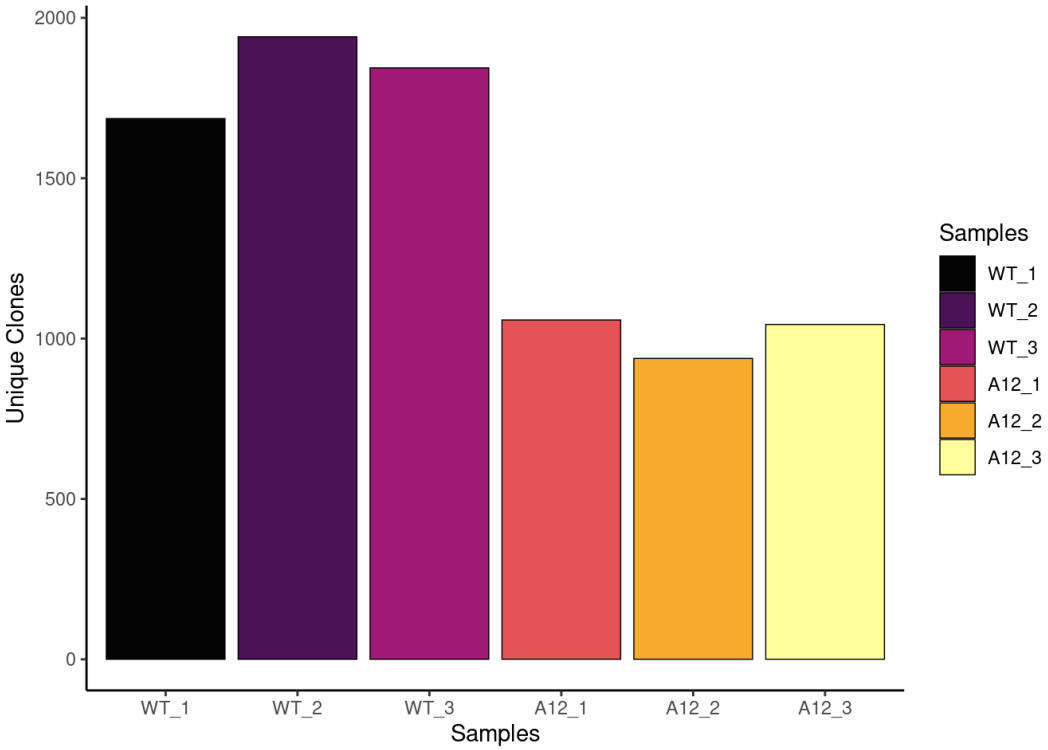
Clonal visualization

```
# Definir el orden deseado para los niveles de la columna 'Cluster'
cluster_levels <- c("WT", "A12")
```

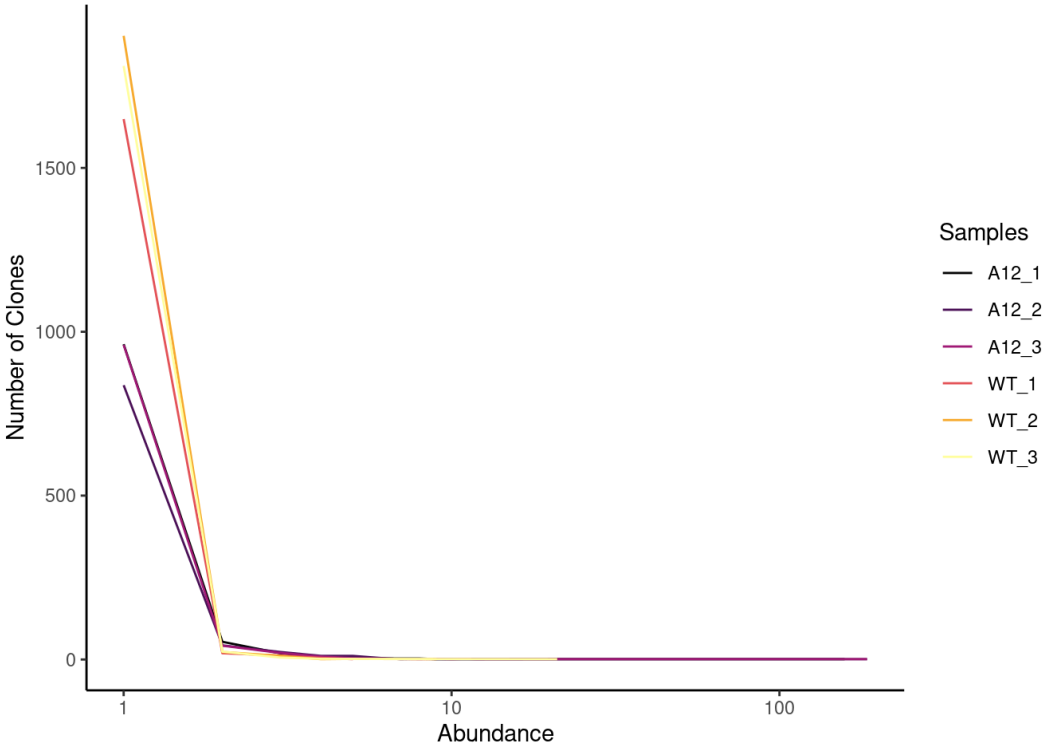
```
clonalProportion(combined_BCR_BM,
  cloneCall = "strict",
  chain = "both",
  group.by = NULL,
  exportTable = FALSE)
```



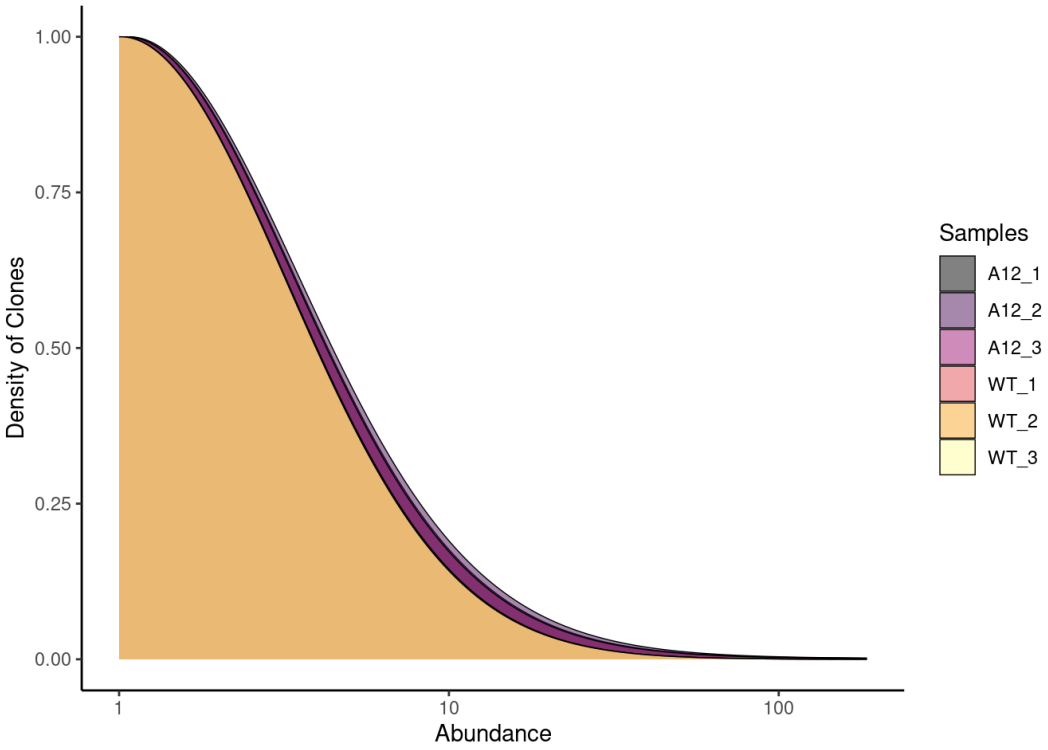

```
clonalQuant(combined_BCR_BM,
  cloneCall = "strict",
  chain = "both",
  group.by = NULL,
  exportTable = FALSE)
```



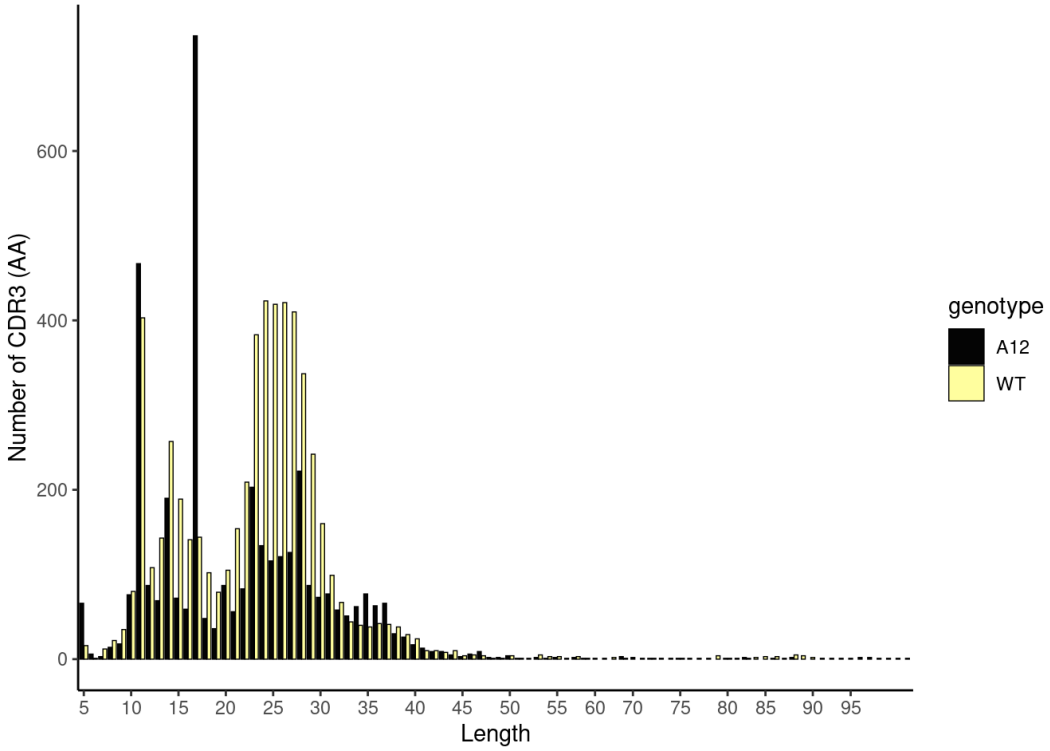

```
clonalAbundance(combined_BCR_BM,
  cloneCall = "strict",
  chain = "both",
  scale = FALSE)
```



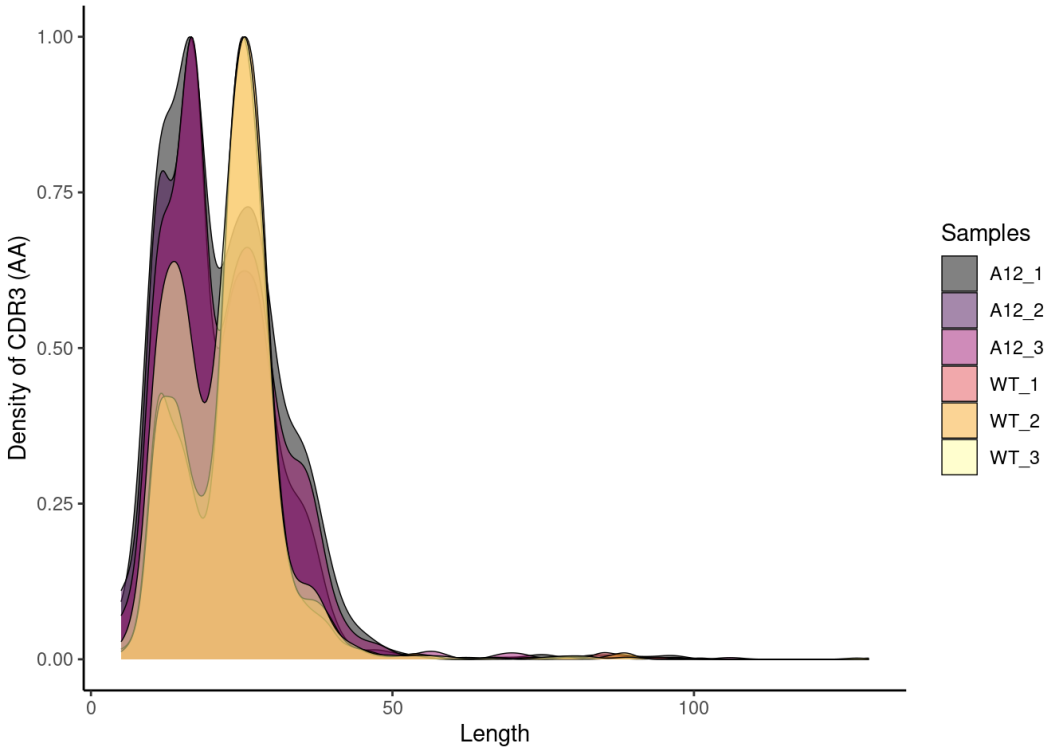

```
clonalAbundance(combined_BCR_BM,
  cloneCall = "strict",
  chain = "both",
  scale = TRUE)
```



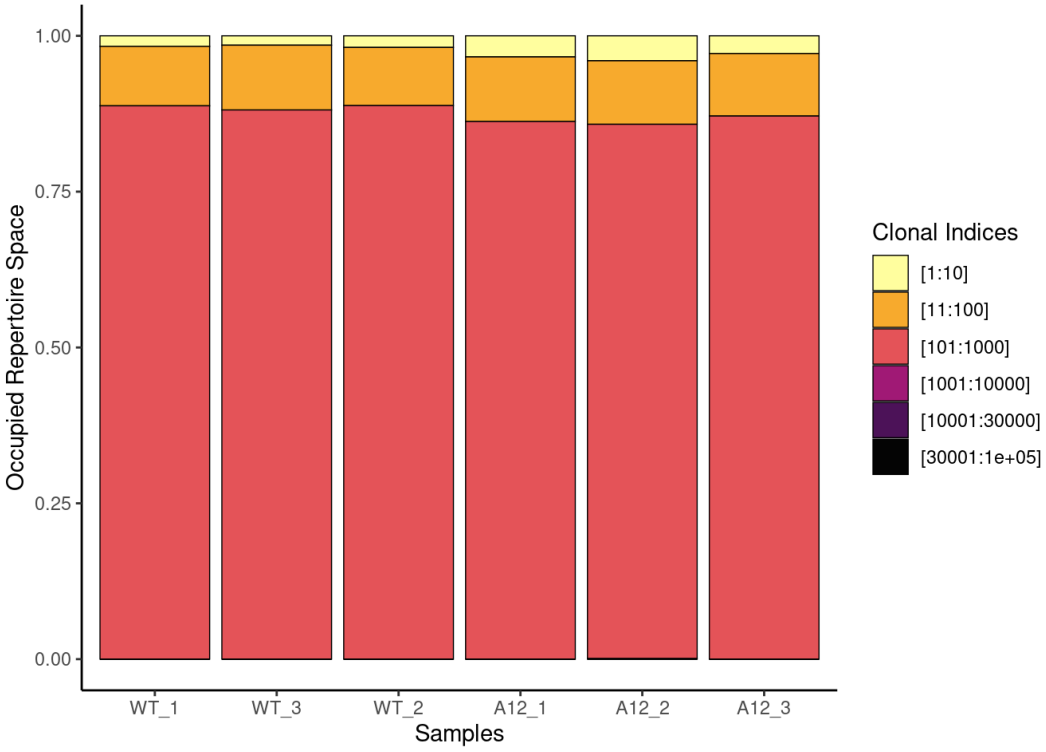

```
clonalLength(combined_BCR_BM,
  cloneCall = "aa",
  chain = "both",
  scale = FALSE,
  group.by = "genotype")
```



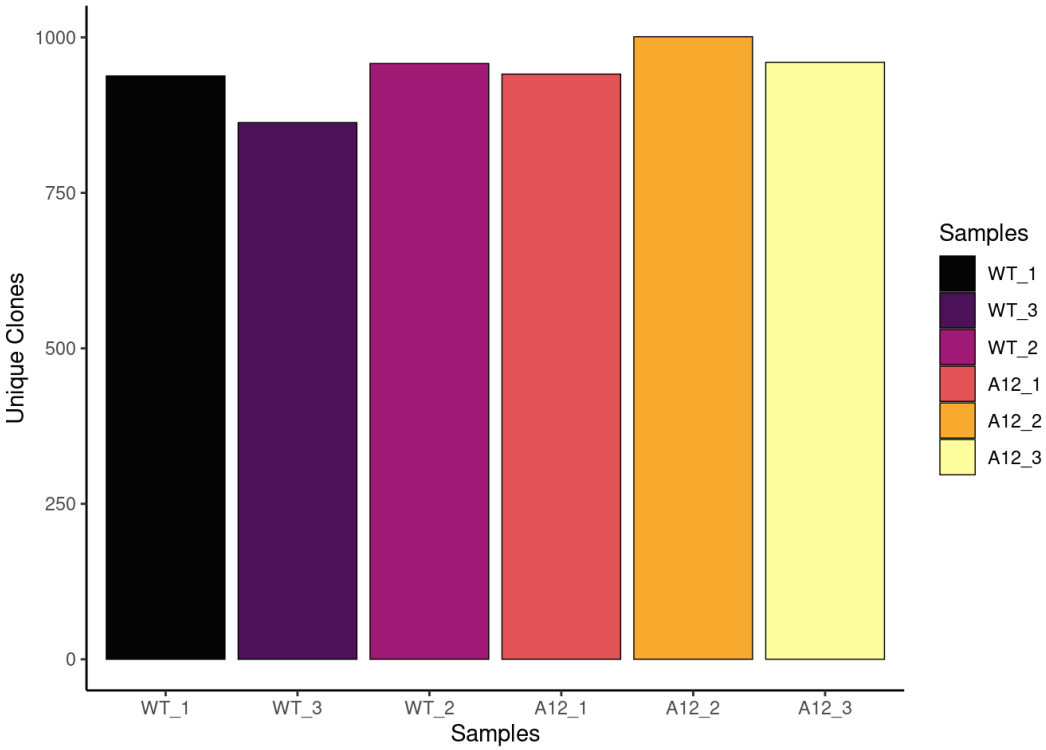

```
clonalLength(combined_BCR_BM,
  cloneCall = "aa",
  chain = "both",
  scale = TRUE)
```



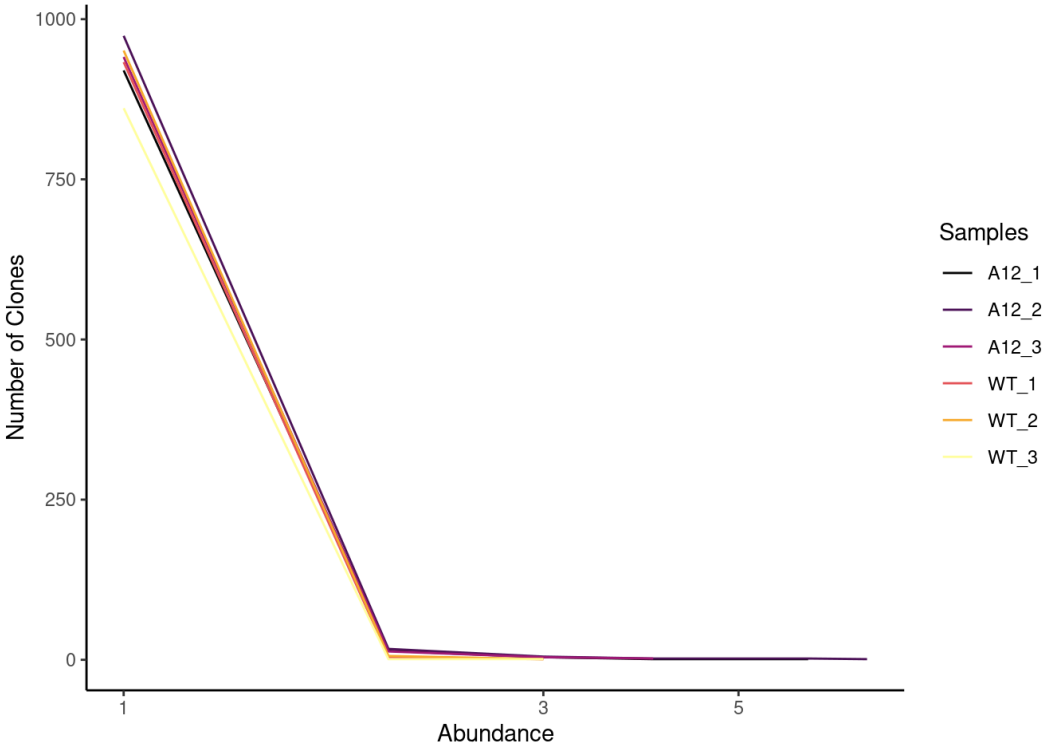

```
clonalProportion(combined_BCR_SP,
  cloneCall = "strict",
  chain = "both",
  group.by = NULL,
  exportTable = FALSE)
```



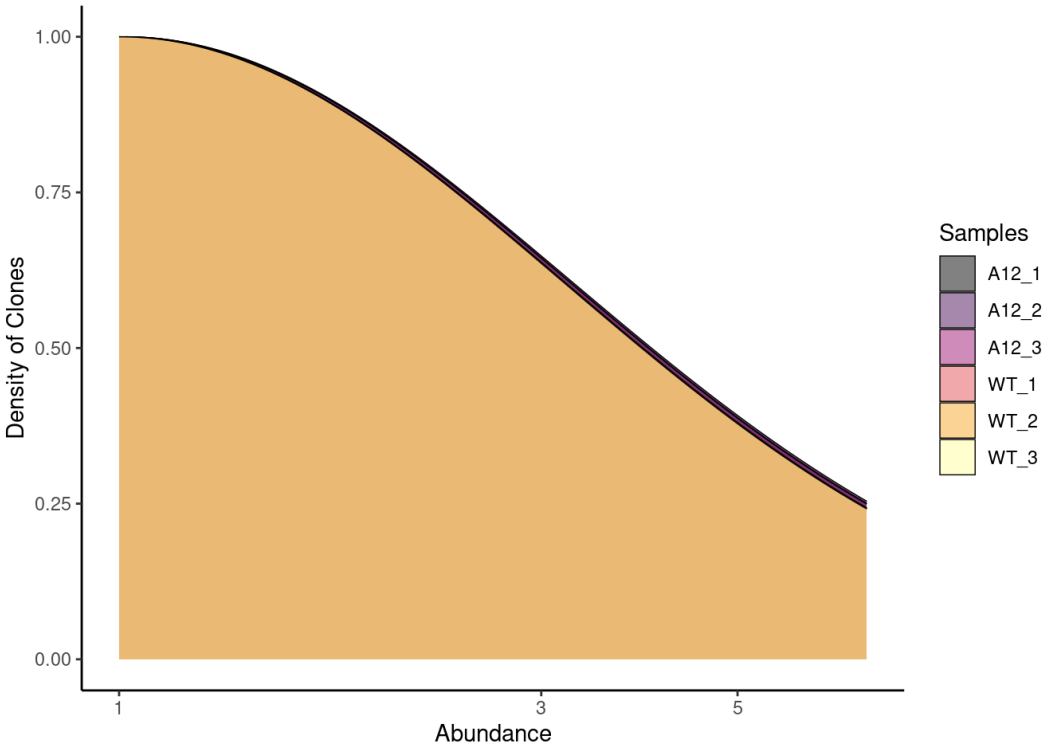

```
clonalQuant(combined_BCR_SP,
  cloneCall = "strict",
  chain = "both",
  group.by = NULL,
  exportTable = FALSE)
```



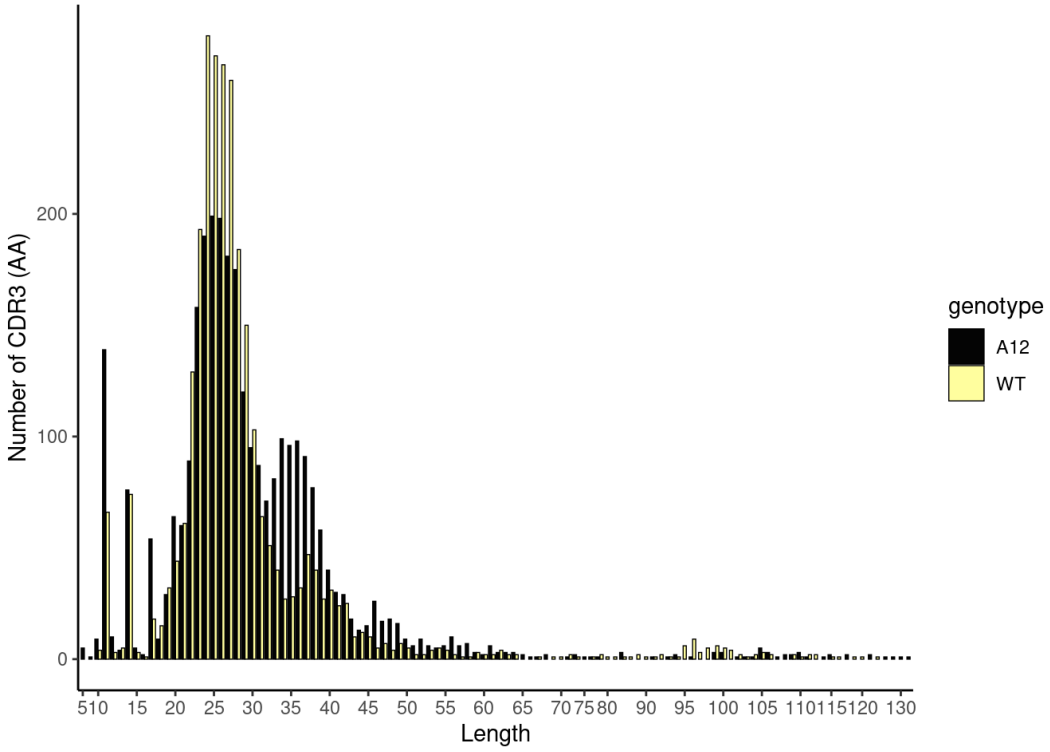

```
clonalAbundance(combined_BCR_SP,
  cloneCall = "strict",
  chain = "both",
  scale = FALSE)
```



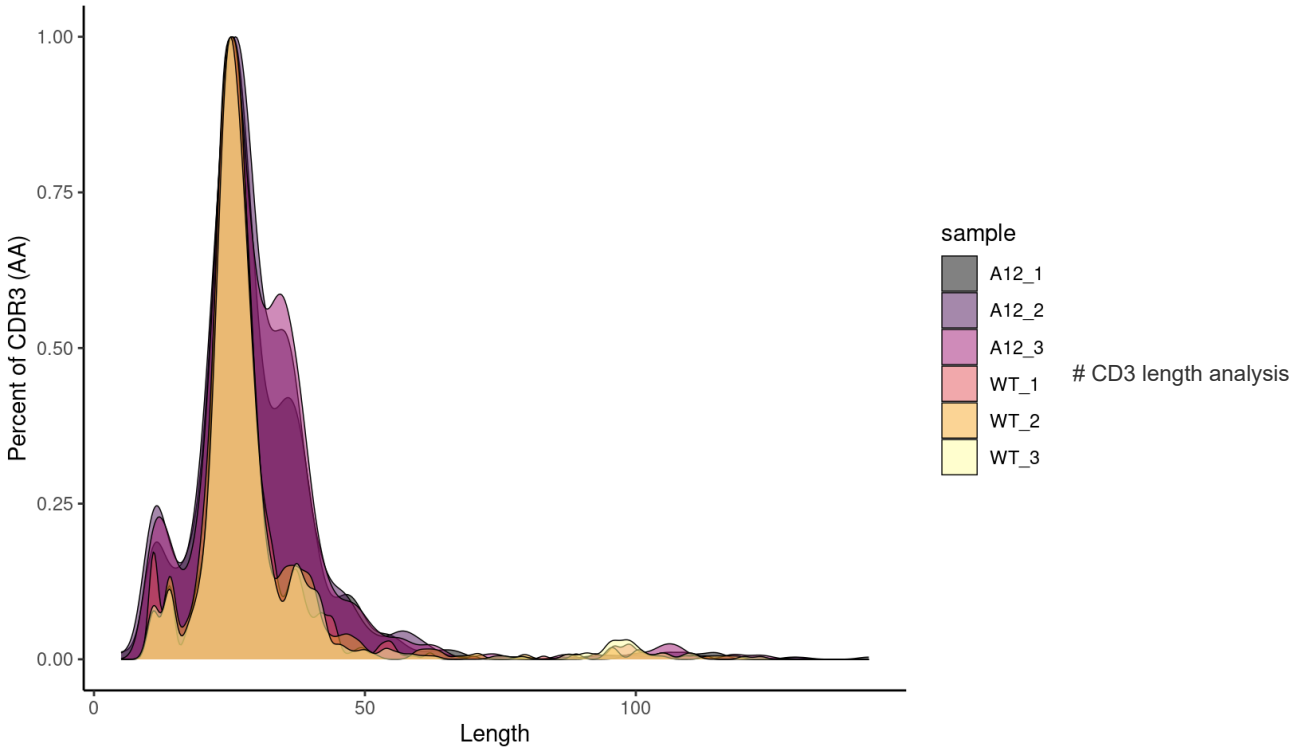

```
clonalAbundance(combined_BCR_SP,
  cloneCall = "strict",
  chain = "both",
  scale = TRUE)
```




```
clonalLength(combined_BCR_SP,
  cloneCall = "aa",
  chain = "both",
  scale = FALSE,
  group.by = "genotype",
  palette = "inferno")
```




```
clonalLength(combined_BCR_SP,
  cloneCall = "aa",
  chain = "both",
  scale = TRUE,
  group.by = "sample")
```



We identified different lengths in CDR3. There is a clear difference between A12 mice and WT mice. In spleen we observe that the V region, CDR3 of the Igh reduces drastically (tolerance). However, JH1(A12) region mantains with same huge proportion in spleen. What CDR3 length do this JH1 have?

Add a column with JH usage information

```
add_IGHJ_columns <- function(df) {
  for (i in 1:4) {
    column_name <- paste0("VHJ", i)
    pattern <- paste0("IGHJ", i)
    df[[column_name]] <- grepl(pattern, df$IGH)
  }
  return(df)
}

combined_BCR_BM <- lapply(combined_BCR_BM, add_IGHJ_columns)
combined_BCR_SP <- lapply(combined_BCR_SP, add_IGHJ_columns)
combined_BCR_BM_A12 <- lapply(combined_BCR_BM_A12, add_IGHJ_columns)
combined_BCR_SP_A12 <- lapply(combined_BCR_SP_A12, add_IGHJ_columns)
```

Mark cells with two heavy chains

```
# BM

combined_BCR_BM <- lapply(combined_BCR_BM, function(df) {
  df$two_Igh <- grepl(";", df$IGH) &
  !grepl("(^NA;|;NA$|;NA;)", df$cdr3_aa2)
  return(df)
})

#SP
combined_BCR_SP <- lapply(combined_BCR_SP, function(df) {
  df$two_Igh <- grepl(";", df$IGH) &
  !grepl("(^NA;|;NA$|;NA;)", df$cdr3_aa2)
  return(df)
})

lapply(combined_BCR_BM, function(x) length(x$two_Igh[x$two_Igh == "TRUE"]))
```

```
## $WT_1
## [1] 91
##
## $WT_2
## [1] 108
##
## $WT_3
## [1] 109
##
## $A12_1
## [1] 45
##
## $A12_2
## [1] 42
##
## $A12_3
## [1] 39
```

```
lapply(combined_BCR_SP, function(x) length(x$two_Igh[x$two_Igh == "TRUE"]))
```

```
## $WT_1
## [1] 167
##
## $WT_3
## [1] 147
##
## $WT_2
## [1] 155
##
## $A12_1
## [1] 116
```

```
##
## $A12_2
## [1] 127
##
## $A12_3
## [1] 138
```

BM IGH

```
combined_BCR_BM_IGH <- lapply(combined_BCR_BM, function(x) {
  subset(x, !is.na(x$cdr3_aa1) & x$two_Igh == FALSE)
})

combined_BCR_SP_IGH <- lapply(combined_BCR_SP, function(x) {
  subset(x, !is.na(x$cdr3_aa1) & x$two_Igh == FALSE)
})


combined_BCR_BM_MOUSE_MIX <- do.call(rbind, combined_BCR_BM)
combined_BCR_SP_MOUSE_MIX <- do.call(rbind, combined_BCR_SP)
combined_BCR_BM_A12_MOUSE_MIX <- do.call(rbind, combined_BCR_BM[c("A12_1", "A12_2", "A12_3")])
combined_BCR_SP_A12_MOUSE_MIX <- do.call(rbind, combined_BCR_SP[c("A12_1", "A12_2", "A12_3")])
combined_BCR_BM_WT_MOUSE_MIX <- do.call(rbind, combined_BCR_BM[c("WT_1", "WT_2", "WT_3")])
combined_BCR_SP_WT_MOUSE_MIX <- do.call(rbind, combined_BCR_SP[c("WT_1", "WT_2", "WT_3")])
```

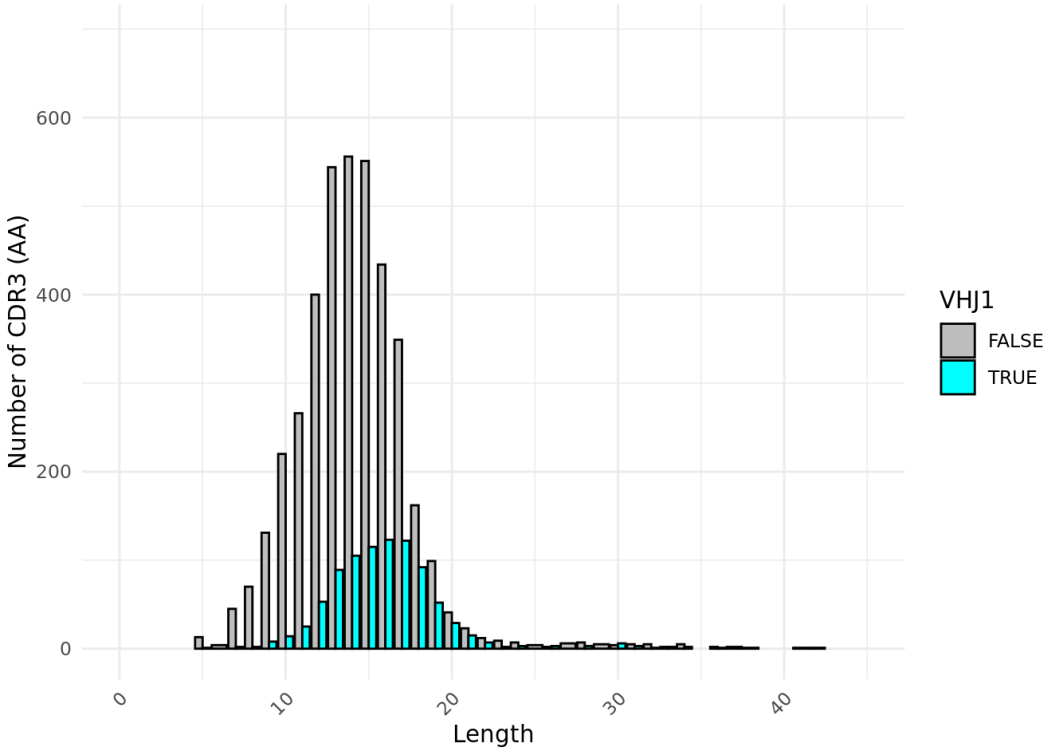
CDR3 length and JH1, JH2, JH3 and JH4 plots

```
df_plot <- combined_BCR_BM_WT_MOUSE_MIX %>%
  mutate(VHJ1_group = ifelse(grepl("IGHJ1", IGH), "TRUE", "FALSE")) %>%
  group_by(len_aa_CDR3, VHJ1_group) %>%
  summarise(cell_count = n())
```

```
## `summarise()` has grouped output by 'len_aa_CDR3'. You can override using the
## `.groups` argument.
```

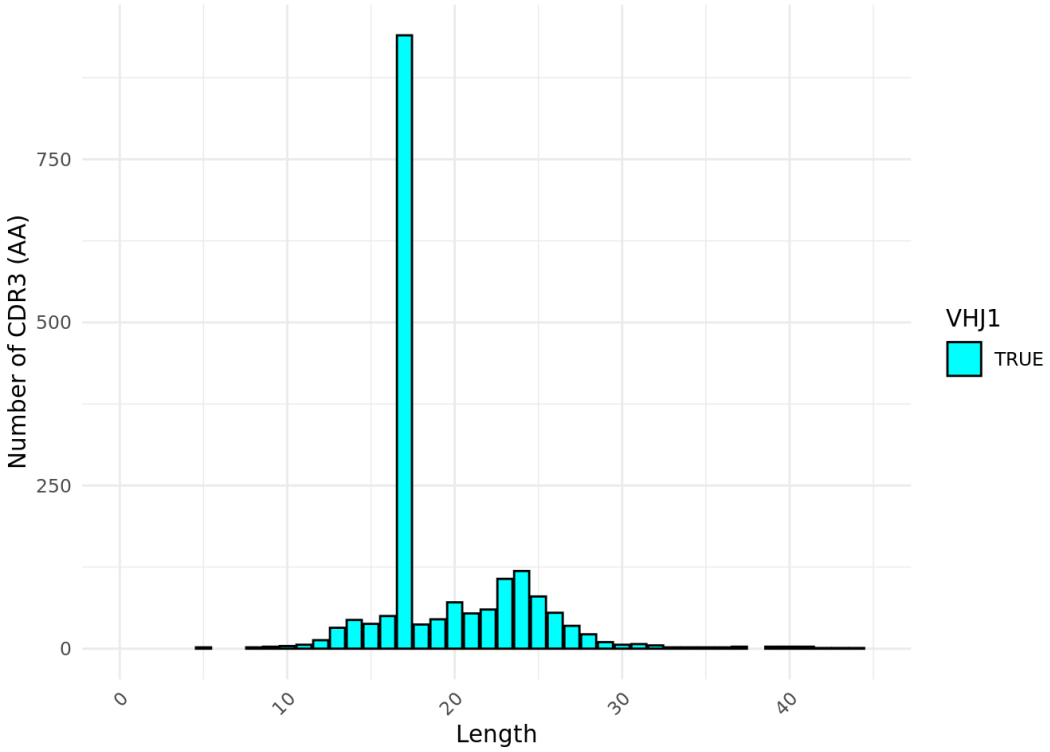
```
# Graficando con ggplot2
ggplot(df_plot, aes(x = len_aa_CDR3, y = cell_count, fill = VHJ1_group)) +
  geom_bar(stat = "identity", position = "dodge", color = "black") +
  scale_fill_manual(values = c("grey", "cyan")) +
  theme_minimal() +
  labs(x = "Length", y = "Number of CDR3 (AA)", fill = "VHJ1") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), text = element_text(family = "Calibri")) +
  scale_x_continuous(limits = c(0, 45))
```

```
## Warning: Removed 26 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```



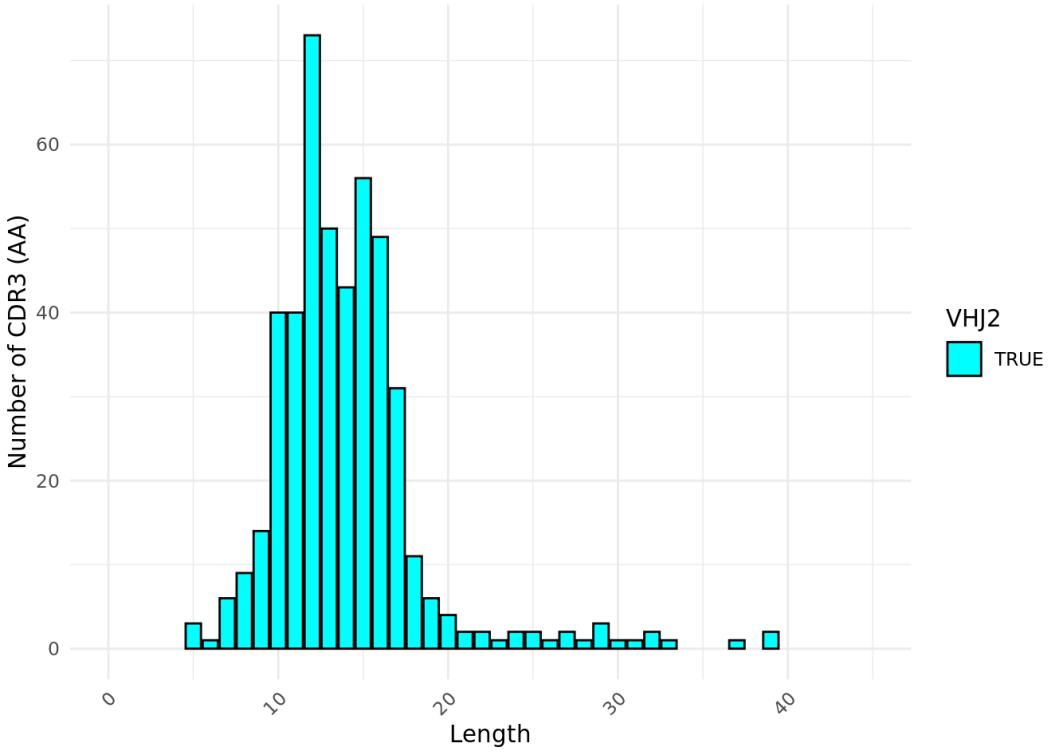

```
df_plot <- combined_BCR_BM_A12_MOUSE_MIX %>%
  mutate(VHJ1_group = ifelse(grepl("IGHJ1", IGH), "TRUE", "FALSE")) %>%
  group_by(len_aa_CDR3, VHJ1_group) %>%
  summarise(cell_count = n(), .groups = "drop")

# Graficando con ggplot2
ggplot(df_plot[df_plot$VHJ1_group == "TRUE",], aes(x = len_aa_CDR3, y = cell_count, fill = VHJ1_group)) +
  geom_bar(stat = "identity", position = "dodge", color = "black") +
  scale_fill_manual(values = c("cyan")) +
  theme_minimal() +
  labs(x = "Length", y = "Number of CDR3 (AA)", fill = "VHJ1") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), text = element_text(family = "Calibri")) +
  scale_x_continuous(limits = c(0, 45))
```

```
## Warning: Removed 10 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```



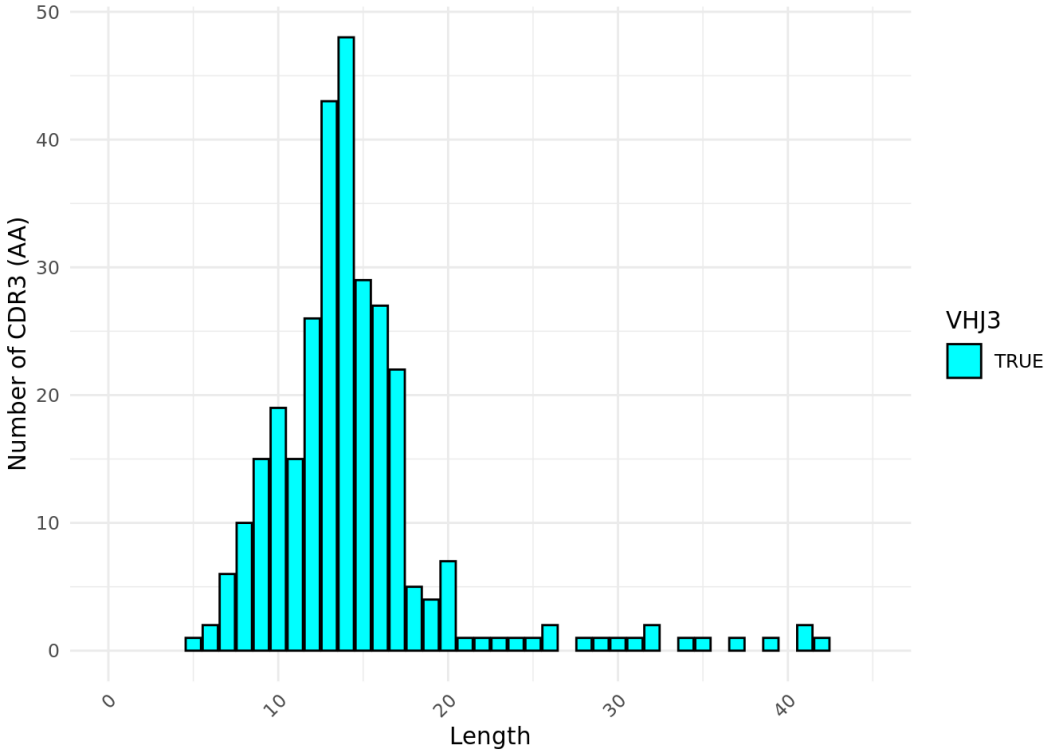

```
df_plot <- combined_BCR_BM_A12_MOUSE_MIX %>%
  mutate(VHJ2_group = ifelse(grepl("IGHJ2", IGH), "TRUE", "FALSE")) %>%
  group_by(len_aa_CDR3, VHJ2_group) %>%
  summarise(cell_count = n(), .groups = "drop")

# Graficando con ggplot2
ggplot(df_plot[df_plot$VHJ2_group == "TRUE",], aes(x = len_aa_CDR3, y = cell_count, fill = VHJ2_group)) +
  geom_bar(stat = "identity", position = "dodge", color = "black") +
  scale_fill_manual(values = c("cyan")) +
  theme_minimal() +
  labs(x = "Length", y = "Number of CDR3 (AA)", fill = "VHJ2") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), text = element_text(family = "Calibri")) +
  scale_x_continuous(limits = c(0, 45))
```

```
## Warning: Removed 5 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```



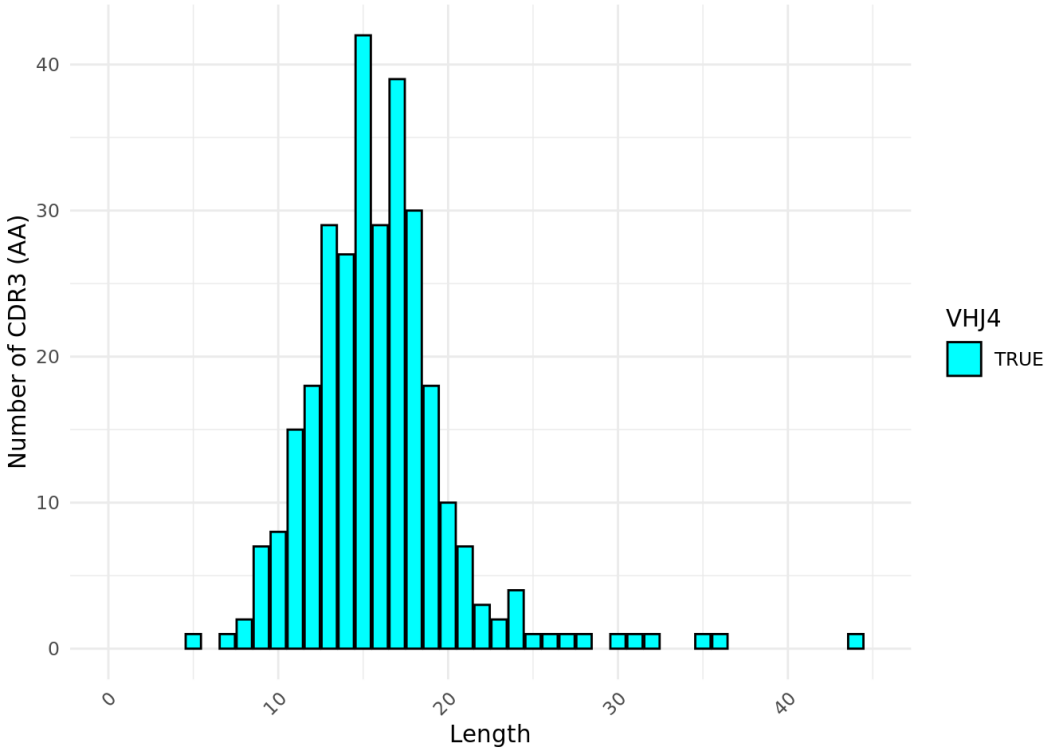

```
df_plot <- combined_BCR_BM_A12_MOUSE_MIX %>%
  mutate(VHJ3_group = ifelse(grepl("IGHJ3", IGH), "TRUE", "FALSE")) %>%
  group_by(len_aa_CDR3, VHJ3_group) %>%
  summarise(cell_count = n(), .groups = "drop")

# Graficando con ggplot2
ggplot(df_plot[df_plot$VHJ3_group == "TRUE",], aes(x = len_aa_CDR3, y = cell_count, fill = VHJ3_group)) +
  geom_bar(stat = "identity", position = "dodge", color = "black") +
  scale_fill_manual(values = c("cyan")) +
  theme_minimal() +
  labs(x = "Length", y = "Number of CDR3 (AA)", fill = "VHJ3") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), text = element_text(family = "Calibri")) +
  scale_x_continuous(limits = c(0, 45))
```

```
## Warning: Removed 5 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```




```
df_plot <- combined_BCR_BM_A12_MOUSE_MIX %>%
  mutate(VHJ4_group = ifelse(grepl("IGHJ4", IGH), "TRUE", "FALSE")) %>%
  group_by(len_aa_CDR3, VHJ4_group) %>%
  summarise(cell_count = n(), .groups = "drop")

# Graficando con ggplot2
ggplot(df_plot[df_plot$VHJ4_group == "TRUE",], aes(x = len_aa_CDR3, y = cell_count, fill = VHJ4_group)) +
  geom_bar(stat = "identity", position = "dodge", color = "black") +
  scale_fill_manual(values = c("cyan")) +
  theme_minimal() +
  labs(x = "Length", y = "Number of CDR3 (AA)", fill = "VHJ4") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), text = element_text(family = "Calibri")) +
  scale_x_continuous(limits = c(0, 45))
```

```
## Warning: Removed 9 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```

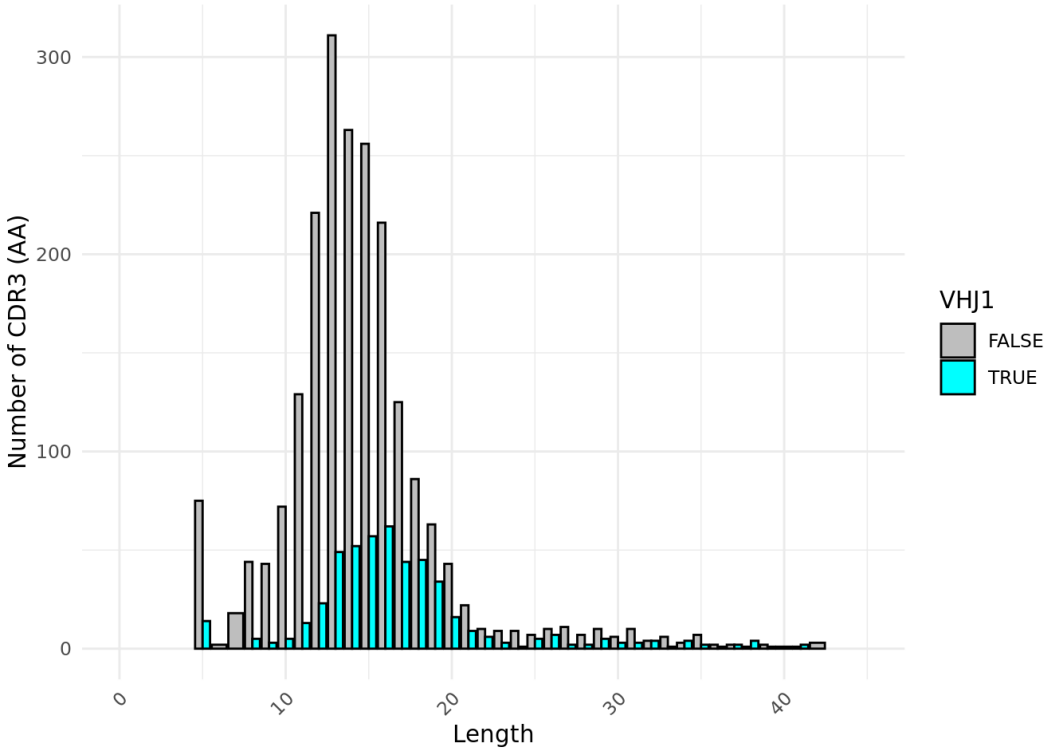



```
# Creando un nuevo dataframe con las columnas necesarias
df_plot <- combined_BCR_SP_WT_MOUSE_MIX %>%
  mutate(VHJ1_group = ifelse(grepl("IGHJ1", IGH), "TRUE", "FALSE")) %>%
  group_by(len_aa_CDR3, VHJ1_group) %>%
  summarise(cell_count = n())
```

```
## `summarise()` has grouped output by 'len_aa_CDR3'. You can override using the
## `.groups` argument.
```

```
# Graficando con ggplot2
ggplot(df_plot, aes(x = len_aa_CDR3, y = cell_count, fill = VHJ1_group)) +
  geom_bar(stat = "identity", position = "dodge", color = "black") +
  scale_fill_manual(values = c("grey", "cyan")) +
  theme_minimal() +
  labs(x = "Length", y = "Number of CDR3 (AA)", fill = "VHJ1") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), text = element_text(family = "Calibri")) +
  scale_x_continuous(limits = c(0, 45))
```

```
## Warning: Removed 43 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```

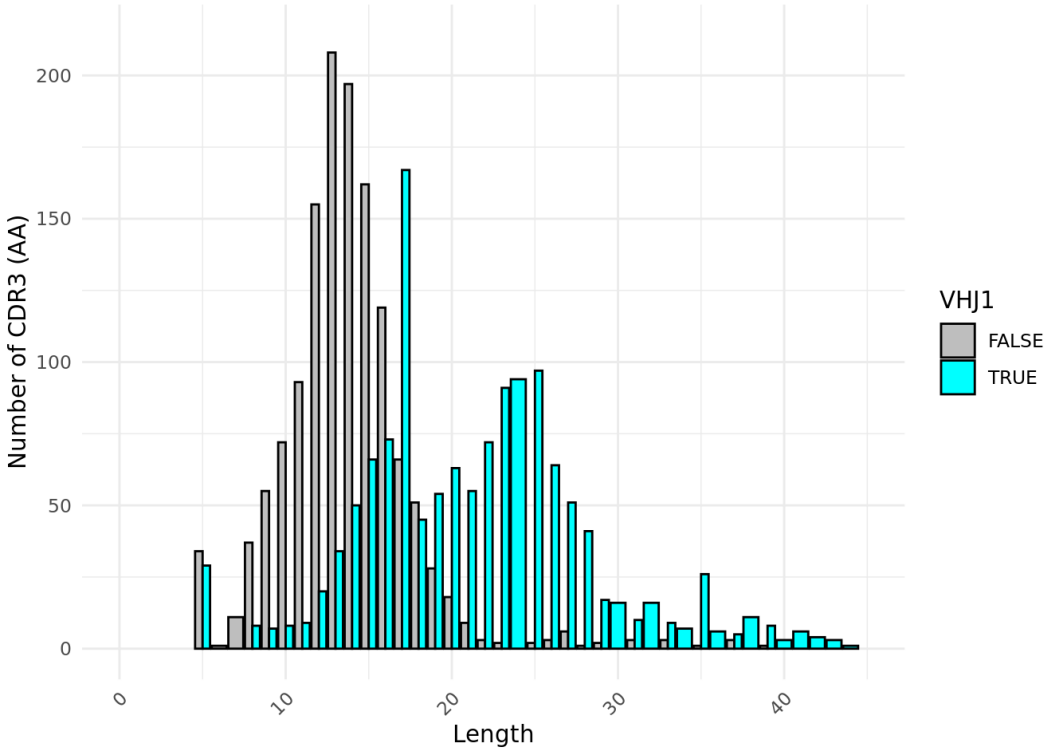



```
df_plot <- combined_BCR_SP_A12_MOUSE_MIX %>%
  mutate(VHJ1_group = ifelse(grepl("IGHJ1", IGH), "TRUE", "FALSE")) %>%
  group_by(len_aa_CDR3, VHJ1_group) %>%
  summarise(cell_count = n())
```

```
## `summarise()` has grouped output by 'len_aa_CDR3'. You can override using the
## `.groups` argument.
```

```
# Graficando con ggplot2
ggplot(df_plot, aes(x = len_aa_CDR3, y = cell_count, fill = VHJ1_group)) +
  geom_bar(stat = "identity", position = "dodge", color = "black") +
  scale_fill_manual(values = c("grey", "cyan")) +
  theme_minimal() +
  labs(x = "Length", y = "Number of CDR3 (AA)", fill = "VHJ1") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), text = element_text(family = "Calibri")) +
  scale_x_continuous(limits = c(0, 45))
```

```
## Warning: Removed 41 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```

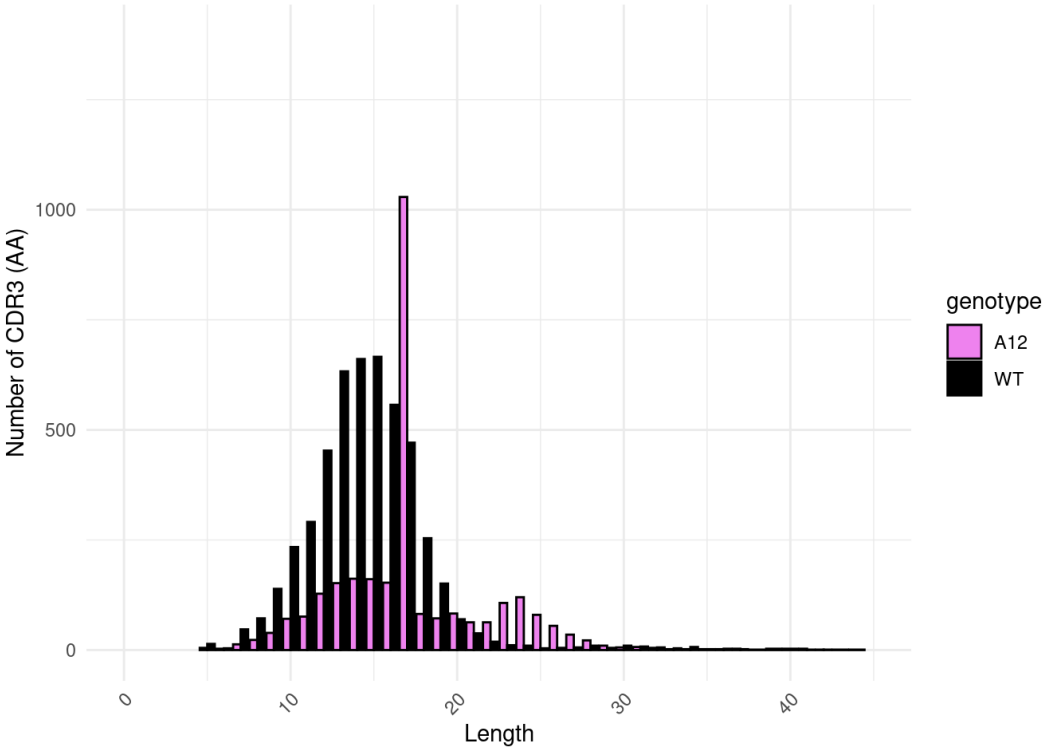



```
df_plot <- combined_BCR_BM_MOUSE_MIX %>%
  group_by(len_aa_CDR3, genotype) %>%
  summarise(cell_count = n())
```

```
## `summarise()` has grouped output by 'len_aa_CDR3'. You can override using the
## `.groups` argument.
```

```
# Graficando con ggplot2
ggplot(df_plot, aes(x = len_aa_CDR3, y = cell_count, fill = genotype)) +
  geom_bar(stat = "identity", position = "dodge", color = "black") +
  scale_fill_manual(values = c("violet", "black")) +
  theme_minimal() +
  labs(x = "Length", y = "Number of CDR3 (AA)", fill = "genotype") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_x_continuous(limits = c(0, 45))
```

```
## Warning: Removed 41 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```

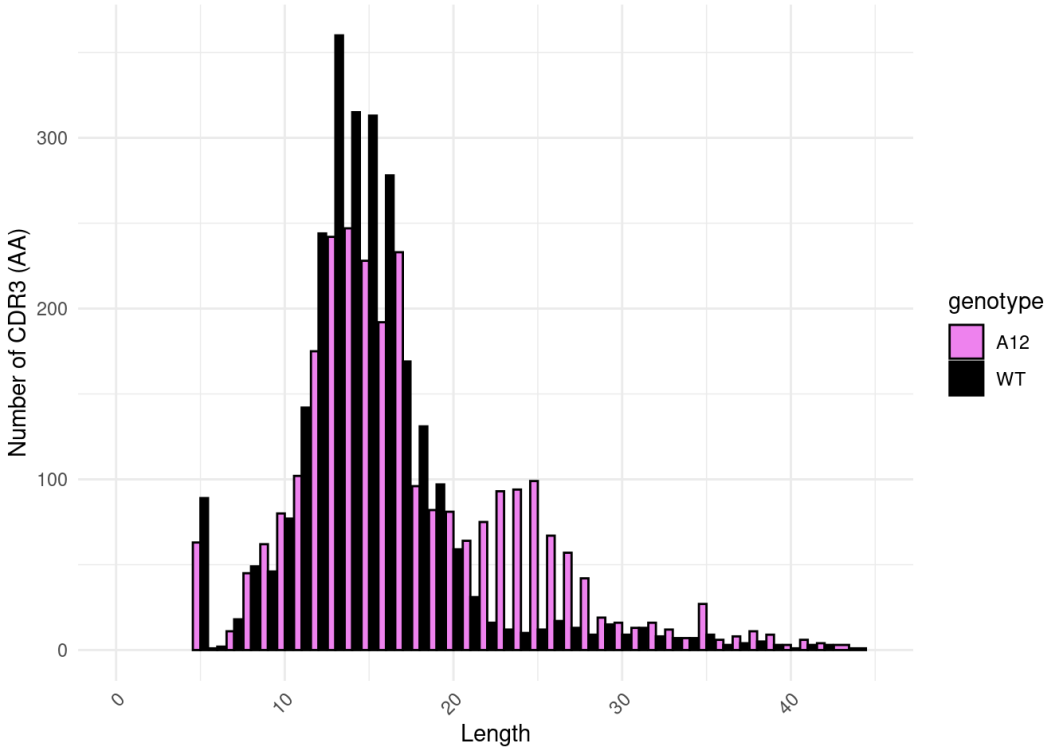



```
df_plot <- combined_BCR_SP_MOUSE_MIX %>%
  group_by(len_aa_CDR3, genotype) %>%
  summarise(cell_count = n())
```

```
## `summarise()` has grouped output by 'len_aa_CDR3'. You can override using the
## `.groups` argument.
```

```
# Graficando con ggplot2
ggplot(df_plot, aes(x = len_aa_CDR3, y = cell_count, fill = genotype)) +
  geom_bar(stat = "identity", position = "dodge", color = "black") +
  scale_fill_manual(values = c("violet", "black")) +
  theme_minimal() +
  labs(x = "Length", y = "Number of CDR3 (AA)", fill = "genotype") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_x_continuous(limits = c(0, 45))
```

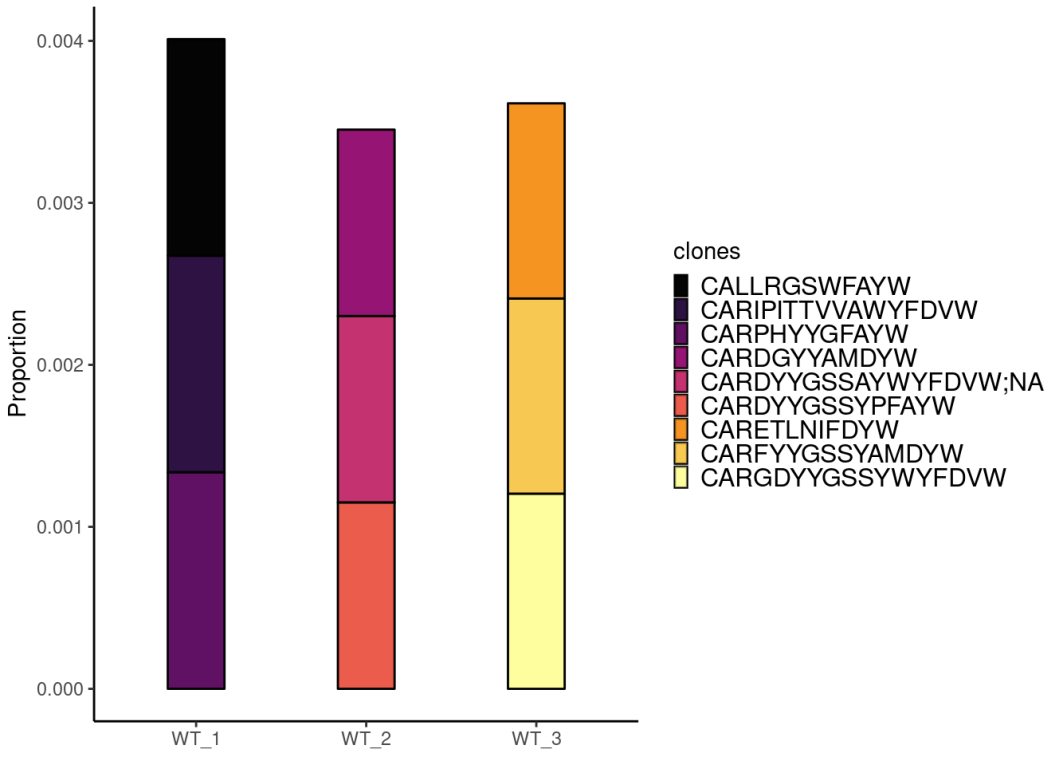
```
## Warning: Removed 74 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```



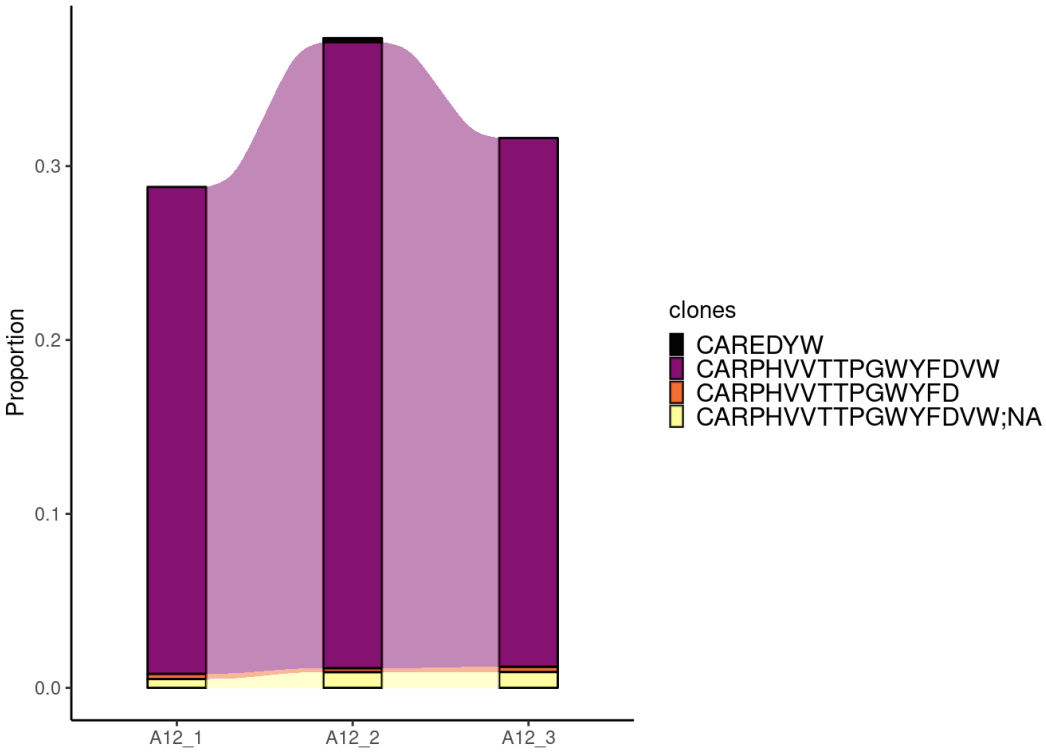
Clonal compare

We can also look at clones between samples and changes in dynamics by using the clonalCompare() function.

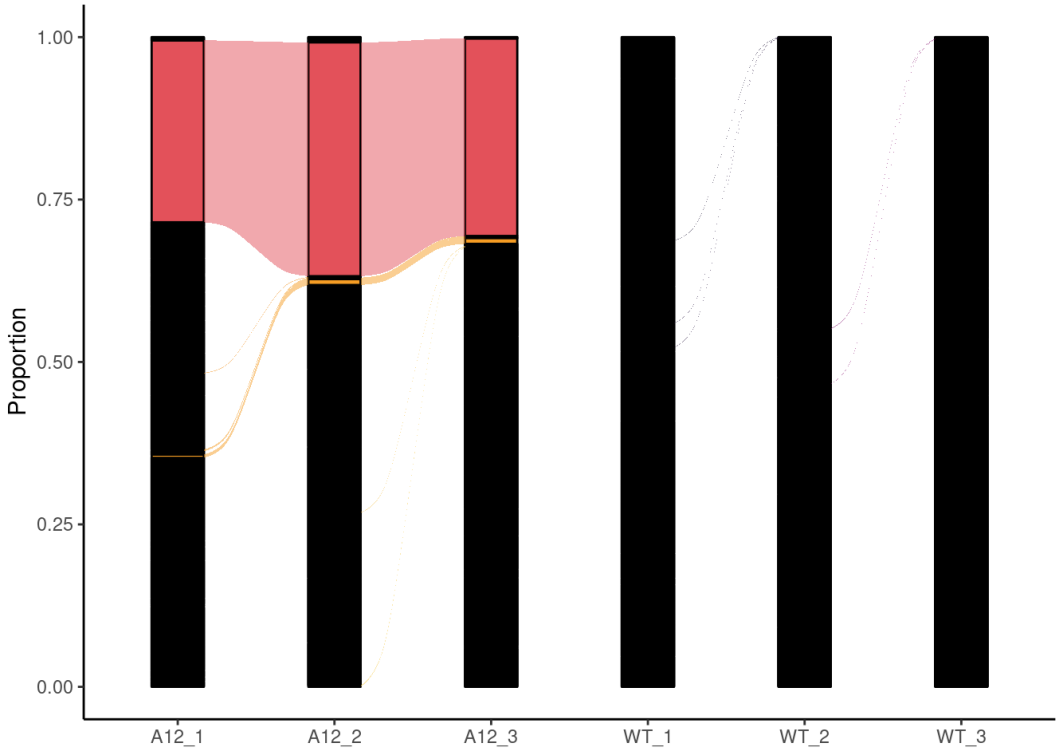
```
q <- clonalCompare(combined_BCR_BM,
                   top.clones = 3,
                   samples = c("WT_1", "WT_2", "WT_3"),
                   cloneCall="aa",
                   chain = "IGH",
                   graph = "alluvial")
q + theme(legend.text = element_text(size = 12))
```



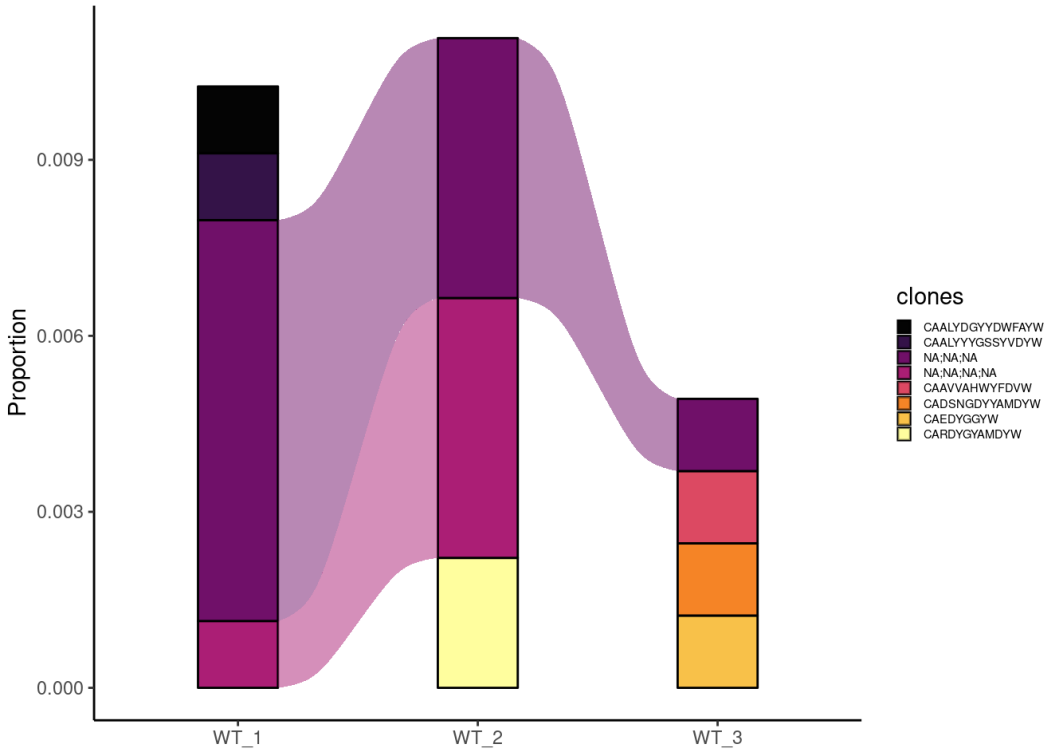
```
# Captura el gráfico generado por clonalCompare
p <- clonalCompare(combined_BCR_BM,
                   top.clones = 3,
                   samples = c("A12_1", "A12_2", "A12_3"),
                   cloneCall = "aa",
                   chain = "IGH",
                   graph = "alluvial")

# Modifica el gráfico para ajustar el tamaño de la leyenda
p + theme(legend.text = element_text(size = 12))  # Ajusta el valor de 'size' según prefieras
```




```
r <- clonalCompare(combined_BCR_BM,
                   top.clones = 2000,
                   samples = c("WT_1", "WT_2", "WT_3", "A12_1", "A12_2", "A12_3"),
                   cloneCall="aa",
                   chain = "IGH",
                   graph = "alluvial")
r + theme(legend.position = "none")
```

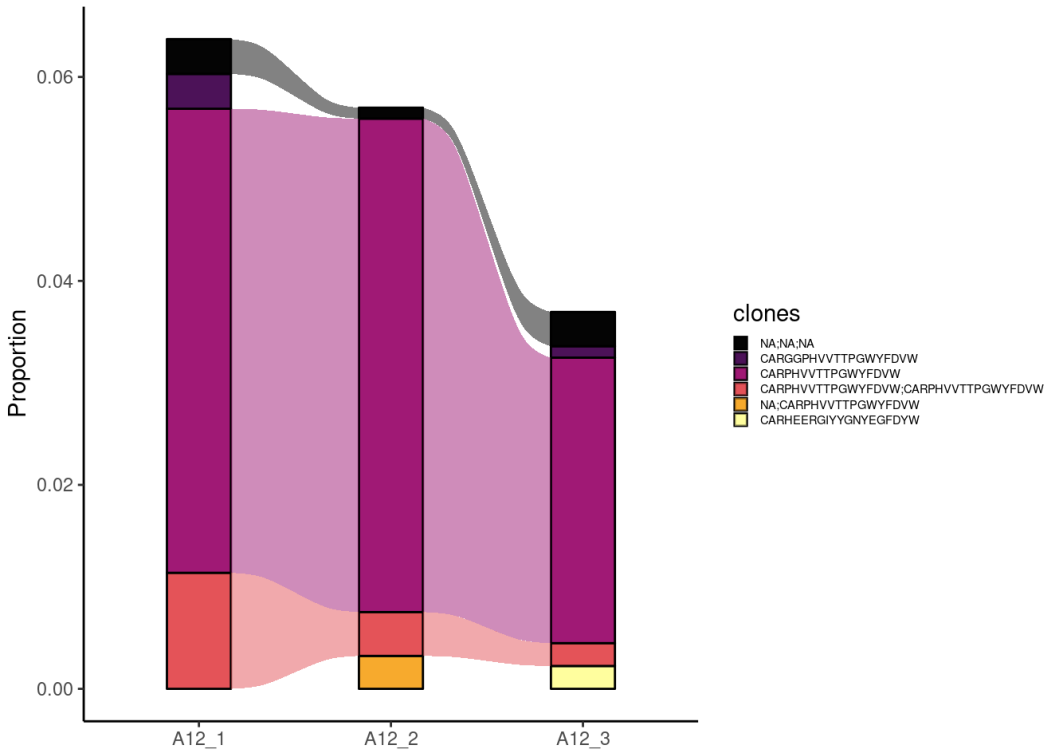



```
clonalCompare(combined_BCR_SP,
                  top.clones = 3,
                  samples = c("WT_1", "WT_2", "WT_3"),
                  cloneCall="aa",
                  chain = "IGH",
                  graph = "alluvial")
```

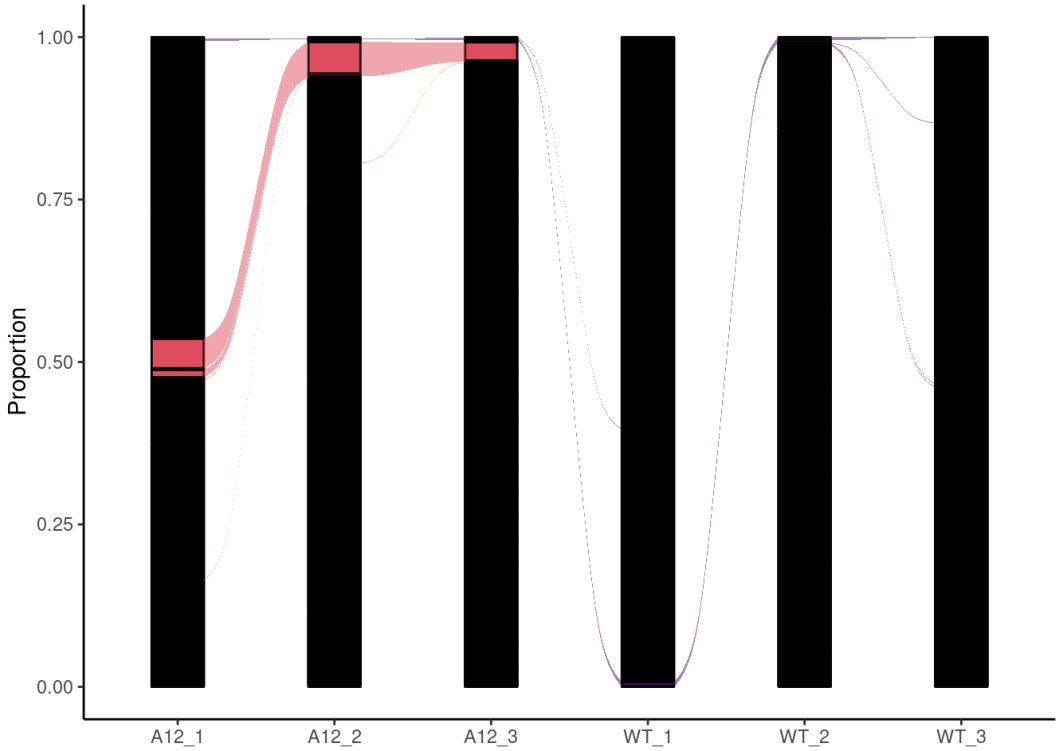



```
clonalCompare(combined_BCR_SP,
                  top.clones = 3,
                  samples = c("A12_1", "A12_2", "A12_3"),
                  chain = "IGH",
                  cloneCall="aa",
```

```
                graph = "alluvial")
```



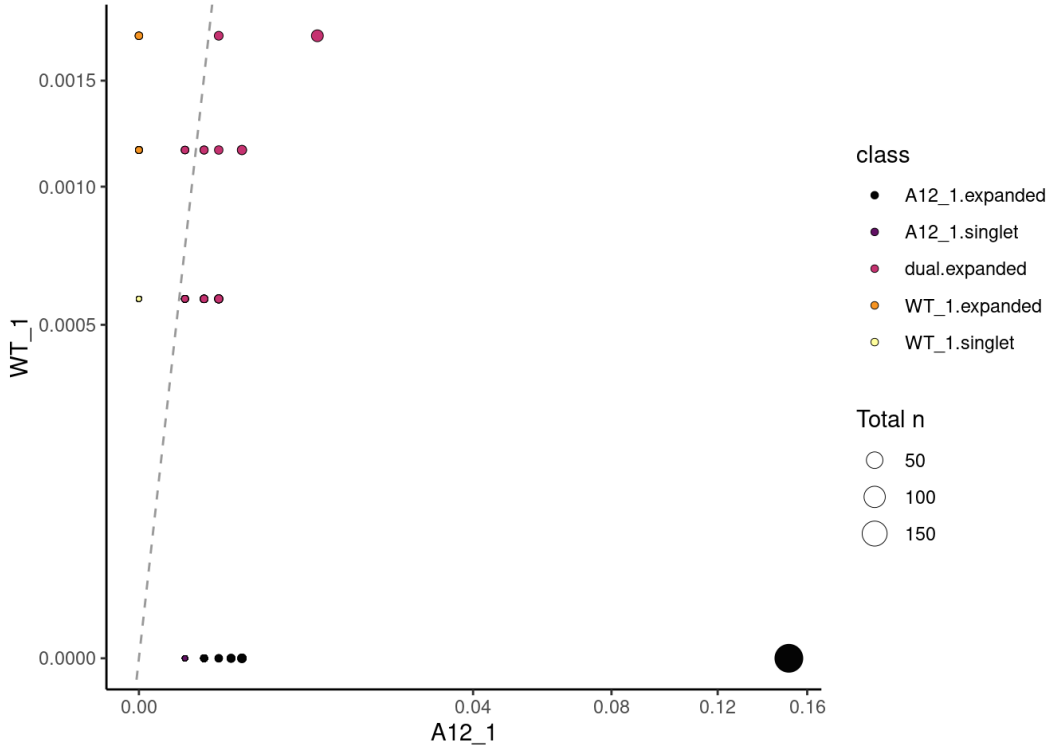

```
clonalCompare(combined_BCR_SP,
                top.clones = 2000,
                samples = c("WT_1", "WT_2", "WT_3", "A12_1", "A12_2", "A12_3"),
                cloneCall="aa",
                chain = "IGH",
                graph = "alluvial") + theme(legend.position = "none")
```



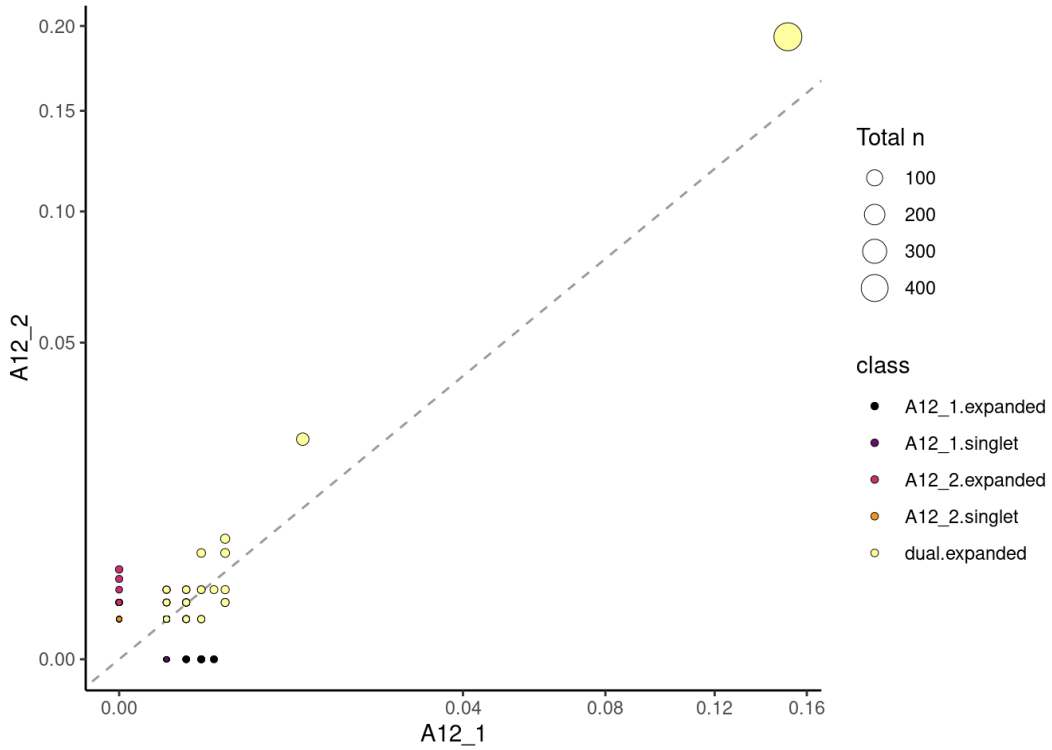
clonalScatter() will organize two repertoires, quantify the relative clone sizes, and produce a scatter plot comparing the two samples.

```
clonalScatter(combined_BCR_BM,
```

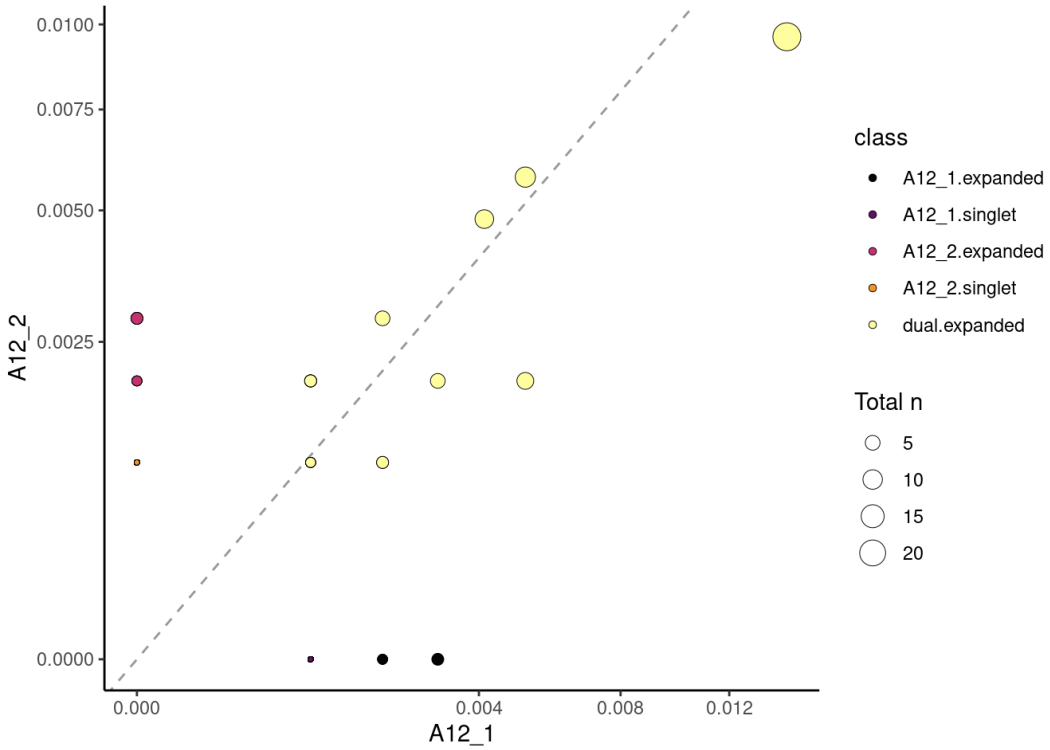
```
                cloneCall ="aa",
                x.axis = "A12_1",
                y.axis = "WT_1",
                dot.size = "total",
                graph = "proportion")
```



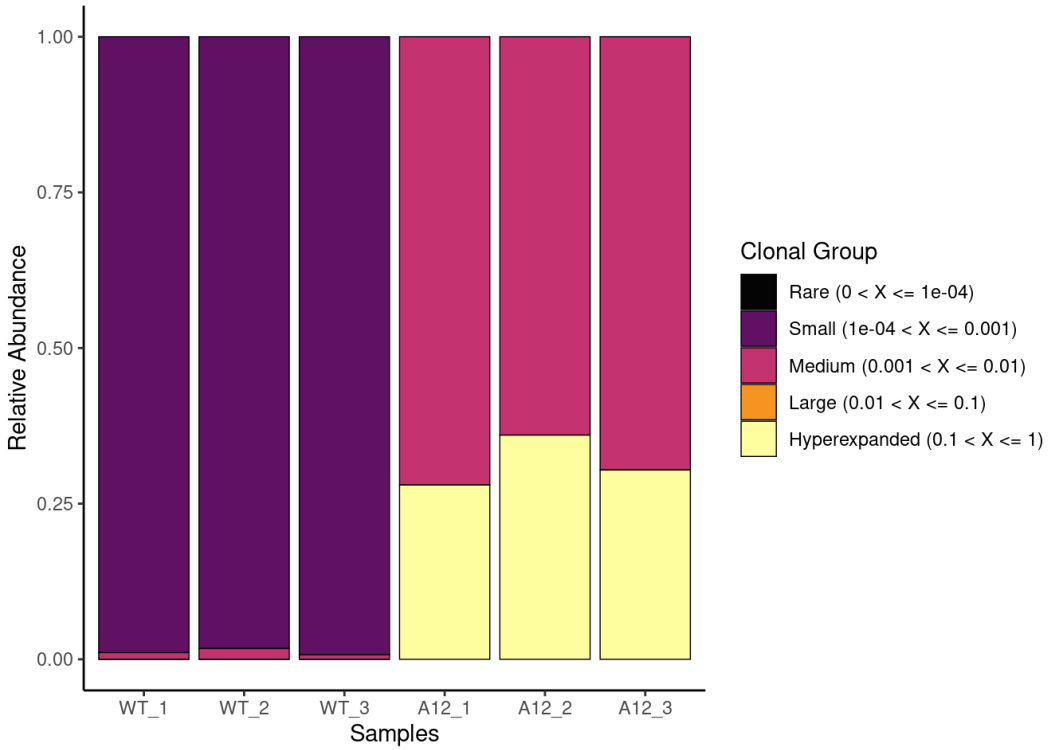

```
clonalScatter(combined_BCR_BM,
                cloneCall ="aa",
                x.axis = "A12_1",
                y.axis = "A12_2",
                dot.size = "total",
                graph = "proportion")
```



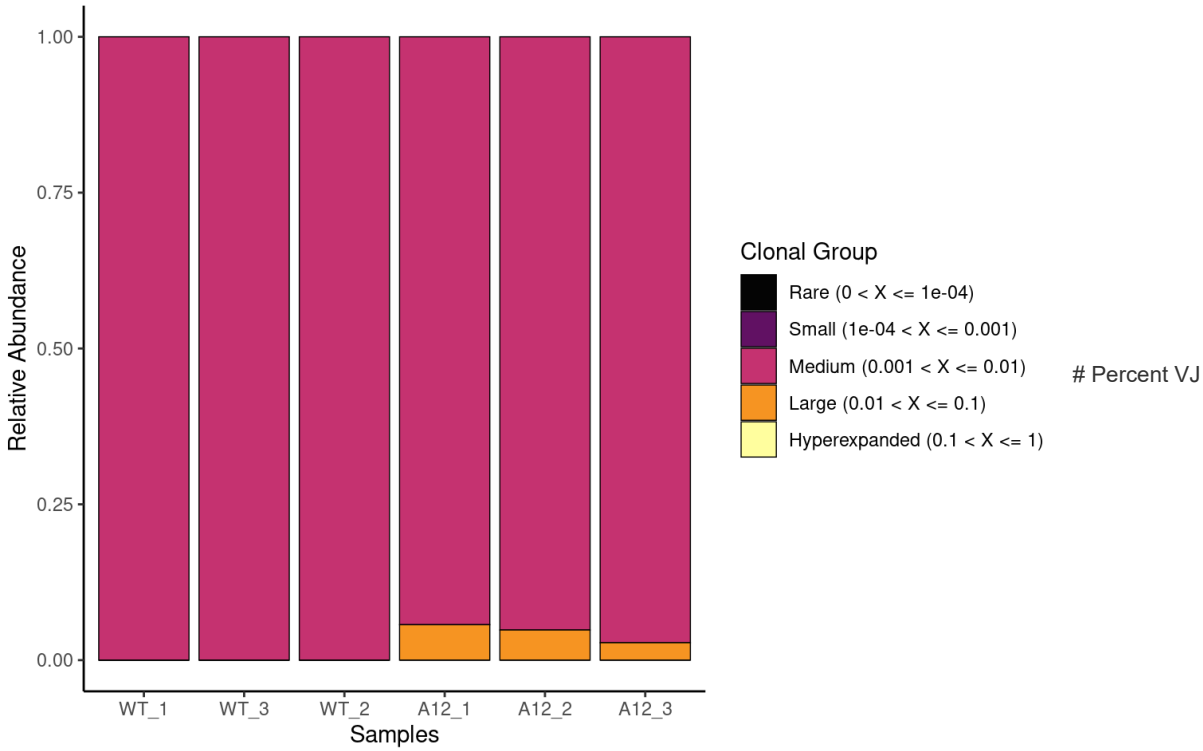
```
clonalScatter(combined_BCR_SP,
              cloneCall ="aa",
              x.axis = "A12_1",
              y.axis = "A12_2",
              dot.size = "total",
              graph = "proportion")
```



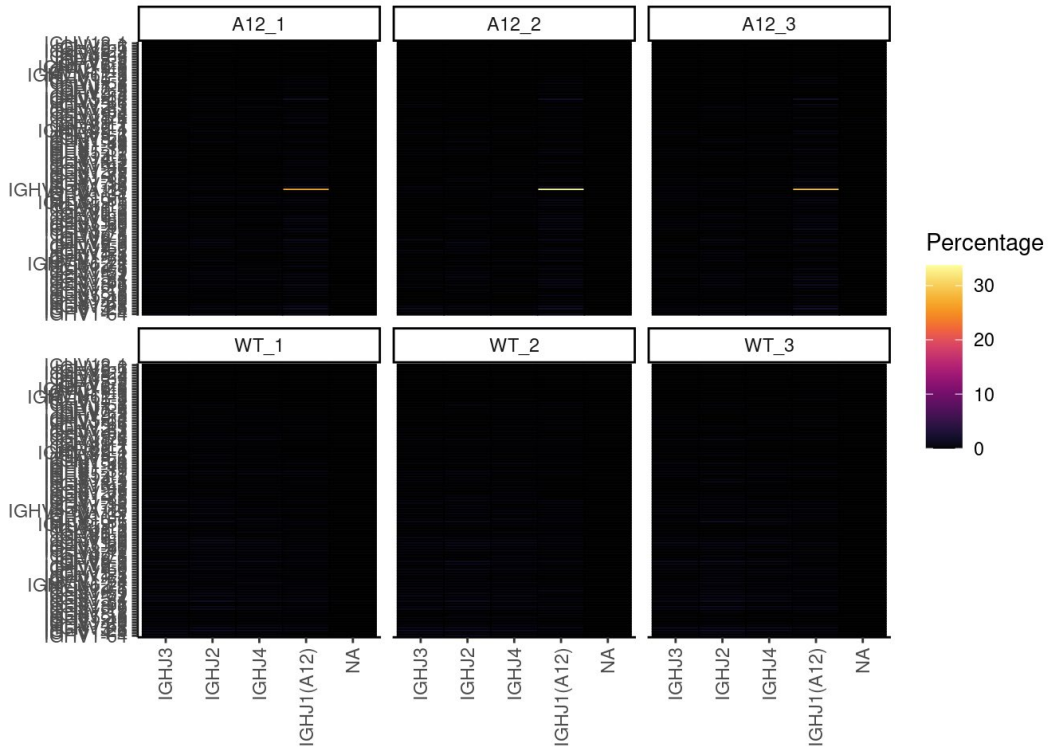

```
clonalHomeostasis(combined_BCR_BM,
                  chain = "IGH",
                  cloneCall = "aa")
```



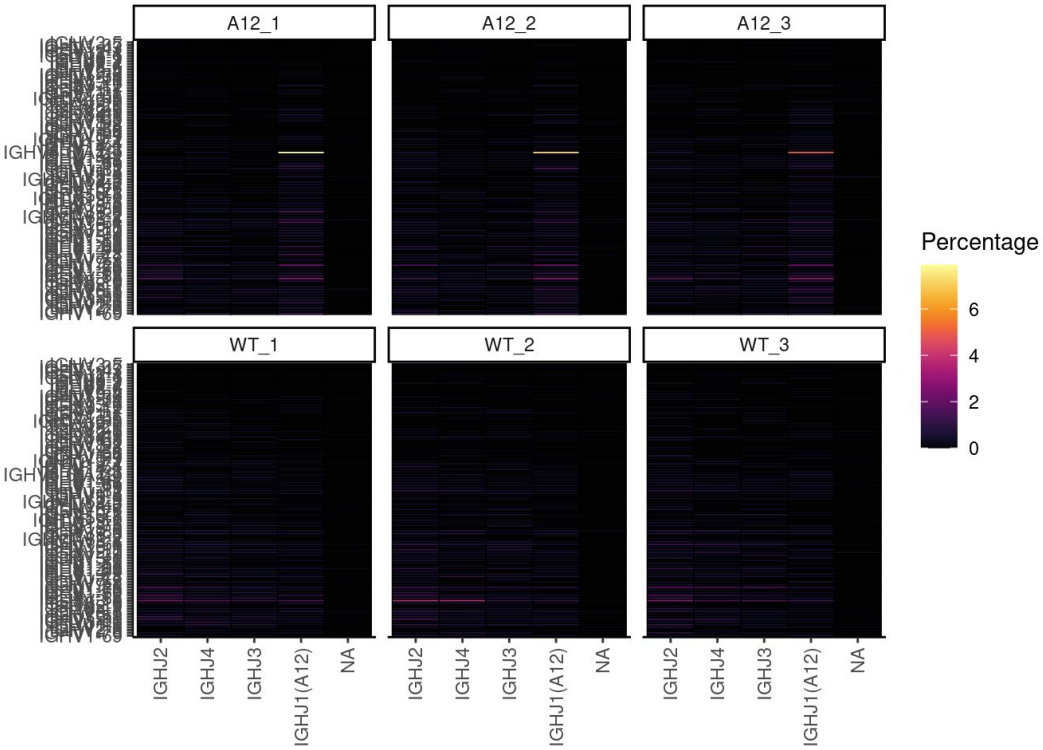
```
clonalHomeostasis(combined_BCR_SP,
                  chain = "IGH",
                  cloneCall = "aa")
```




```
percentVJ(combined_BCR_BM, chain = "IGH")
```

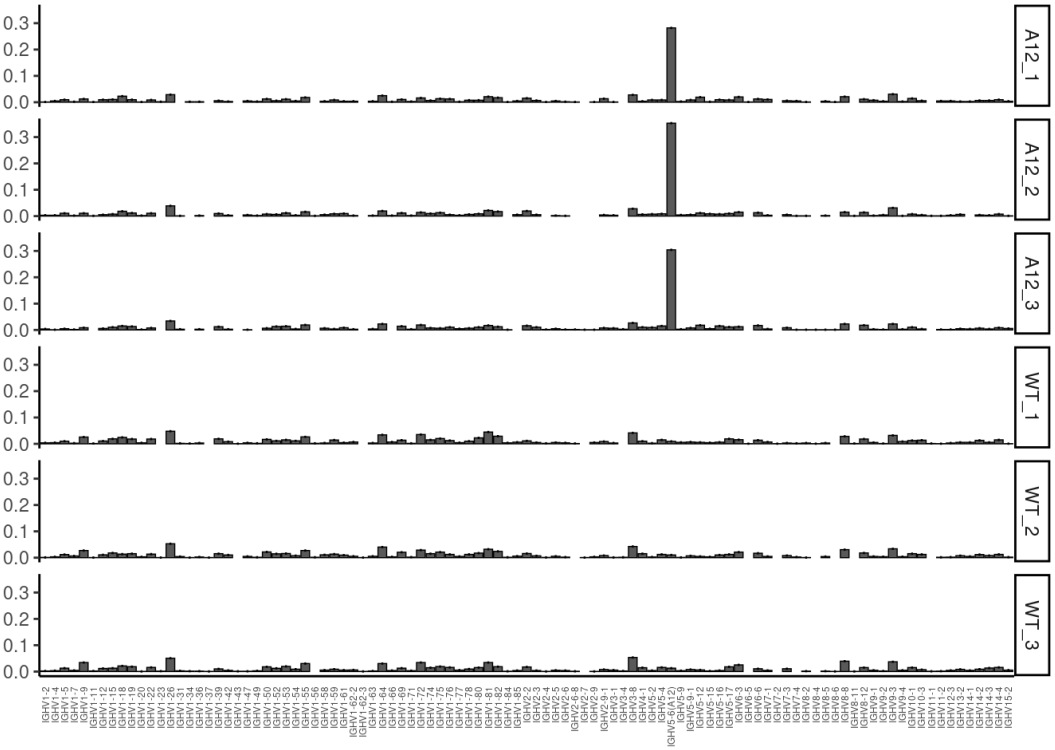



```
percentVJ(combined_BCR_SP, chain = "IGH")
```



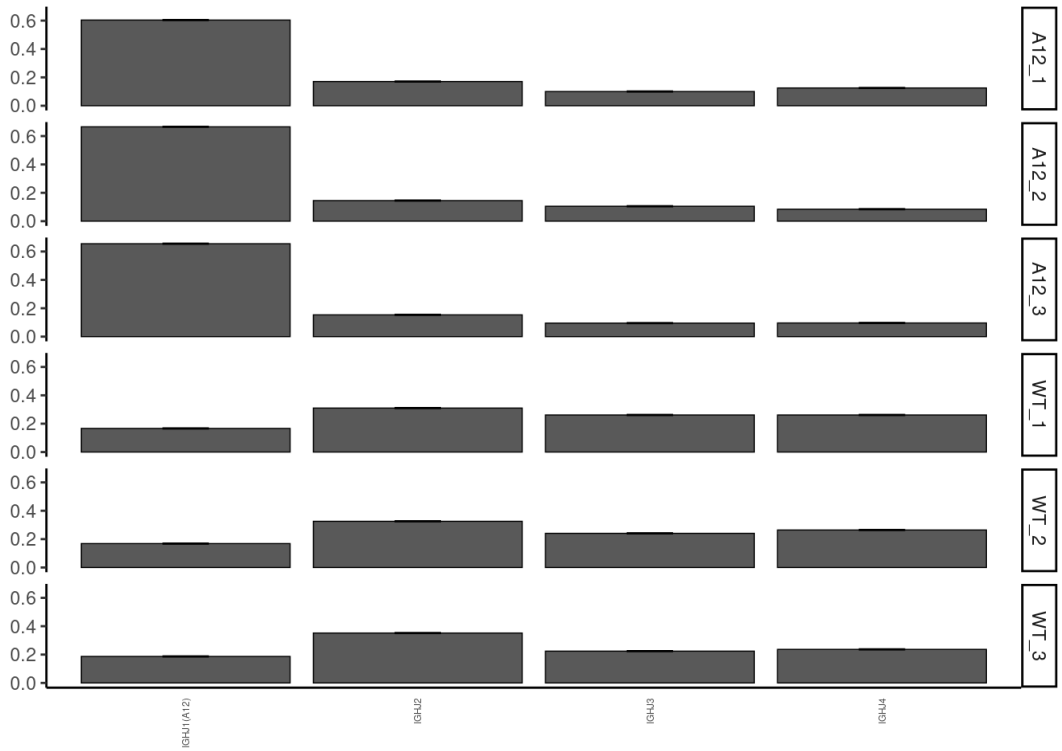
Sumarizing Repertoires for BM

```
vizGenes(combined_BCR_BM,
         x.axis = "IGHV",
         y.axis = NULL,
         plot = "barplot",
         scale = TRUE)
```




```
vizGenes(combined_BCR_BM,
         x.axis = "IGHJ",
         y.axis = NULL,
```

```
        plot = "barplot",
        scale = TRUE)
```



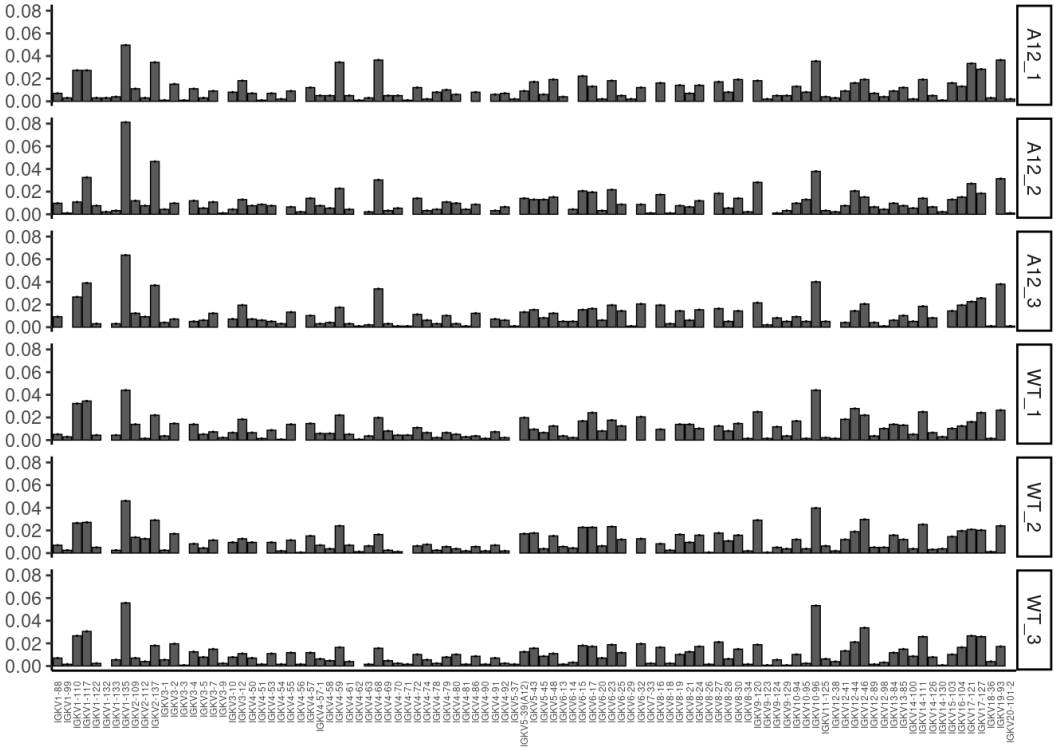

```
Seurat_BM_2 <- Seurat_BM
Seurat_SP_2 <- Seurat_SP
```

Then we modify the meta.data of the seurat object

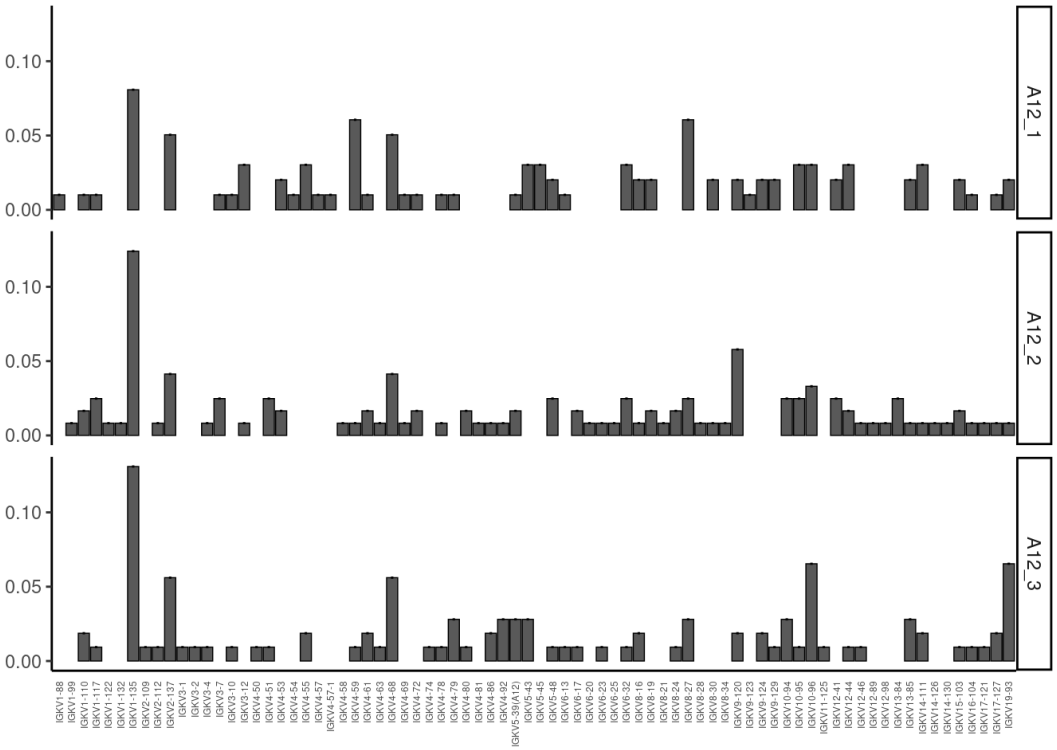
```
cell.barcodes <- rownames(Seurat_BM_2[[]])
# remove the _1 at the end of the barcodes)
cell.barcodes <- stringr::str_split(cell.barcodes, "_", simplify = TRUE)[, 1]
# add the prefix of the orig.ident to the barcodes
cell.barcodes <- paste0(Seurat_BM_2$mice, "_", cell.barcodes)
Seurat_BM_2 <- RenameCells(Seurat_BM_2, new.names = cell.barcodes)

cell.barcodes <- rownames(Seurat_SP_2[[]])
# remove the _1 at the end of the barcodes
cell.barcodes <- stringr::str_split(cell.barcodes, "_", simplify = TRUE)[, 1]
# add the prefix of the orig.ident to the barcodes
cell.barcodes <- paste0(Seurat_SP_2$mice, "_", cell.barcodes)
Seurat_SP_2 <- RenameCells(Seurat_SP_2, new.names = cell.barcodes)
```

```
barcodes <- names(Seurat_SP_2$orig.ident[Seurat_SP_2$seurat_clusters_new_fin == "A12 expressing"])

combined_A12_UNIQ_POP <- subset(combined_BCR_SP_A12_MOUSE_MIX, barcode %in% barcodes)

View(combined_A12_UNIQ_POP)

write.csv(combined_A12_UNIQ_POP, "../Results/uniq_A12_population_SP_VDJ_OLD.csv", row.names = FALSE)
```

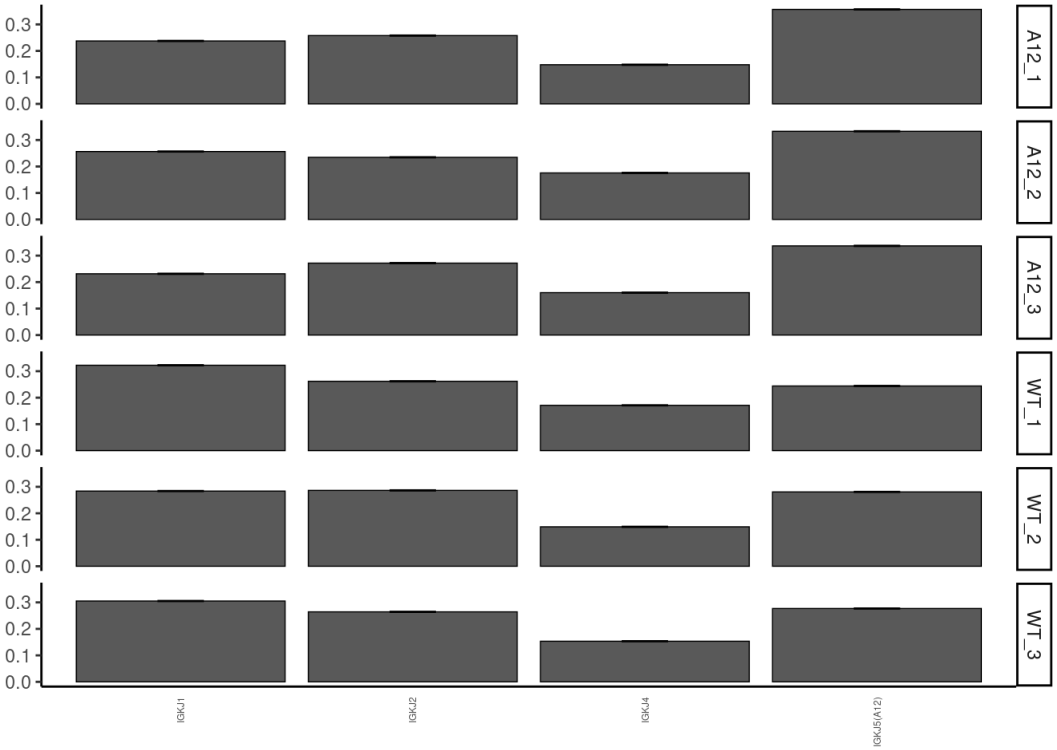
```
vizGenes(combined_BCR_BM,
        x.axis = "IGKV",
        y.axis = NULL,
        plot = "barplot",
        scale = TRUE)
```



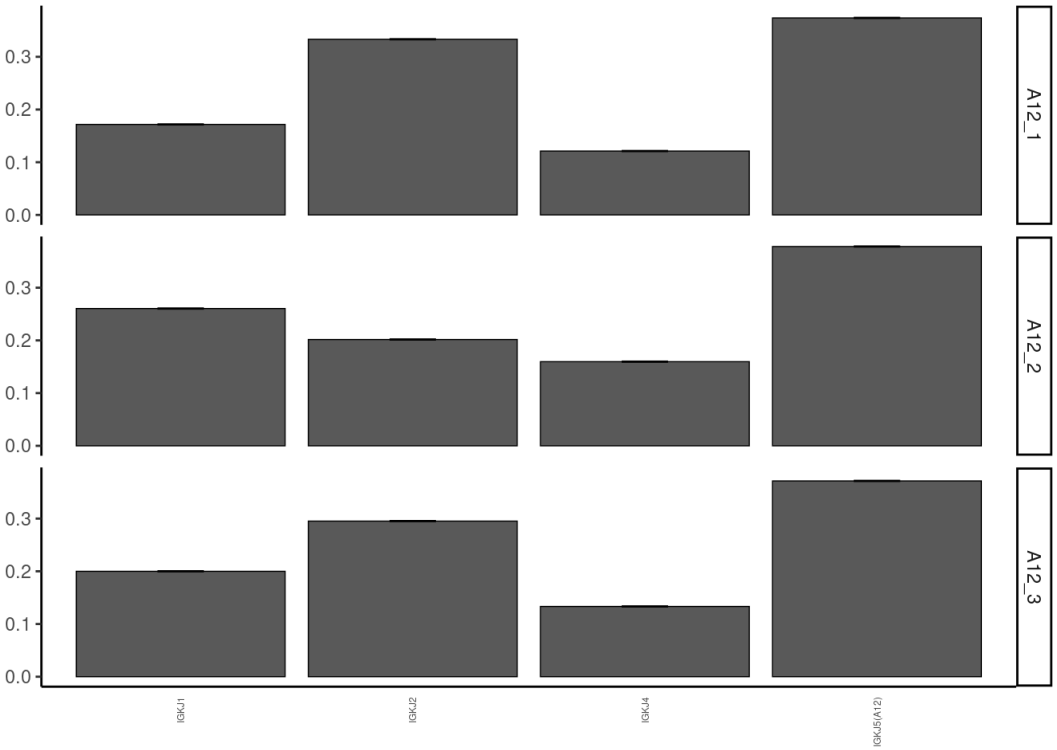

```
vizGenes(combined_BCR_BM_A12_IghA12,
         x.axis = "IGKV",
         y.axis = NULL,
         plot = "barplot",
         scale = TRUE)
```




```
vizGenes(combined_BCR_BM,
         x.axis = "IGKJ",
         y.axis = NULL,
         plot = "barplot",
         scale = TRUE)
```




```
vizGenes(combined_BCR_BM_A12_IghA12,
         x.axis = "IGKJ",
         y.axis = NULL,
         plot = "barplot",
         scale = TRUE)
```




```
IGHG2B <- (sum(grepl("IGHG2B", combined_A12_UNIQ_POP$IGH)))
IGHG3 <- (sum(grepl("IGHG3", combined_A12_UNIQ_POP$IGH)))
IGHA <- (sum(grepl("IGHA", combined_A12_UNIQ_POP$IGH)))
IGHG1 <- (sum(grepl("IGHG1", combined_A12_UNIQ_POP$IGH)))

A12_uniq_pop <- subset(Seurat_SP_2, seurat_clusters_new_fin == "A12 expressing")
```

```
Ighg2b_expression <- A12_uniq_pop@assays$RNA$counts["Ighg2b", ]
Ighg3_expression <- A12_uniq_pop@assays$RNA$counts["Ighg3", ]
Ighg1_expression <- A12_uniq_pop@assays$RNA$counts["Ighg1", ]
Igha_expression <- A12_uniq_pop@assays$RNA$counts["Igha", ]

num_cells_expressing_Ighg2b <- sum(Ighg2b_expression > 2)
num_cells_expressing_Ighg3 <- sum(Ighg3_expression > 2)
num_cells_expressing_Ighg1 <- sum(Ighg1_expression > 2)
num_cells_expressing_Igha <- sum(Igha_expression > 2)

cat("N cells Ighg2b:", IGHG2B)
```

```
## N cells Ighg2b: 5
```

```
cat("\nN cells Ighg3:", IGHG3)
```

```
##
## N cells Ighg3: 7
```

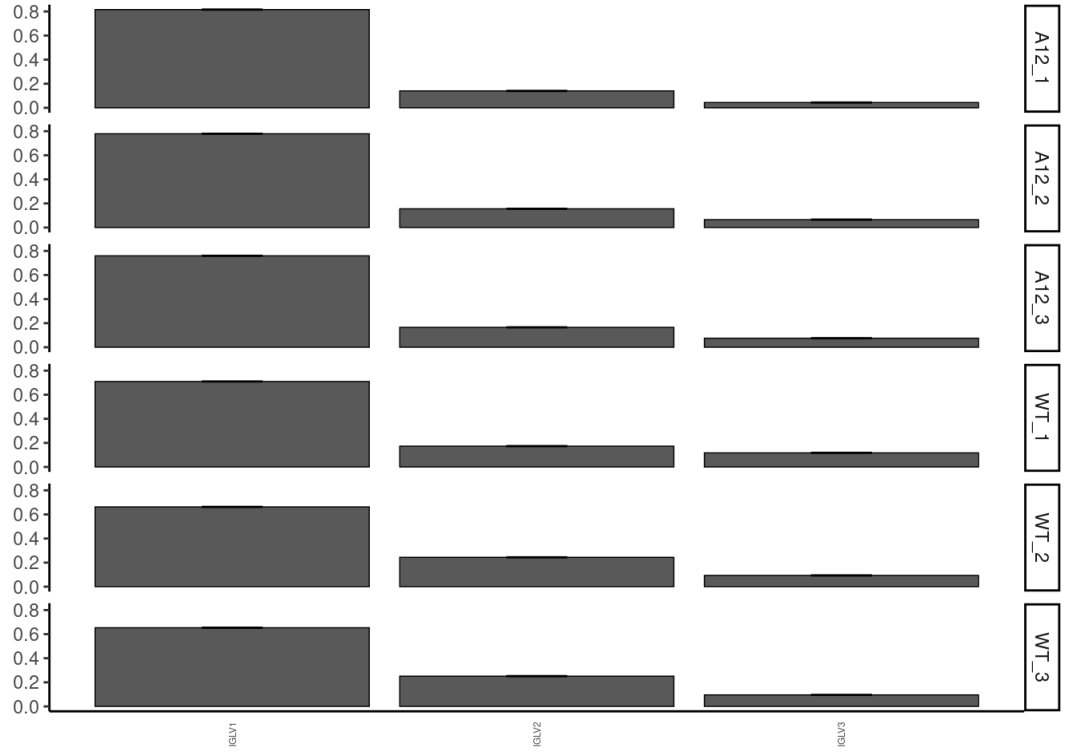
```
cat("\nN cells Ighg1:", IGHG1)
```

```
##
## N cells Ighg1: 1
```

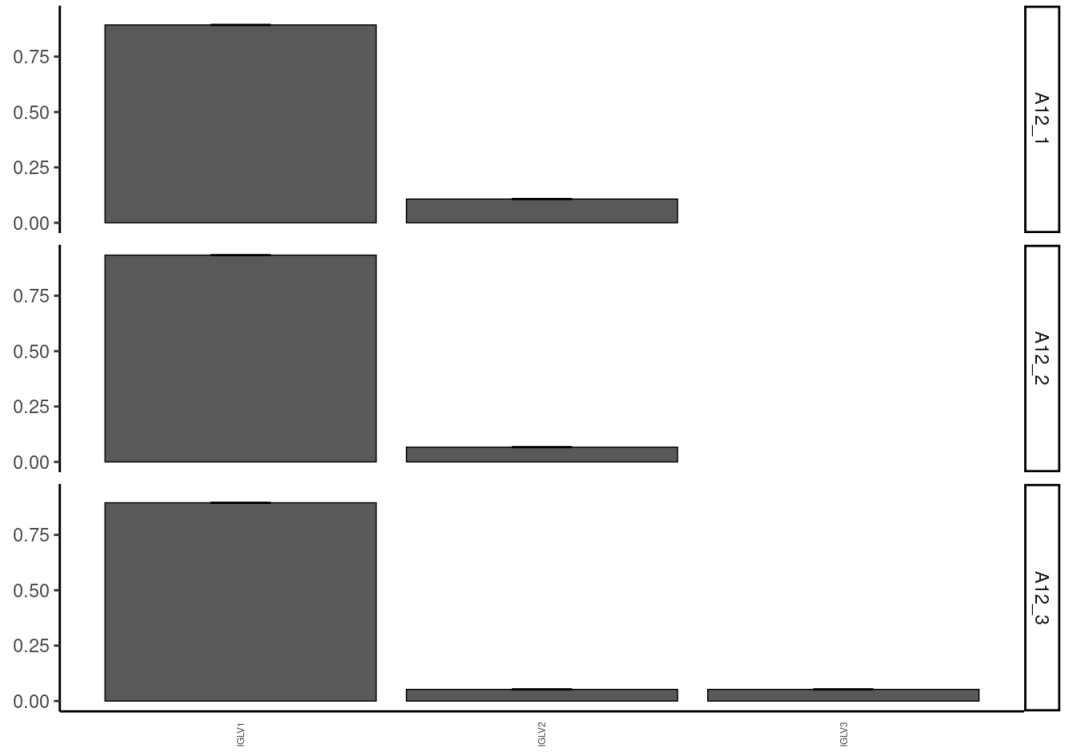
```
cat("\nN cells Igha:", IGHA)
```

```
##
## N cells Igha: 3
```

```
vizGenes(combined_BCR_BM,
         x.axis = "IGLV",
         y.axis = NULL,
         plot = "barplot",
         scale = TRUE)
```



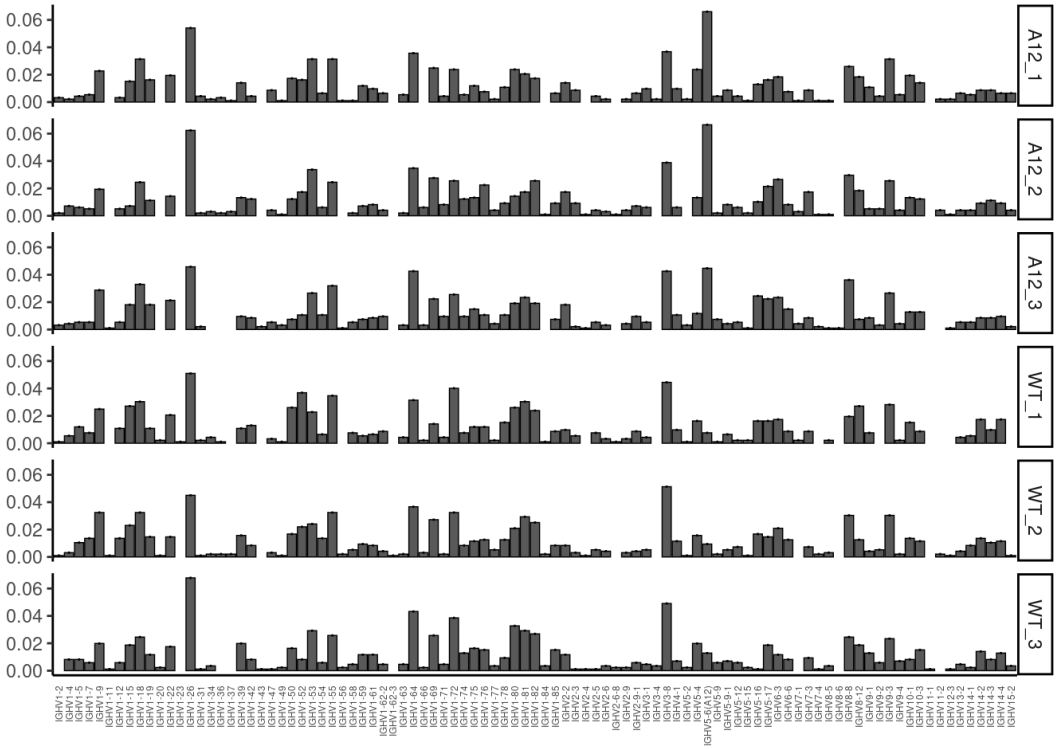

```
vizGenes(combined_BCR_BM_A12_IghA12,
         x.axis = "IGLV",
         y.axis = NULL,
         plot = "barplot",
         scale = TRUE)
```



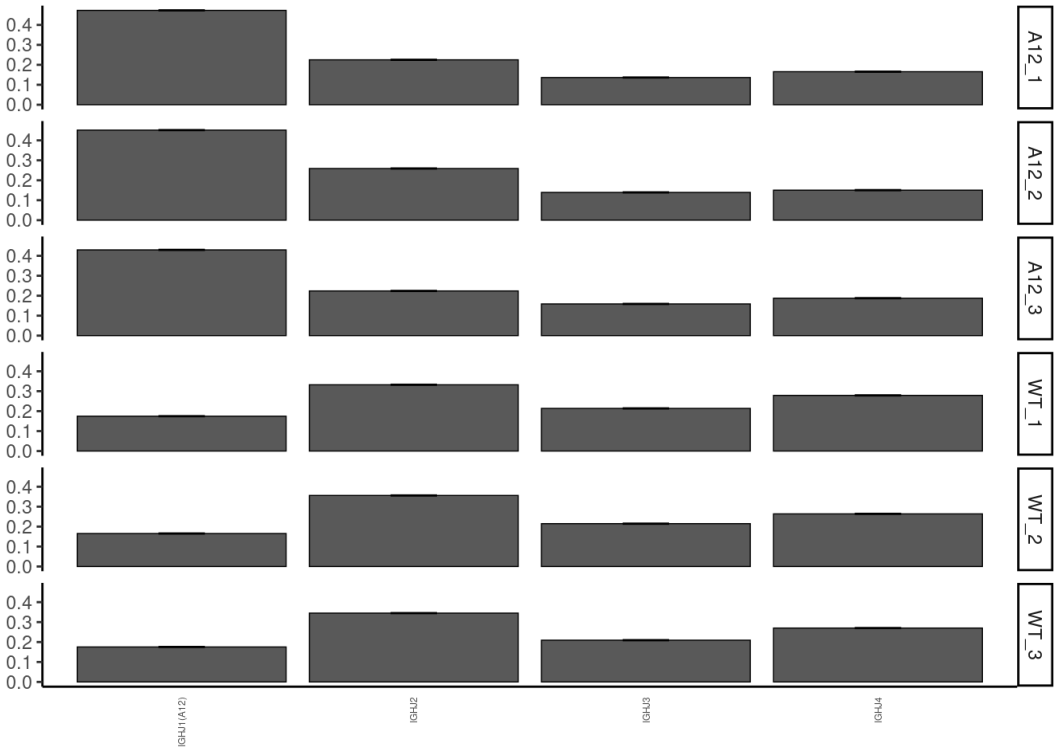
Sumarizing Repertoires for SP

```
vizGenes(combined_BCR_SP,
         x.axis = "IGHV",
         y.axis = NULL,
```

```
        plot = "barplot",
        scale = TRUE)
```

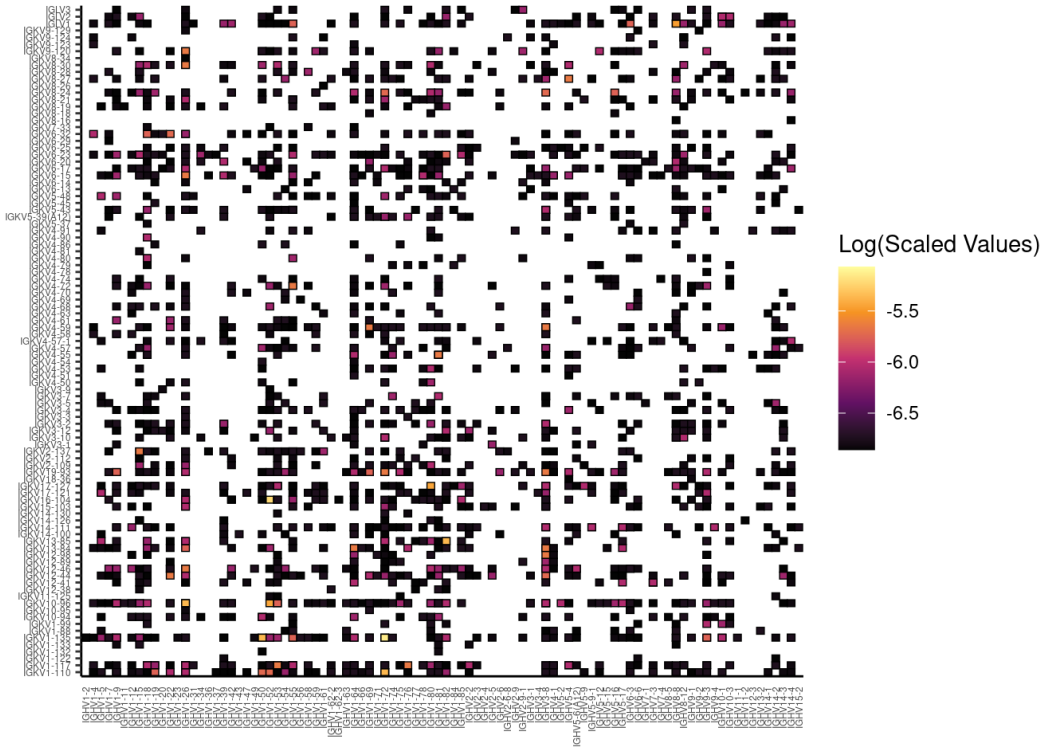



```
vizGenes(combined_BCR_SP,
        x.axis = "IGHJ",
        y.axis = NULL,
        plot = "barplot",
        scale = TRUE)
```

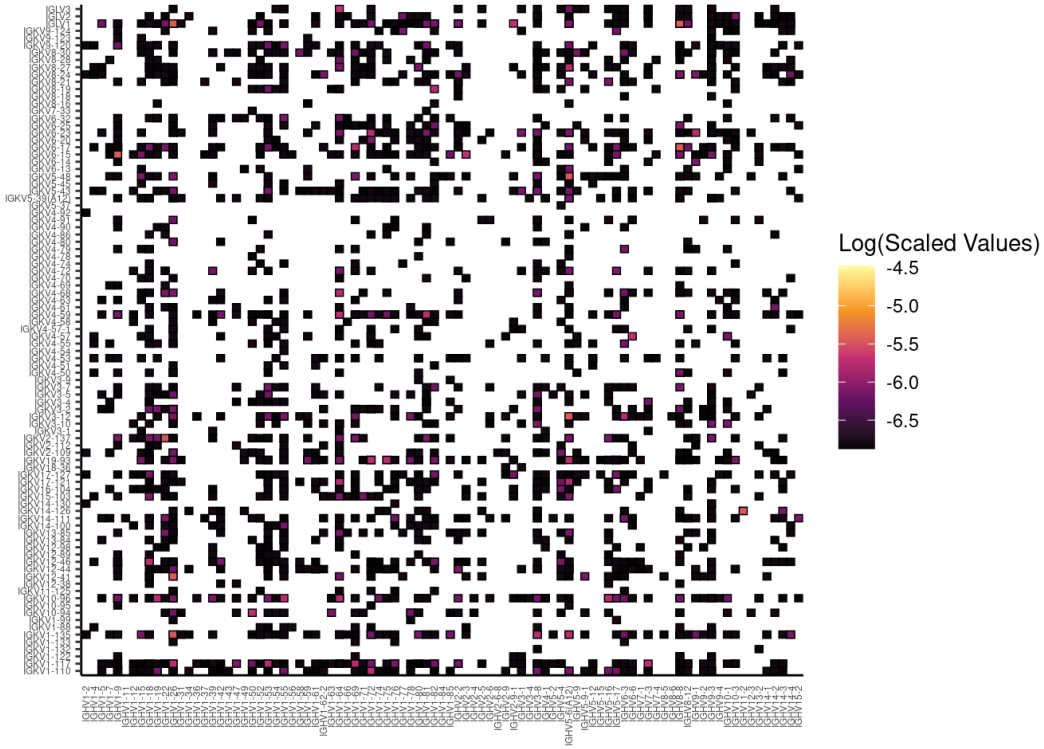



```
vizGenes(combined_BCR_SP[c("WT_1", "WT_2", "WT_3")],
        x.axis = "IGHV",
        y.axis = "IGKV",
```

```
          plot = "heatmap",
          scale = TRUE)
```



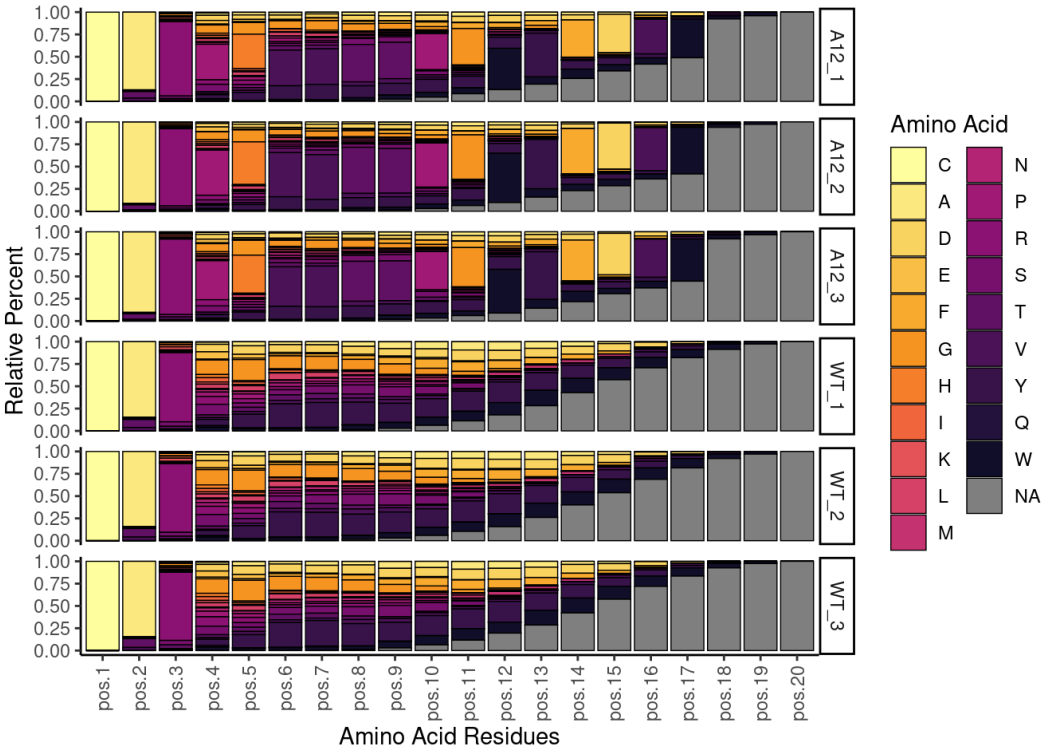

```
vizGenes(combined_BCR_SP[c("A12_1", "A12_2", "A12_3")],
          x.axis = "IGHV",
          y.axis = "IGKV",
          plot = "heatmap",
          scale = TRUE)
```



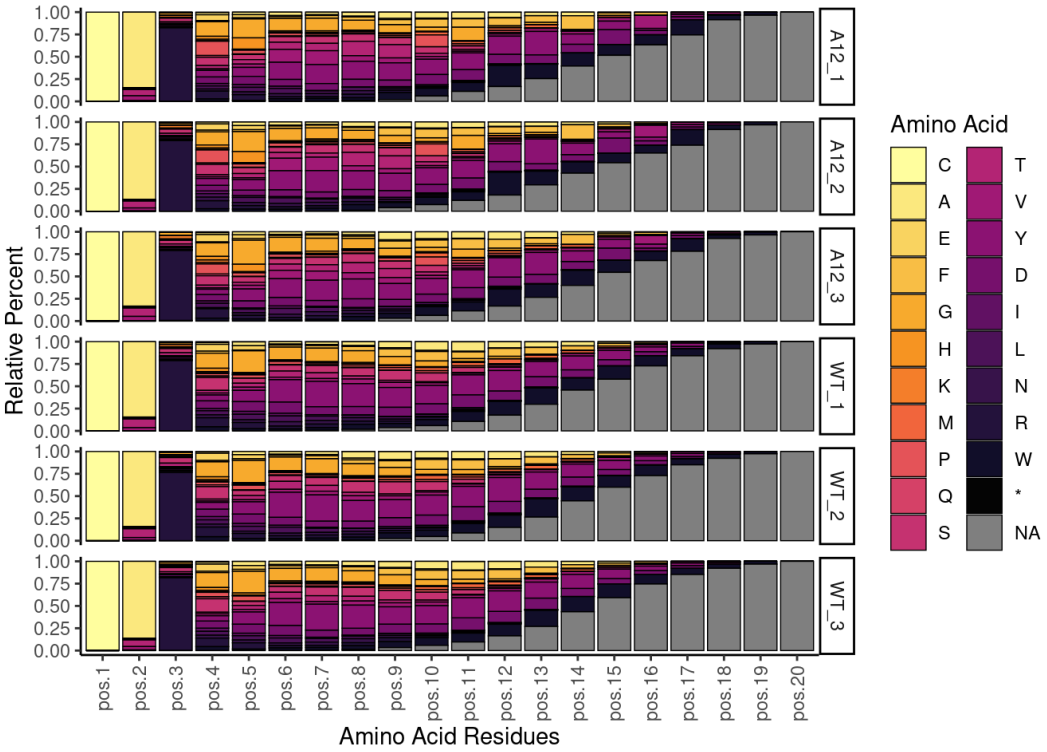
Percent aa

Quantify the proportion of amino acids along the cdr3 sequence with percentAA(). By default, the function will pad the sequences with NAs up to the maximum of aa.length. Sequences longer than aa.length will be removed before visualization (default aa.length = 20).

```
percentAA(combined_BCR_BM,
          chain = "IGH",
          aa.length = 20)
```

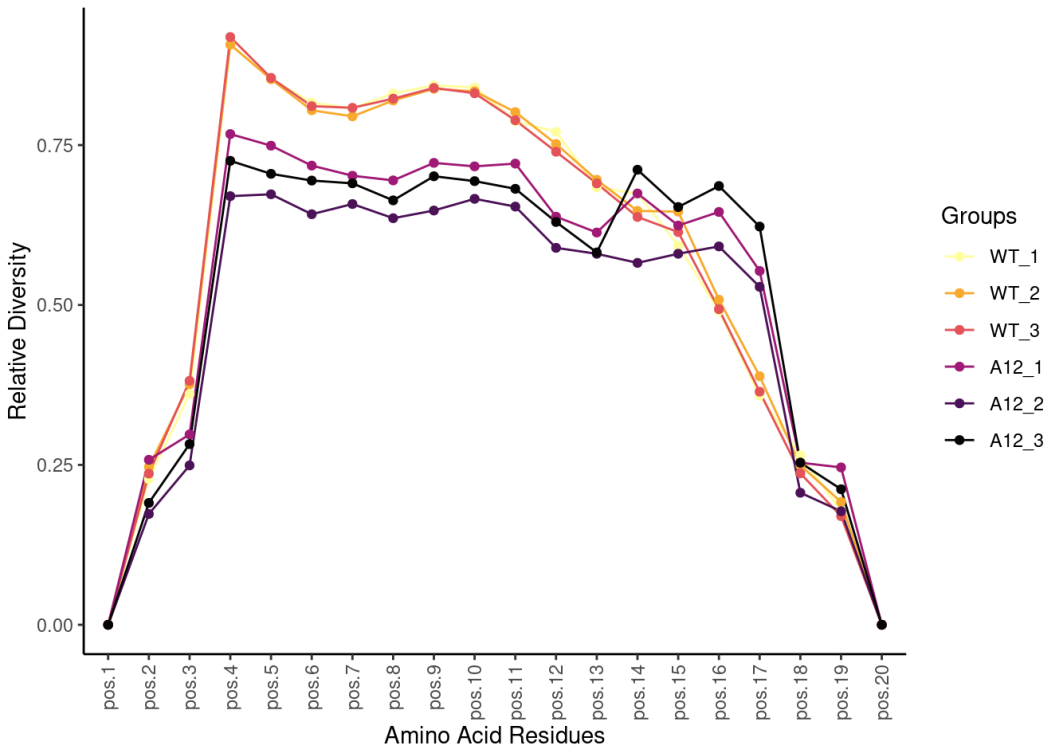



```
percentAA(combined_BCR_SP,
          chain = "IGH",
          aa.length = 20)
```

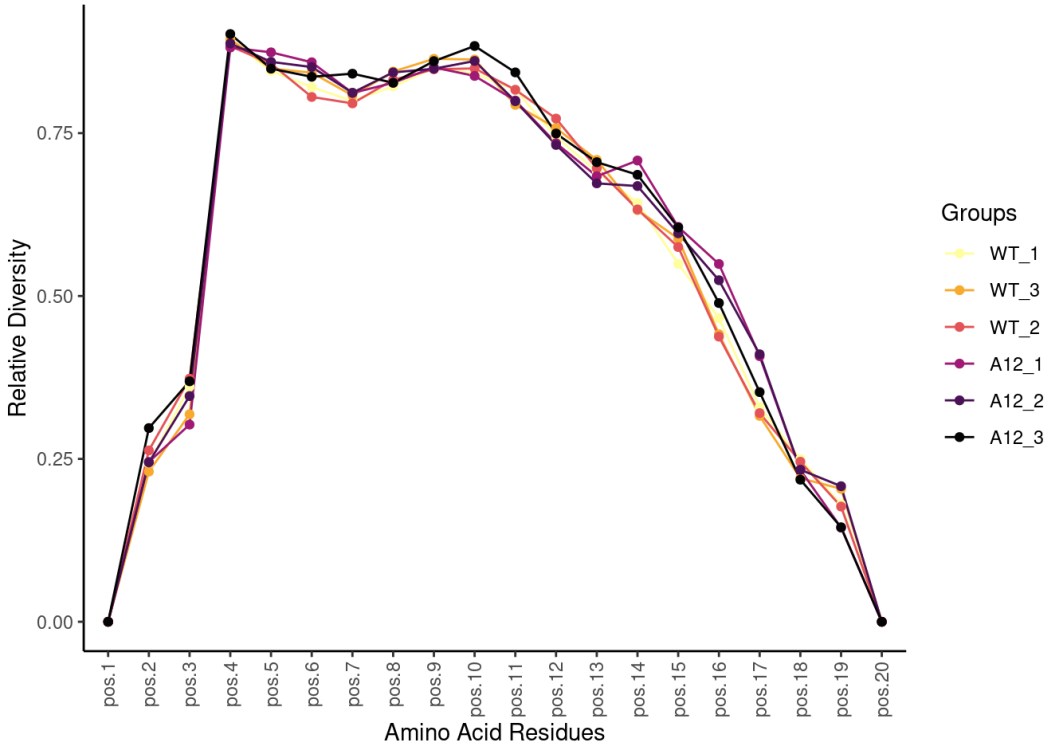



```
positionalEntropy(combined_BCR_BM,
```

```
                  chain = "IGH",
                  aa.length = 20)
```



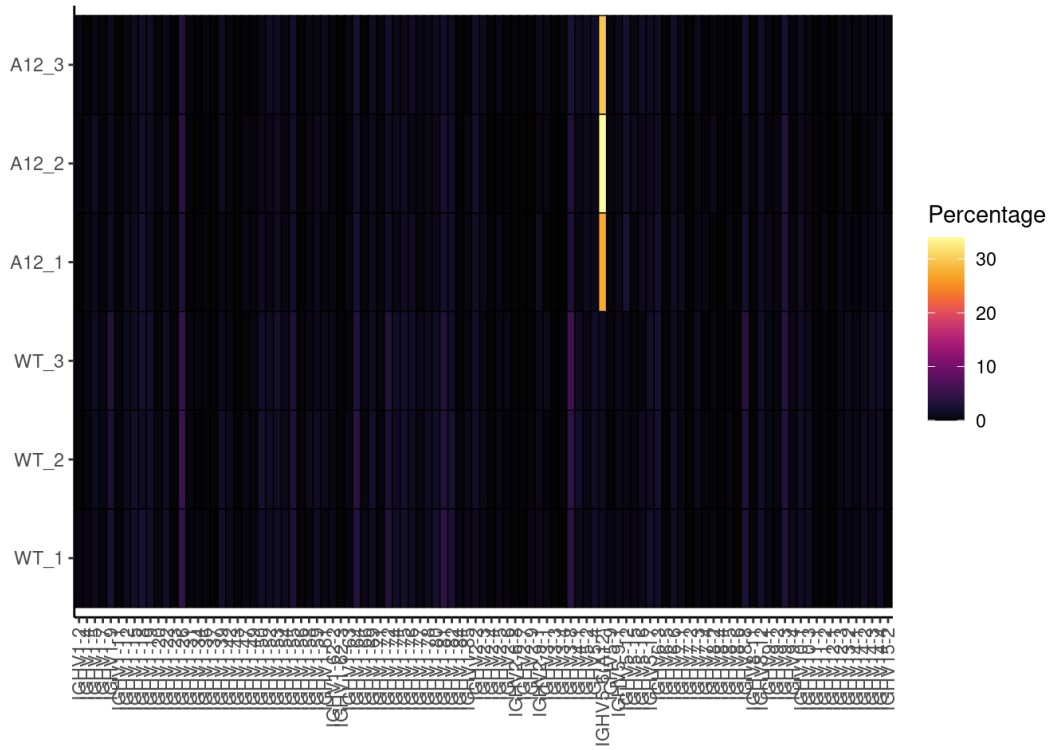

```
positionalEntropy(combined_BCR_SP,
                  chain = "IGH",
                  aa.length = 20)
```



Percent GENES BM

```
percentGenes(combined_BCR_BM,
```

```
            chain = "IGH",
            gene = "Vgene")
```



Combining clones with the Seurat Object

After processing the contig data into clones via combineBCR() or combineTCR(), we can add the clonal information to the single-cell object using combineExpression().

Importantly, the major requirement for the attachment is matching contig cell barcodes and barcodes in the row names of the meta data of the Seurat or Single-Cell Experiment object. If these do not match, the attachment will fail. We made changes to the single-cell object barcodes.

This function adds the immune receptor information to the Seurat or SCE object to the meta data. By default this function also calculates the frequencies and proportion of the clones by sequencing run (group.by = NULL). To change how the frequencies/proportions are calculated, select a column header for the group.by variable. Importantly, before using combineExpression ensure the barcodes of the single-cell object object match the barcodes in the output of the combineTCR or combineBCR.

First we must change names of barcodes naming them the same as in the combined_BCR object (mice_barcode)

```
Seurat_BM_2 <- Seurat_BM
Seurat_SP_2 <- Seurat_SP
```

Then we modify the meta.data of the seurat object

```
cell.barcodes <- rownames(Seurat_BM_2[[]])
# removing the _1 at the end of the barcodes)
cell.barcodes <- stringr::str_split(cell.barcodes, "_", simplify = TRUE)[, 1]
# adding the prefix of the orig.ident to the barcodes, assuming that is the sample ids
cell.barcodes <- paste0(Seurat_BM_2$mice, "_", cell.barcodes)
Seurat_BM_2 <- RenameCells(Seurat_BM_2, new.names = cell.barcodes)

cell.barcodes <- rownames(Seurat_SP_2[[]])
# removing the _1 at the end of the barcodes)
cell.barcodes <- stringr::str_split(cell.barcodes, "_", simplify = TRUE)[, 1]
# adding the prefix of the orig.ident to the barcodes, assuming that is the sample ids
cell.barcodes <- paste0(Seurat_SP_2$mice, "_", cell.barcodes)
Seurat_SP_2 <- RenameCells(Seurat_SP_2, new.names = cell.barcodes)
```

```
sce_BM <- combineExpression(combined_BCR_BM,
```

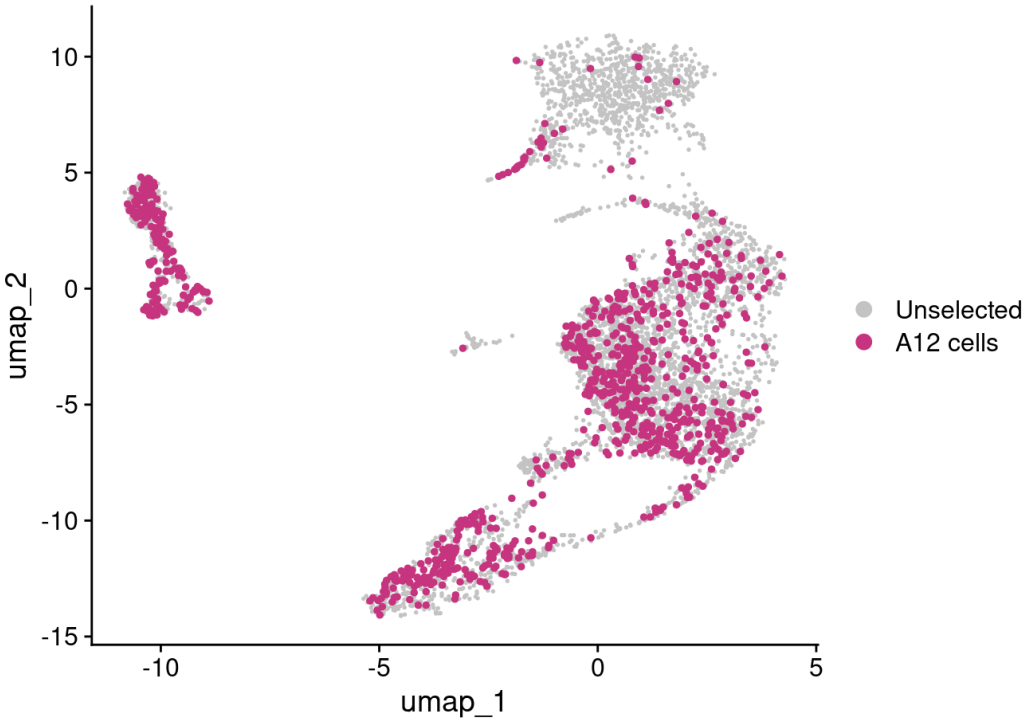
```
                          Seurat_BM_2,
                          cloneCall="aa",
                          chain = "IGH",
                          group.by = "sample",
                          filterNA = FALSE,
                          proportion = TRUE)

sce_SP <- combineExpression(combined_BCR_SP,
                          Seurat_SP_2,
                          cloneCall="aa",
                          chain = "IGH",
                          group.by = "sample",
                          filterNA = FALSE,
                          proportion = TRUE)
```

```
Seurat_BM_2_A12 <- subset(Seurat_BM_2, genotype == "A12")
Seurat_SP_2_A12 <- subset(Seurat_SP_2, genotype == "A12")
Seurat_BM_A12 <- subset(Seurat_BM, genotype == "A12")
Seurat_SP_A12 <- subset(Seurat_SP, genotype == "A12")
Seurat_BM_2_WT <- subset(Seurat_BM_2, genotype == "WT")
Seurat_SP_2_WT <- subset(Seurat_SP_2, genotype == "WT")
Seurat_BM_WT <- subset(Seurat_BM_2, genotype == "WT")
Seurat_SP_WT <- subset(Seurat_SP_2, genotype == "WT")
```

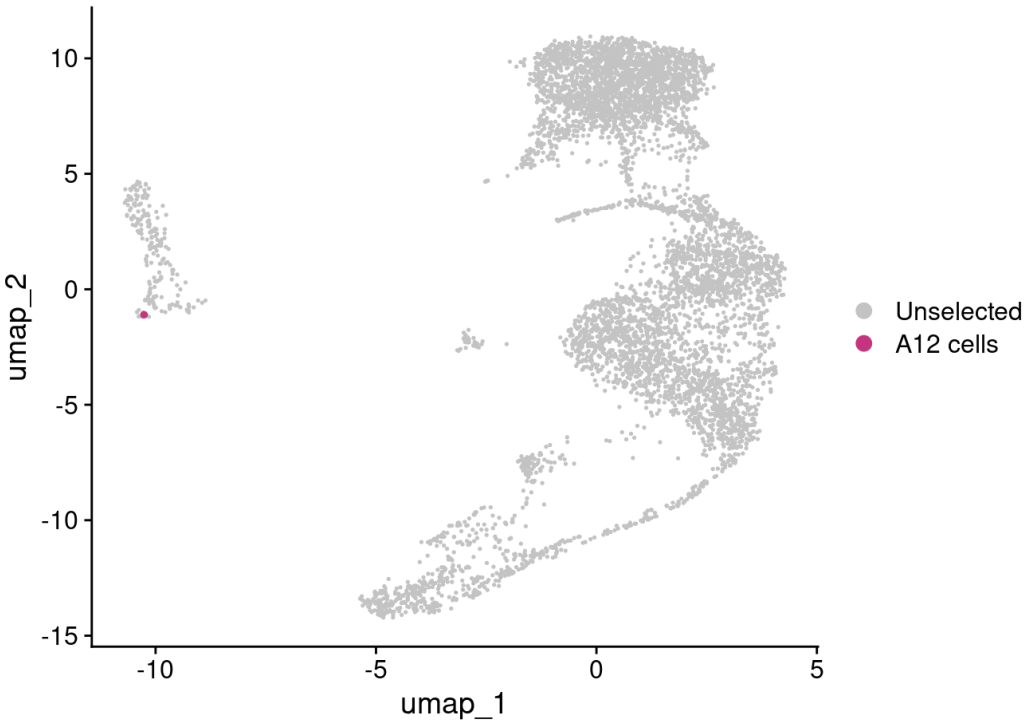
Print JH1 long CD3 cells and A12 cells etc

```
# Combine all A12 SP and BM mice in one dataframe
combined_BCR_BM_IGH_A12_MOUSE_MIX <- do.call(rbind, combined_BCR_BM_IGH[c("A12_1", "A12_2", "A12_3")])
combined_BCR_SP_IGH_A12_MOUSE_MIX <- do.call(rbind, combined_BCR_SP_IGH[c("A12_1", "A12_2", "A12_3")])


combined_BCR_BM_VHJ1 <- lapply(combined_BCR_BM_IGH, function(x) subset(x, VHJ1 == "TRUE"))
combined_BCR_SP_VHJ1 <- lapply(combined_BCR_SP_IGH, function(x) subset(x, VHJ1 == "TRUE"))

# This will contain endogenous Igh cells as we are filtering for JH2/3/4
combined_BCR_BM_VHJ2.3.4 <- lapply(combined_BCR_BM_IGH, function(x) subset(x, VHJ2 == "TRUE"|VHJ3 == "TRUE"|VHJ4 =
= "TRUE"))
combined_BCR_SP_VHJ2.3.4 <- lapply(combined_BCR_SP_IGH, function(x) subset(x, VHJ2 == "TRUE"|VHJ3 == "TRUE"|VHJ4 =
= "TRUE"))

# This contains cells with VHJ1
combined_BCR_BM_VHJ1 <- lapply(combined_BCR_BM_IGH, function(x) subset(x, VHJ1 == "TRUE"))

# Now we select for potential VH replaced cells as they have a long CDR3 and they are JH1
combined_BCR_BM_A12_long_CDR3 <- lapply(combined_BCR_BM_VHJ1, function(x) {
  subset(x, len_aa_CDR3 > 20 & len_aa_CDR3 < 30)
})

# We further select VH replaced cells originated in the A12 locus
combined_BCR_BM_A12_VHJ1_VHrep <- lapply(combined_BCR_BM_VHJ1, function(x) {
  subset(x, len_aa_CDR3 > 20 & len_aa_CDR3 < 30 &
           grepl("A12", x$sample) &
           grepl("^(?!IGHV5-6\\(A12\\)).*IGHDA12\\.IGHJ1\\(A12\\)", x$IGH, perl = TRUE))
})
combined_BCR_SP_A12_VHJ1_VHrep <- lapply(combined_BCR_SP_VHJ1, function(x) {
  subset(x, len_aa_CDR3 > 20 & len_aa_CDR3 < 30 &
           grepl("A12", x$sample) &
           grepl("^(?!IGHV5-6\\(A12\\)).*IGHDA12\\.IGHJ1\\(A12\\)", x$IGH, perl = TRUE))
})

# Subset to separate between genotypes the previous groups
combined_BCR_BM_VHJ2.3.4_A12 <- lapply(combined_BCR_BM_VHJ2.3.4, function(x) subset(x, grepl("A12", x$sample)))
combined_BCR_SP_VHJ2.3.4_A12 <- lapply(combined_BCR_SP_VHJ2.3.4, function(x) subset(x, grepl("A12", x$sample)))
combined_BCR_BM_VHJ2.3.4_WT <- lapply(combined_BCR_BM_VHJ2.3.4, function(x) subset(x, grepl("WT", x$sample)))
combined_BCR_BM_VHJ1_WT <- lapply(combined_BCR_BM_VHJ1, function(x) subset(x, grepl("WT", x$sample)))
```

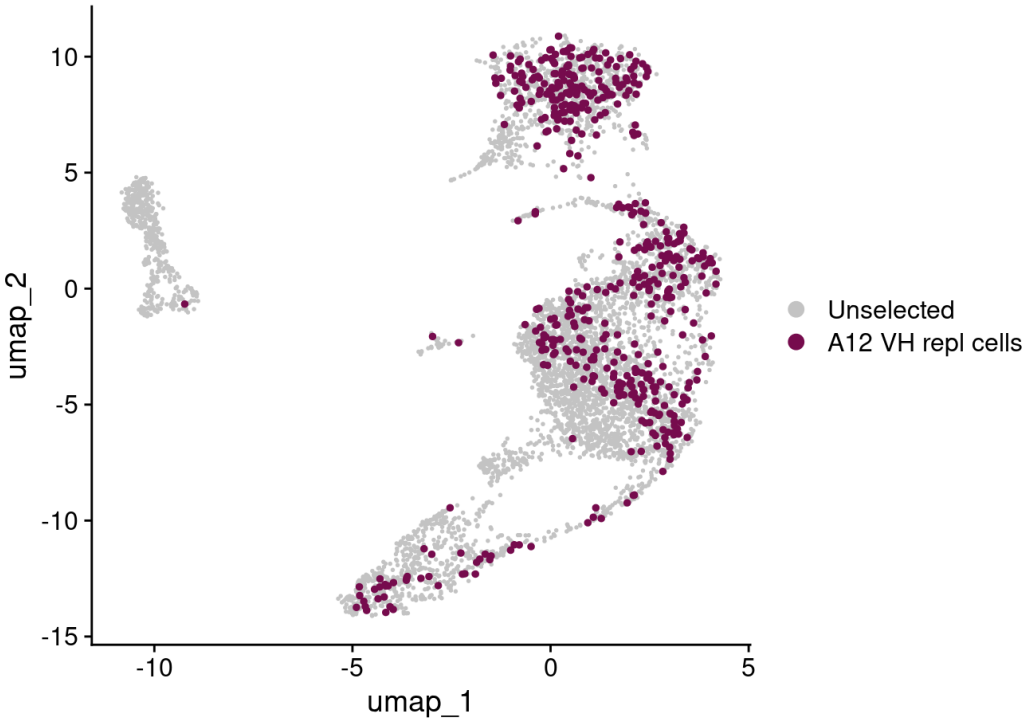
```
combined_BCR_BM_A12_long_CDR3_A12 <- lapply(combined_BCR_BM_A12_long_CDR3, function(x) subset(x, grepl("A12", x$sa
mple)))

# Transform to dataframe in order to have all mice in the same dataframe
combined_BCR_BM_MOUSE_MIX <- do.call(rbind, combined_BCR_BM)
combined_BCR_SP_MOUSE_MIX <- do.call(rbind, combined_BCR_SP)
combined_BCR_BM_A12_MOUSE_MIX <- do.call(rbind, combined_BCR_BM_A12)
combined_BCR_SP_A12_MOUSE_MIX <- do.call(rbind, combined_BCR_SP_A12)
combined_BCR_BM_A12_VHJ1_VHrep_MOUSE_MIX <- do.call(rbind, combined_BCR_BM_A12_VHJ1_VHrep)
combined_BCR_SP_A12_VHJ1_VHrep_MOUSE_MIX <- do.call(rbind, combined_BCR_SP_A12_VHJ1_VHrep)
combined_BCR_BM_A12_VHJ2.3.4_MOUSE_MIX_A12 <- do.call(rbind, combined_BCR_BM_VHJ2.3.4_A12)
combined_BCR_BM_A12_VHJ2.3.4_MOUSE_MIX_WT <- do.call(rbind, combined_BCR_BM_VHJ2.3.4_WT)
combined_BCR_SP_A12_VHJ2.3.4_MOUSE_MIX_A12 <- do.call(rbind, combined_BCR_SP_VHJ2.3.4_A12)
combined_BCR_BM_A12_VHJ1_MOUSE_MIX_WT <- do.call(rbind, combined_BCR_BM_VHJ1_WT)
combined_BCR_BM_A12_long_CDR3_MOUSE_MIX_A12 <- do.call(rbind, combined_BCR_BM_A12_long_CDR3_A12)

# Plot A12 cells, VH-replaced cells and endogenous Igh cells. Also check there are no A12 cells in WT

DimPlot(Seurat_BM_2_A12, reduction = "umap", cells.highlight = list("A12 cells" = combined_BCR_BM_IGH_A12_MOUSE_MI
X$barcode[grepl("IGHV5-6\\(A12\\)\\.IGHDA12\\.IGHJ1\\(A12\\)", combined_BCR_BM_IGH_A12_MOUSE_MIX$IGH)]), cols.high
light = "#C6347F")
```



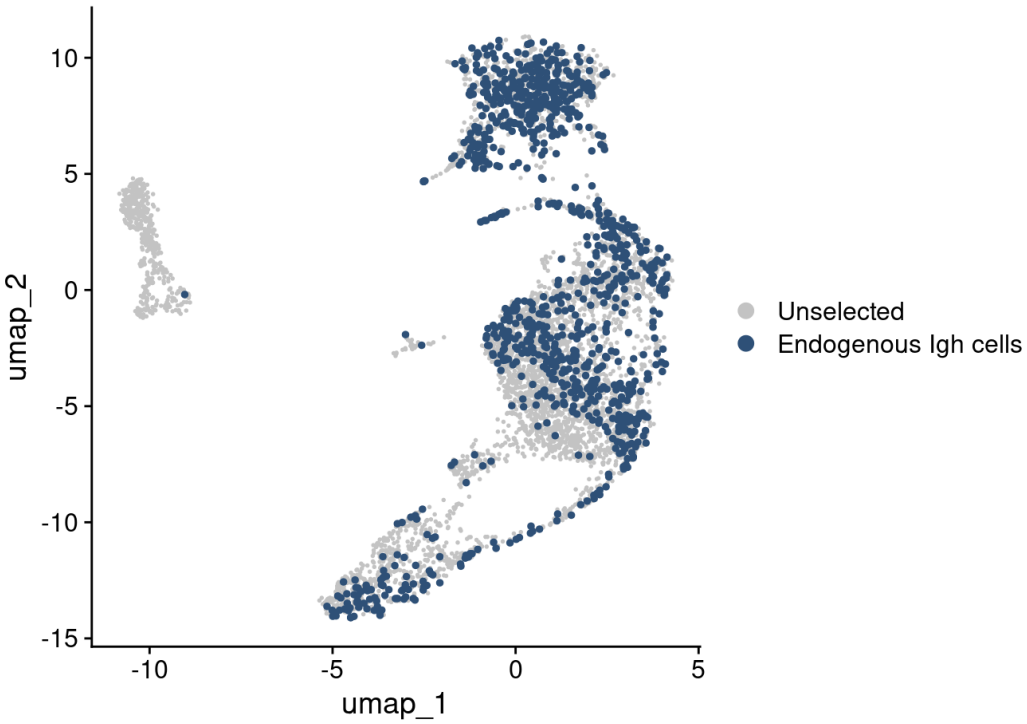

```
DimPlot(Seurat_BM_2_WT, reduction = "umap", cells.highlight = list("A12 cells" = combined_BCR_BM_WT_MOUSE_MIX$barc
ode[grepl("IGHV5-6\\(A12\\)\\.IGHDA12\\.IGHJ1\\(A12\\)", combined_BCR_BM_WT_MOUSE_MIX$IGH)]), cols.highlight = "#C
6347F")
```



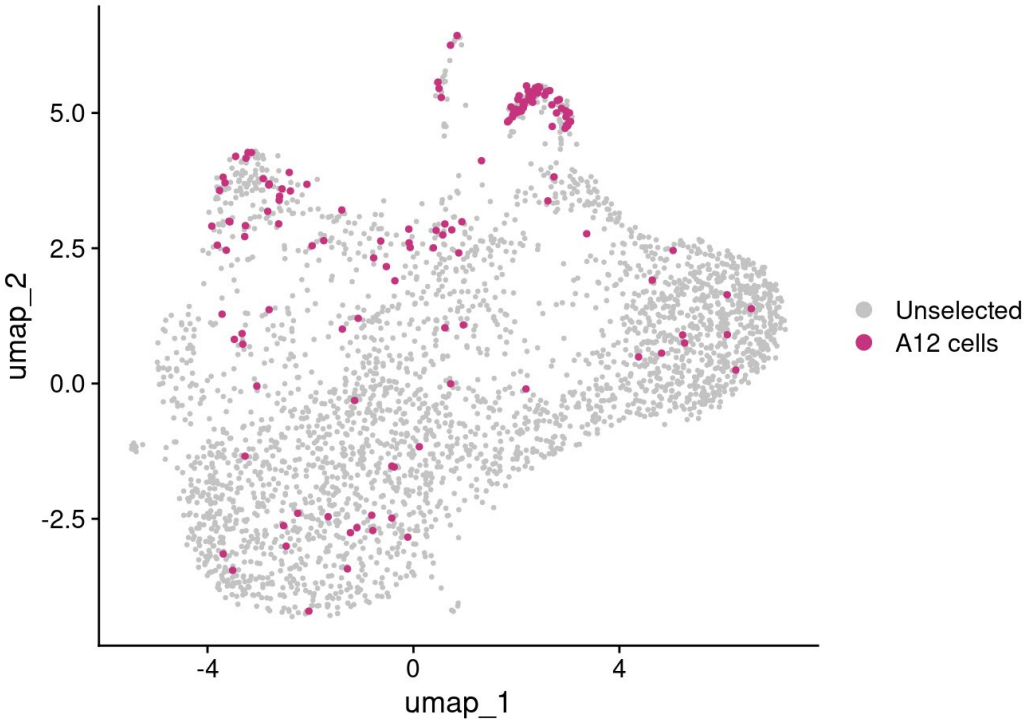

```
DimPlot(Seurat_BM_2_A12, reduction = "umap", cells.highlight = list("A12 VH repl cells" = combined_BCR_BM_A12_VHJ1_VHrep_MOUSE_MIX$barcode), cols.highlight = "#760B4D")
```



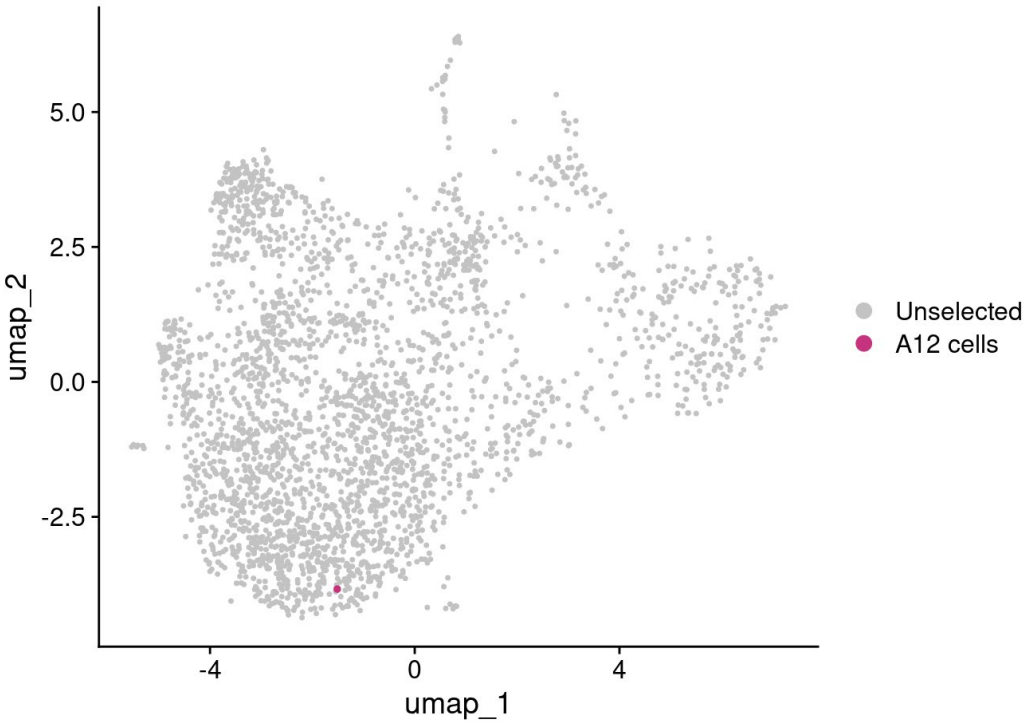

```
DimPlot(Seurat_BM_2_A12, reduction = "umap", cells.highlight = list("Endogenous Igh cells" = combined_BCR_BM_A12_VHJ2.3.4_MOUSE_MIX_A12$barcode), cols.highlight = "#2E5077")
```



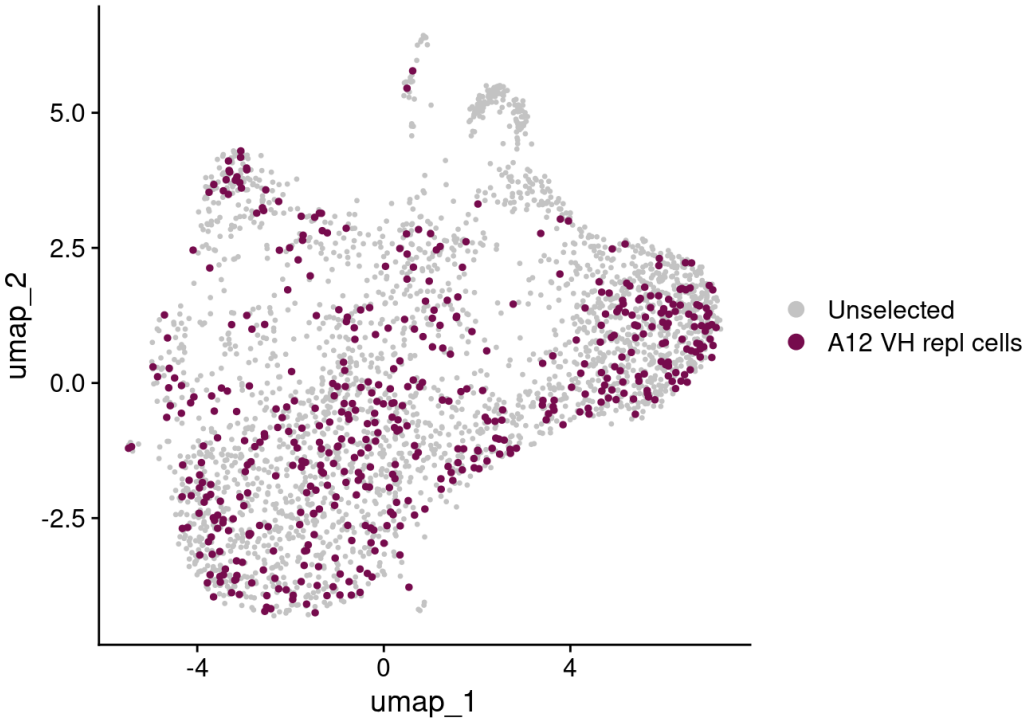

```
DimPlot(Seurat_SP_2_A12, reduction = "umap", cells.highlight = list("A12 cells" = combined_BCR_SP_IGH_A12_MOUSE_MIX$barcode[grepl("IGHV5-6\\(A12\\)\\.IGHDA12\\.IGHJ1\\(A12\\)", combined_BCR_SP_IGH_A12_MOUSE_MIX$IGH)]), cols.highlight = "#C6347F")
```



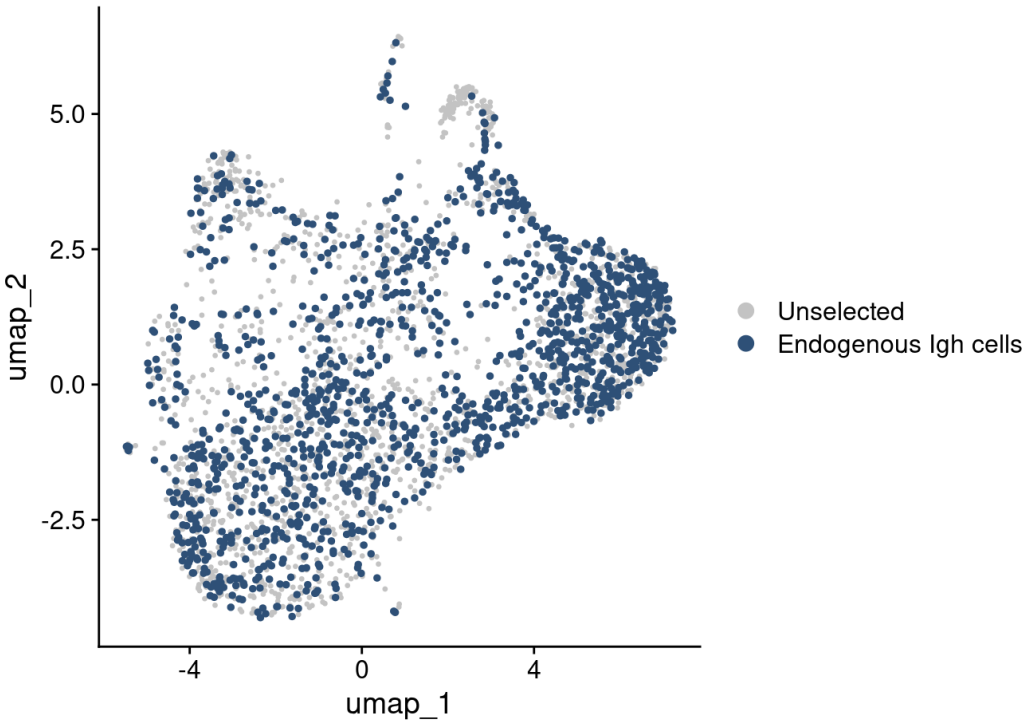

```
DimPlot(Seurat_SP_2_WT, reduction = "umap", cells.highlight = list("A12 cells" = combined_BCR_SP_WT_MOUSE_MIX$barcode[grepl("IGHV5-6\\(A12\\)\\.IGHDA12\\.IGHJ1\\(A12\\)", combined_BCR_SP_WT_MOUSE_MIX$IGH)]), cols.highlight = "#C6347F")
```



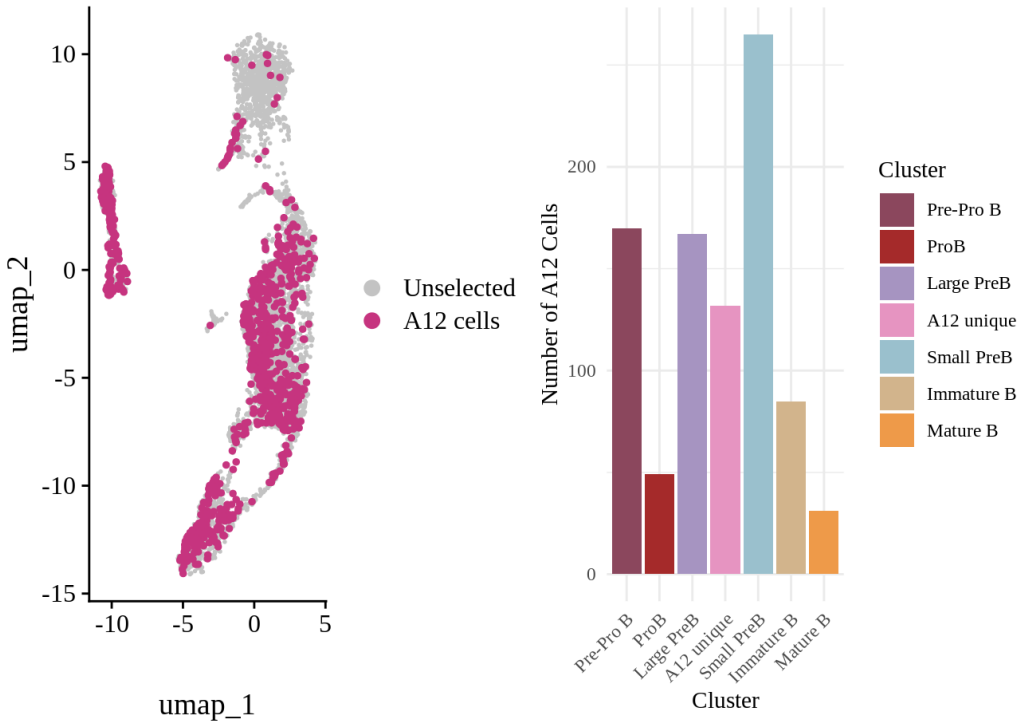

```
DimPlot(Seurat_SP_2_A12, reduction = "umap", cells.highlight = list("A12 VH repl cells" = combined_BCR_SP_A12_VHJ1
_VHrep_MOUSE_MIX$barcode), cols.highlight = "#760B4D")
```




```
DimPlot(Seurat_SP_2_A12, reduction = "umap", cells.highlight = list("Endogenous Igh cells" = combined_BCR_SP_A12_V
HJ2.3.4_MOUSE_MIX_A12$barcode), cols.highlight = "#2E5077")
```




```
# Plot A12 bone marrow umap plot with A12-expressing cells + barplot with cell counts

A12_cells <- combined_BCR_BM_IGH_A12_MOUSE_MIX$barcode[grepl("IGHV5-6\\(A12\\)\\.IGHDA12\\.IGHJ1\\(A12\\)", combin
ed_BCR_BM_IGH_A12_MOUSE_MIX$IGH)]


p1 <- DimPlot(Seurat_BM_2_A12, reduction = "umap",
              cells.highlight = list("A12 cells" = A12_cells),
              cols.highlight = "#C6347F") +
  theme(text = element_text(family = "Times New Roman"))

A12_cells_in_clusters <- Seurat_BM_2_A12$simplified_clusters[row.names(Seurat_BM_2_A12@meta.data) %in% A12_cells]

cluster_counts <- table(A12_cells_in_clusters)

cluster_df <- as.data.frame(cluster_counts)
colnames(cluster_df) <- c("Cluster", "CellCount")

cluster_colors <- c("Immature B" = "tan",
                    "Mature B" = "tan2",
                    "A12 unique" = "#E694C1",
                    "Small PreB" = "lightblue3",
                    "ProB" = "#A52A2A",
                    "Pre-Pro B" = "palevioletred4",
                    "Large PreB" = "#A694C1")

p2 <- ggplot(cluster_df, aes(x = Cluster, y = CellCount, fill = Cluster)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = cluster_colors) +
  theme_minimal() +
  labs(x = "Cluster", y = "Number of A12 Cells") +
  theme(text = element_text(family = "Times New Roman"),
        axis.text.x = element_text(angle = 45, hjust = 1))

p1 + p2
```




```
Cells_end_Igh <- combined_BCR_BM_A12_VHJ2.3.4_MOUSE_MIX_A12$barcode
Cells_VH_rep <- combined_BCR_BM_A12_VHJ1_VHrep_MOUSE_MIX$barcode
Cells_A12_vdj <- combined_BCR_BM_IGH_A12_MOUSE_MIX$barcode[grepl("IGHV5-6\\(A12\\)\\.IGHDA12\\.IGHJ1\\(A12\\)", co
mbined_BCR_BM_IGH_A12_MOUSE_MIX$IGH)]
```

Proportion of Cells VH rep, A12, endogenous etc

```
combined_BCR_BM_IGH_A12_MOUSE_MIX <- do.call(rbind, combined_BCR_BM_IGH[c("A12_1", "A12_2", "A12_3")])
combined_BCR_SP_IGH_A12_MOUSE_MIX <- do.call(rbind, combined_BCR_SP_IGH[c("A12_1", "A12_2", "A12_3")])
```

```
N_endogenous_Igh <- length(combined_BCR_BM_A12_VHJ2.3.4_MOUSE_MIX_A12$barcode)
N_A12_VHreplacement <- length(combined_BCR_BM_A12_VHJ1_VHrep_MOUSE_MIX$barcode)
N_A12 <- length(combined_BCR_BM_IGH_A12_MOUSE_MIX$barcode[grepl("IGHV5-6\\(A12\\)\\.IGHDA12\\.IGHJ1\\(A12\\)", com
bined_BCR_BM_IGH_A12_MOUSE_MIX$IGH)])

total <- sum(N_endogenous_Igh, N_A12, N_A12_VHreplacement)

N_endogenous_Igh/total
```

```
## [1] 0.4157209
```

```
N_A12_VHreplacement/total
```

```
## [1] 0.2059689
```

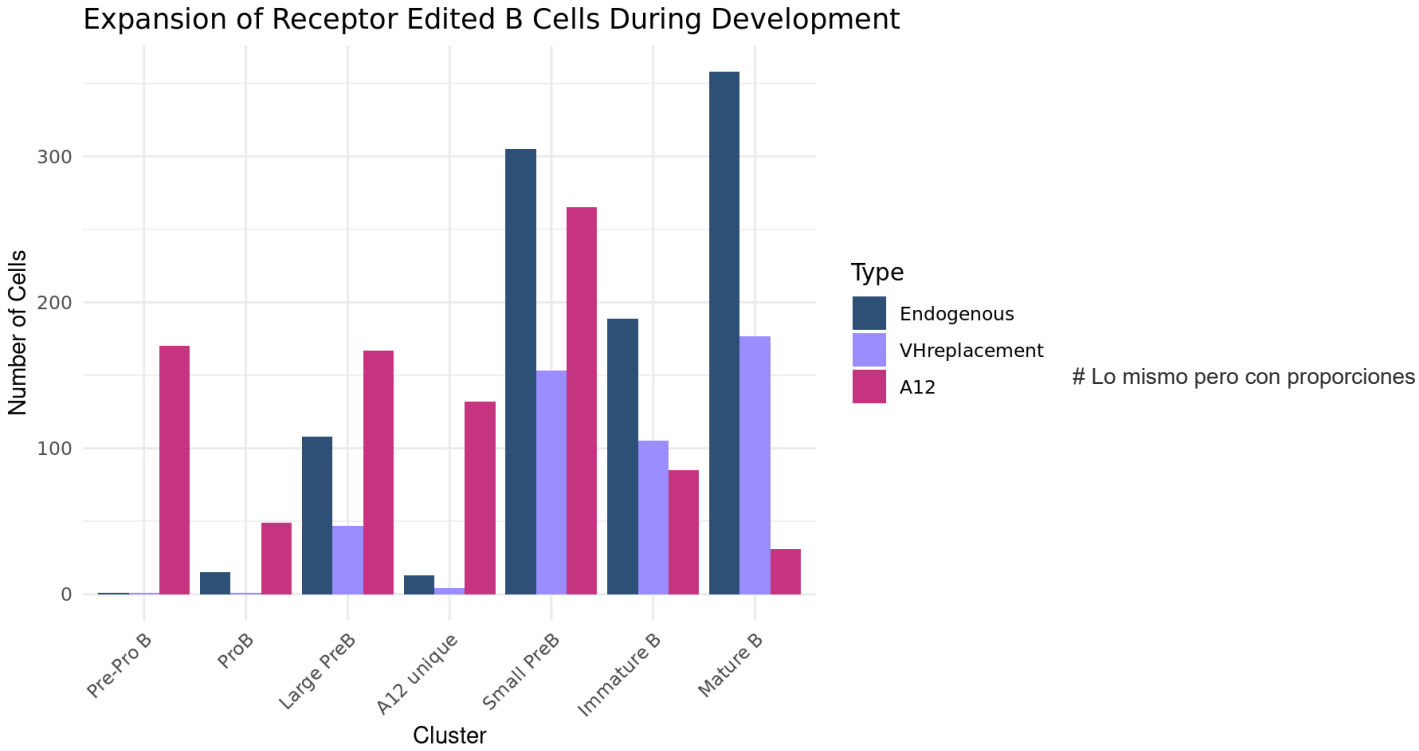
```
N_A12/total
```

```
## [1] 0.3783102
```

Cuantify VDJ of each type (A12-expressing, VH-replaced, endogenous Igh) in bone marrow

```
# Step 1: Identify the cells of each type

# Endogenous cells
endogenous_cells <- combined_BCR_BM_A12_VHJ2.3.4_MOUSE_MIX_A12$barcode

# Cells with VH replacement
VHreplacement_cells <- combined_BCR_BM_A12_VHJ1_VHrep_MOUSE_MIX$barcode

# Cells A12-expressing
A12_cells <- combined_BCR_BM_IGH_A12_MOUSE_MIX$barcode[
    grepl("IGHV5-6\\(A12\\)\\.IGHDA12\\.IGHJ1\\(A12\\)",
          combined_BCR_BM_IGH_A12_MOUSE_MIX$IGH)
]

# Step 2: Assign the cells to their clusters

# Endogenous cells in clusters
endogenous_clusters <- Seurat_BM_2@meta.data$simplified_clusters[
  rownames(Seurat_BM_2@meta.data) %in% endogenous_cells
]

# Cells with VH replacement in clusters
VHreplacement_clusters <- Seurat_BM_2@meta.data$simplified_clusters[
  rownames(Seurat_BM_2@meta.data) %in% VHreplacement_cells
]

# Cells with A12-specific IGH in clusters
A12_clusters <- Seurat_BM_2@meta.data$simplified_clusters[
  rownames(Seurat_BM_2@meta.data) %in% A12_cells
]

# Step 3: Count the number of cells of each type in each cluster

# Count endogenous cells in clusters
endogenous_counts <- table(endogenous_clusters)
# Count cells with VH replacement in clusters
VHreplacement_counts <- table(VHreplacement_clusters)

# Count cells with A12-specific IGH in clusters
A12_counts <- table(A12_clusters)

# Step 4: Combine the counts into a dataframe

# Create a data frame with clusters and counts
combined_counts <- data.frame(
  Cluster = names(endogenous_counts),
  Endogenous = as.integer(endogenous_counts),
  VHreplacement = as.integer(VHreplacement_counts[match(names(endogenous_counts), names(VHreplacement_counts))]),
  A12 = as.integer(A12_counts[match(names(endogenous_counts), names(A12_counts))])
)

# Replace NAs with zeros if a cluster has no cells of a certain type
combined_counts[is.na(combined_counts)] <- 0

library(reshape2)
combined_counts_long <- melt(combined_counts, id.vars = "Cluster", variable.name = "Cell_Type", value.name = "Coun
t")
combined_counts_long$Cluster <- factor(combined_counts_long$Cluster, levels = c("Pre-Pro B", "ProB", "Large PreB",
 "A12 unique",
                    "Small PreB", "Immature B", "Mature B"))

# Step 5: Create the stacked bar plot

# Plot
ggplot(combined_counts_long, aes(x = Cluster, y = Count, fill = Cell_Type)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Expansion of Receptor Edited B Cells During Development",
```

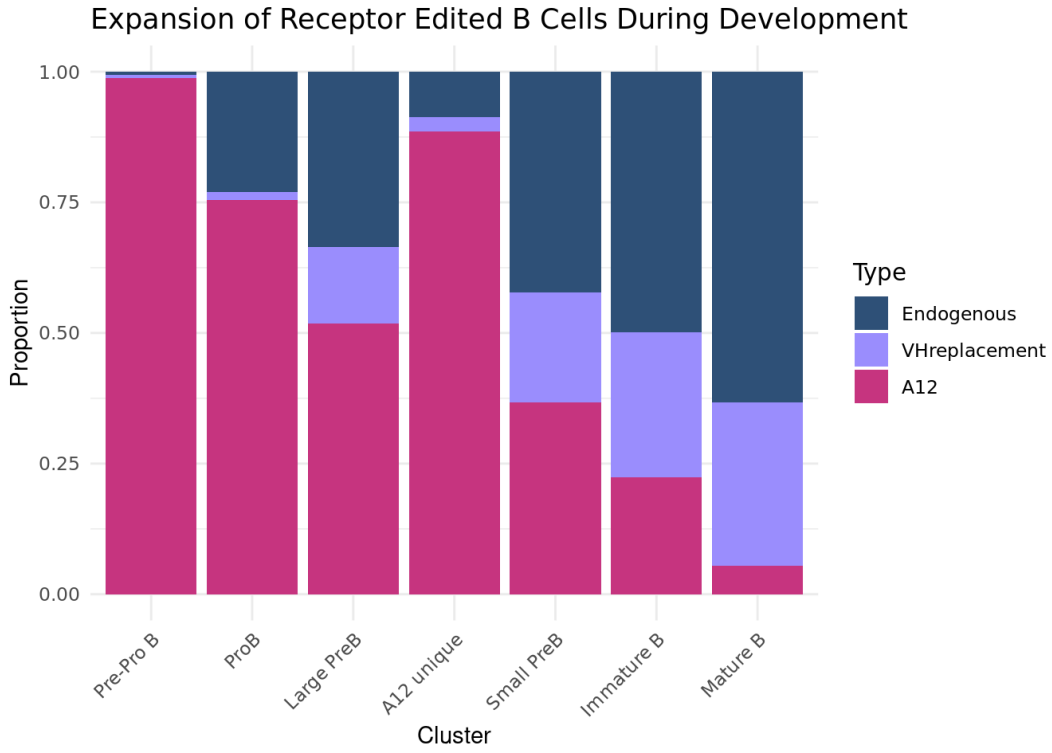
```
      x = "Cluster",
      y = "Number of Cells",
      fill = "Type") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, family = "Calibri"),
        axis.text.y = element_text(family = "Calibri"),
        plot.title = element_text(family = "Calibri"),
        legend.title = element_text(family = "Calibri"),
        legend.text = element_text(family = "Calibri")) +
  scale_fill_manual(values = c("#2E5077", "#9A8DFD", "#C6347F"))
```



Lo mismo pero con proporciones

```
# Step 3: Count the number of cells of each type in each cluster

# Count endogenous cells in clusters
endogenous_counts <- table(endogenous_clusters)
# Count cells with VH replacement in clusters
VHreplacement_counts <- table(VHreplacement_clusters)
# Count cells with IGH A12 in clusters
A12_counts <- table(A12_clusters)

# Step 4: Combine the counts into a dataframe

# Create a data frame with clusters and counts
combined_counts <- data.frame(
  Cluster = names(endogenous_counts),
  Endogenous = as.integer(endogenous_counts),
  VHreplacement = as.integer(VHreplacement_counts[match(names(endogenous_counts), names(VHreplacement_counts))]),
  A12 = as.integer(A12_counts[match(names(endogenous_counts), names(A12_counts))])
)

# Replace NAs with zeros if a cluster has no cells of a certain type
combined_counts[is.na(combined_counts)] <- 0

# Step 5: Calculate proportions

# Calculate the total number of cells in each cluster
combined_counts$Total <- rowSums(combined_counts[, c("Endogenous", "VHreplacement", "A12")])

# Convert counts to proportions
combined_counts$Endogenous <- combined_counts$Endogenous / combined_counts$Total
```

```
combined_counts$VHreplacement <- combined_counts$VHreplacement / combined_counts$Total
combined_counts$A12 <- combined_counts$A12 / combined_counts$Total

# Step 6: Convert the dataframe to long format for ggplot

combined_counts_long <- melt(combined_counts[, -which(names(combined_counts) == "Total")],
                             id.vars = "Cluster",
                             variable.name = "Cell_Type",
                             value.name = "Proportion")

# Ensure the correct order of clusters in the plot
combined_counts_long$Cluster <- factor(combined_counts_long$Cluster,
                                       levels = c("Pre-Pro B", "ProB", "Large PreB", "A12 unique",
                                                  "Small PreB", "Immature B", "Mature B"))

# Step 7: Create a stacked bar plot with proportions

# Proportion plot
ggplot(combined_counts_long, aes(x = Cluster, y = Proportion, fill = Cell_Type)) +
  geom_bar(stat = "identity", position = "fill") +  # 'position = "fill"' for proportions
  labs(title = "Expansion of Receptor Edited B Cells During Development",
       x = "Cluster",
       y = "Proportion",
       fill = "Type") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, family = "Calibri"),
        axis.text.y = element_text(family = "Calibri"),
        plot.title = element_text(family = "Calibri"),
        legend.title = element_text(family = "Calibri"),
        legend.text = element_text(family = "Calibri")) +
  scale_fill_manual(values = c("#2E5077", "#9A8DFD", "#C6347F"))
```

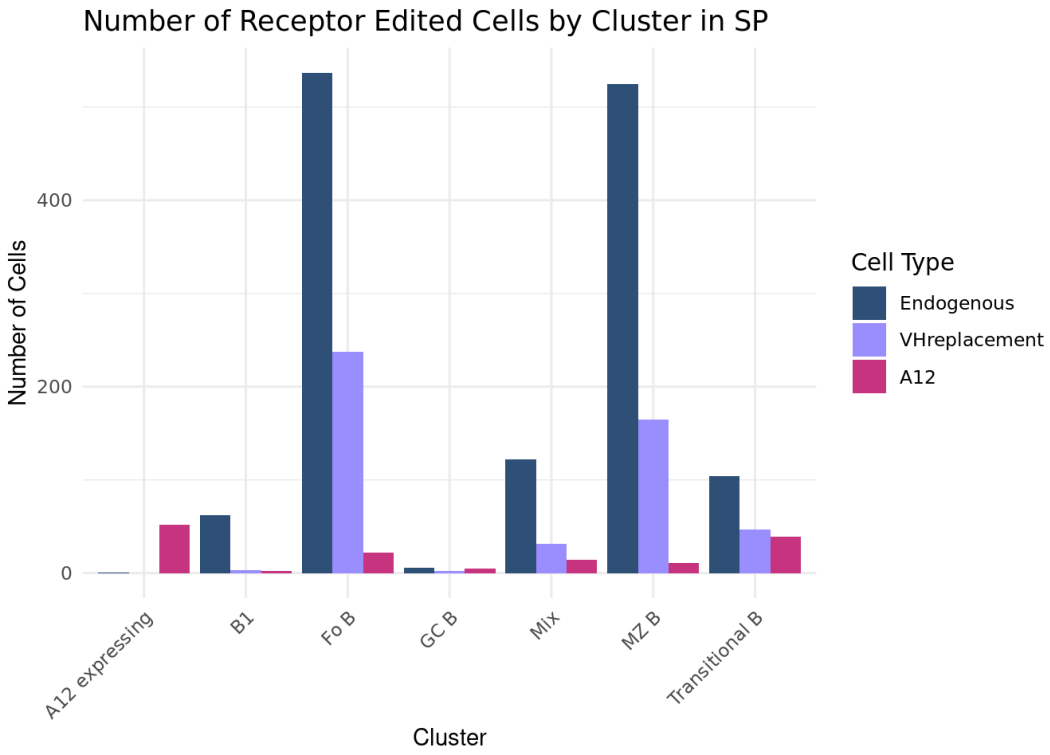


Cuantify VDJ of each type (A12-expressing, VH-replaced, endogenous Igh) in spleen

```
# Step 1: Identify the cells of each type
```

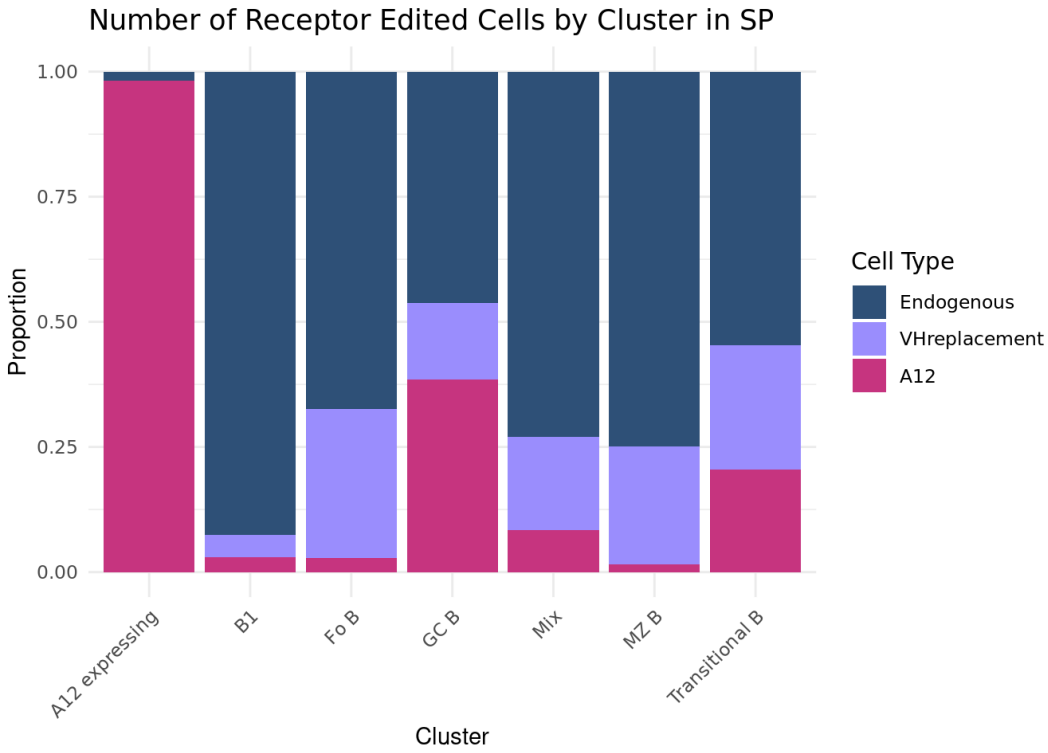
```
# Endogenous cells
endogenous_cells <- combined_BCR_SP_A12_VHJ2.3.4_MOUSE_MIX_A12$barcode

# Cells with VH replacement
VHreplacement_cells <- combined_BCR_SP_A12_VHJ1_VHrep_MOUSE_MIX$barcode

# Cells with A12-expressing
A12_cells <- combined_BCR_SP_IGH_A12_MOUSE_MIX$barcode[
    grepl("IGHV5-6\\(A12\\)\\.IGHDA12\\.IGHJ1\\(A12\\)",
          combined_BCR_SP_IGH_A12_MOUSE_MIX$IGH)
]

# Step 2: Assign cells to their clusters

# Endogenous cells in clusters
endogenous_clusters <- Seurat_SP_2@meta.data$seurat_clusters_new_fin[
  rownames(Seurat_SP_2@meta.data) %in% endogenous_cells
]

# Cells with VH replacement in clusters
VHreplacement_clusters <- Seurat_SP_2@meta.data$seurat_clusters_new_fin[
  rownames(Seurat_SP_2@meta.data) %in% VHreplacement_cells
]

# Cells with IGH A12 in clusters
A12_clusters <- Seurat_SP_2@meta.data$seurat_clusters_new_fin[
  rownames(Seurat_SP_2@meta.data) %in% A12_cells
]

# Step 3: Count the number of cells of each type in each cluster

# Count endogenous cells in clusters
endogenous_counts <- table(endogenous_clusters)
# Count cells with VH replacement in clusters
VHreplacement_counts <- table(VHreplacement_clusters)

# Count cells with IGH A12 in clusters
A12_counts <- table(A12_clusters)

# Step 4: Combine the counts into a dataframe

# Create a data frame with clusters and counts
combined_counts <- data.frame(
  Cluster = names(endogenous_counts),
  Endogenous = as.integer(endogenous_counts),
  VHreplacement = as.integer(VHreplacement_counts[match(names(endogenous_counts), names(VHreplacement_counts))]),
  A12 = as.integer(A12_counts[match(names(endogenous_counts), names(A12_counts))])
)

# Replace NAs with zeros if a cluster has no cells of a certain type
combined_counts[is.na(combined_counts)] <- 0

library(reshape2)
combined_counts_long <- melt(combined_counts, id.vars = "Cluster", variable.name = "Cell_Type", value.name = "Coun
t")

# Step 5: Create the stacked bar plot

# Plot
ggplot(combined_counts_long, aes(x = Cluster, y = Count, fill = Cell_Type)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Number of Receptor Edited Cells by Cluster in SP",
       x = "Cluster",
       y = "Number of Cells",
       fill = "Cell Type") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, family = "Calibri"),
```

```
        axis.text.y = element_text(family = "Calibri"),
        plot.title = element_text(family = "Calibri"),
        legend.title = element_text(family = "Calibri"),
        legend.text = element_text(family = "Calibri")) +
  scale_fill_manual(values = c("#2E5077", "#9A8DFD", "#C6347F"))
```




```
# Step 3: Count the number of cells of each type in each cluster

# Count endogenous cells in the clusters
endogenous_counts <- table(endogenous_clusters)
# Count cells with VH replacement in the clusters
VHreplacement_counts <- table(VHreplacement_clusters)
# Count cells with A12-expressing
A12_counts <- table(A12_clusters)

# Step 4: Combine the counts into a dataframe

# Create a data frame with the clusters and counts
combined_counts <- data.frame(
  Cluster = names(endogenous_counts),
  Endogenous = as.integer(endogenous_counts),
  VHreplacement = as.integer(VHreplacement_counts[match(names(endogenous_counts), names(VHreplacement_counts))]),
  A12 = as.integer(A12_counts[match(names(endogenous_counts), names(A12_counts))])
)

# Replace NAs with zeros if a cluster has no cells of a certain type
combined_counts[is.na(combined_counts)] <- 0

# Step 5: Calculate proportions

# Calculate the total number of cells in each cluster
combined_counts$Total <- rowSums(combined_counts[, c("Endogenous", "VHreplacement", "A12")])

# Convert the counts into proportions
combined_counts$Endogenous <- combined_counts$Endogenous / combined_counts$Total
combined_counts$VHreplacement <- combined_counts$VHreplacement / combined_counts$Total
combined_counts$A12 <- combined_counts$A12 / combined_counts$Total

# Step 6: convert to long format (proportions instead of counts)
combined_counts_long <- melt(combined_counts[, -which(names(combined_counts) == "Total")],
```

```
                              id.vars = "Cluster",
                              variable.name = "Cell_Type",
                              value.name = "Proportion")

# Ensure the order of clusters in the plot

# Step 7: Create the stacked bar plot with proportions

# Proportion plot
ggplot(combined_counts_long, aes(x = Cluster, y = Proportion, fill = Cell_Type)) +
  geom_bar(stat = "identity", position = "fill") +  # 'position = "fill"' for proportions
  labs(title = "Number of Receptor Edited Cells by Cluster in SP",
       x = "Cluster",
       y = "Proportion",
       fill = "Cell Type") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, family = "Calibri"),
        axis.text.y = element_text(family = "Calibri"),
        plot.title = element_text(family = "Calibri"),
        legend.title = element_text(family = "Calibri"),
        legend.text = element_text(family = "Calibri")) +
  scale_fill_manual(values = c("#2E5077", "#9A8DFD", "#C6347F"))
```

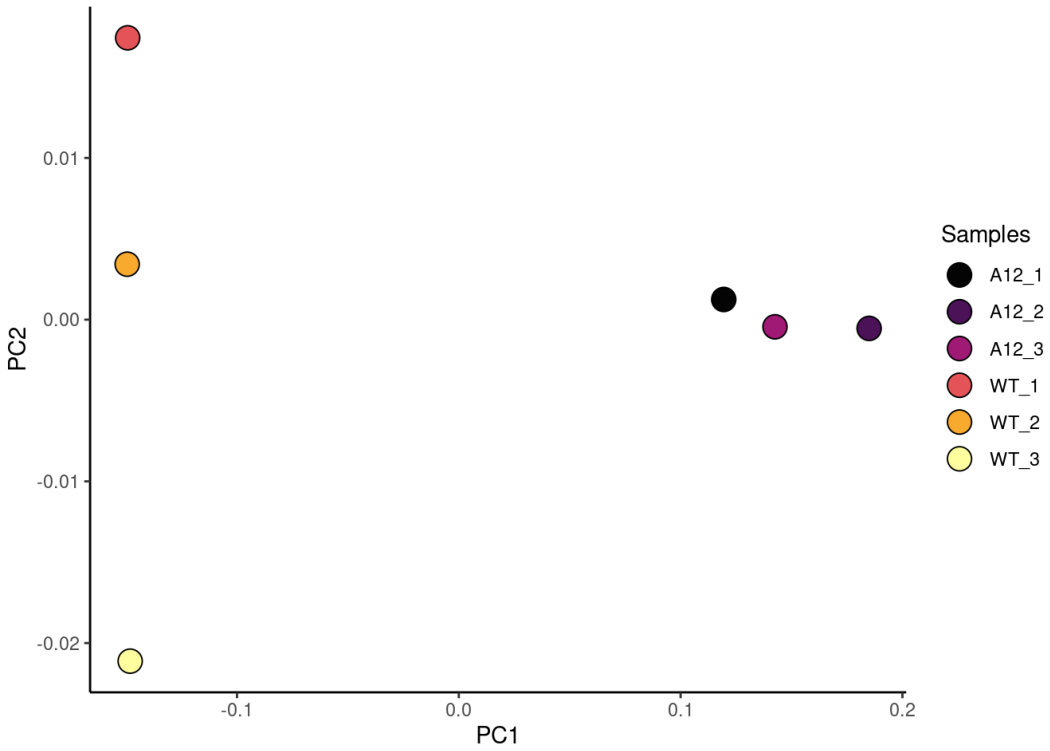


Dimensionality reduction analysis by PCA

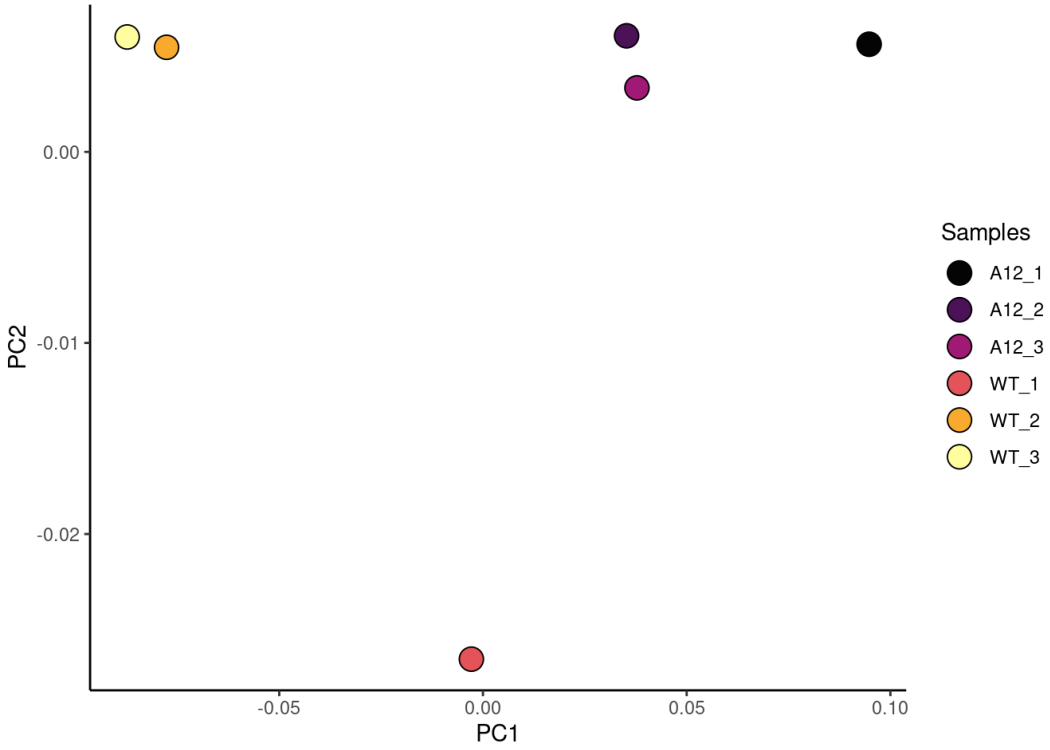
We can observe that A12 mice cluster using IgHV or IgHJ but not IgL

```
df.genes <- percentGenes(combined_BCR_BM,
                         chain = "IGH",
                         gene = "Vgene",
                         exportTable = TRUE)

#Performing PCA
pc <- prcomp(df.genes)

#Getting data frame to plot from
df <- as.data.frame(cbind(pc$x[,1:2], rownames(df.genes)))
df$PC1 <- as.numeric(df$PC1)
df$PC2 <- as.numeric(df$PC2)
```

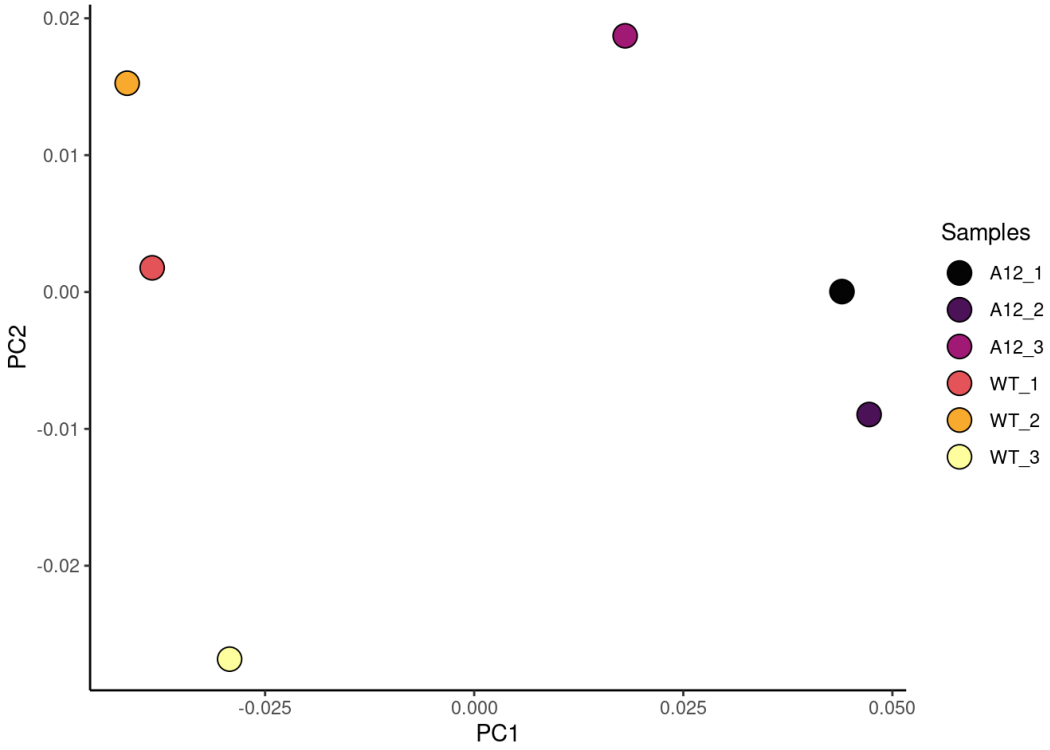
```
#Plotting
ggplot(df, aes(x = PC1, y = PC2)) +
  geom_point(aes(fill =df[,3]), shape = 21, size = 5) +
  guides(fill=guide_legend(title="Samples")) +
  scale_fill_manual(values = hcl.colors(nrow(df), "inferno")) +
  theme_classic()
```



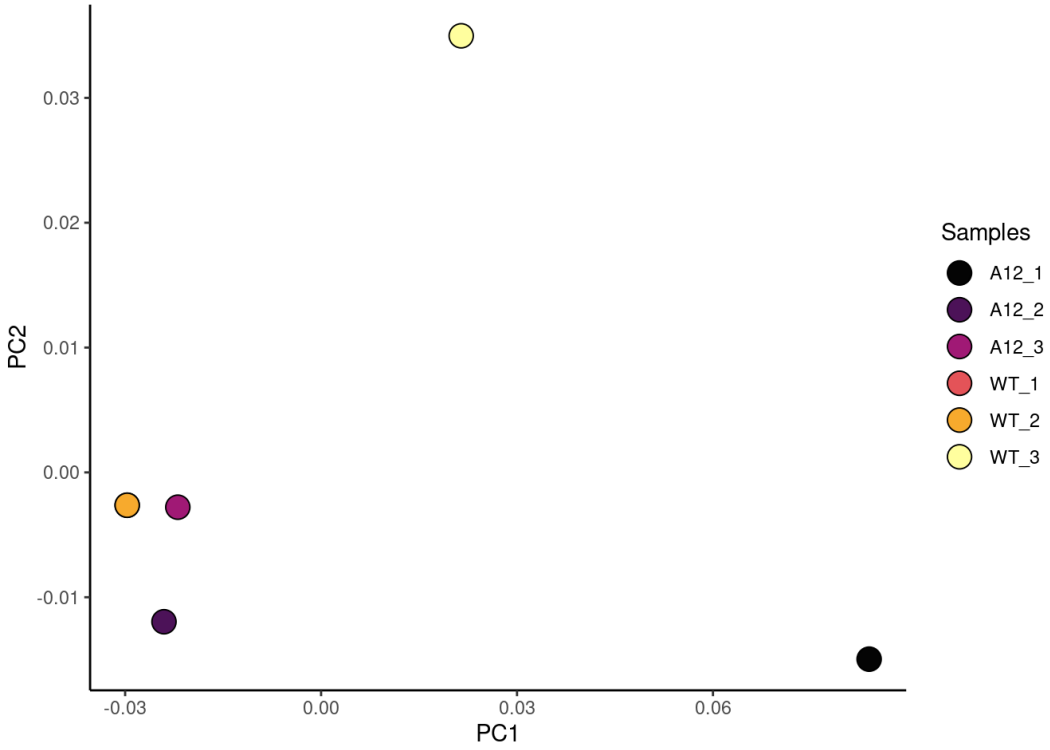

```
df.genes <- percentGenes(combined_BCR_BM,
                         chain = "IGL",
                         gene = "Vgene",
                         exportTable = TRUE)

#Performing PCA
pc <- prcomp(df.genes)

#Getting data frame to plot from
df <- as.data.frame(cbind(pc$x[,1:2], rownames(df.genes)))
df$PC1 <- as.numeric(df$PC1)
df$PC2 <- as.numeric(df$PC2)

#Plotting
ggplot(df, aes(x = PC1, y = PC2)) +
  geom_point(aes(fill =df[,3]), shape = 21, size = 5) +
  guides(fill=guide_legend(title="Samples")) +
  scale_fill_manual(values = hcl.colors(nrow(df), "inferno")) +
  theme_classic()
```




```
df.genes <- percentGenes(combined_BCR_SP,
                         chain = "IGH",
                         gene = "Vgene",
                         exportTable = TRUE)

#Performing PCA
pc <- prcomp(df.genes)

#Getting data frame to plot from
df <- as.data.frame(cbind(pc$x[,1:2], rownames(df.genes)))
df$PC1 <- as.numeric(df$PC1)
df$PC2 <- as.numeric(df$PC2)

#Plotting
ggplot(df, aes(x = PC1, y = PC2)) +
  geom_point(aes(fill =df[,3]), shape = 21, size = 5) +
  guides(fill=guide_legend(title="Samples")) +
  scale_fill_manual(values = hcl.colors(nrow(df), "inferno")) +
  theme_classic()
```

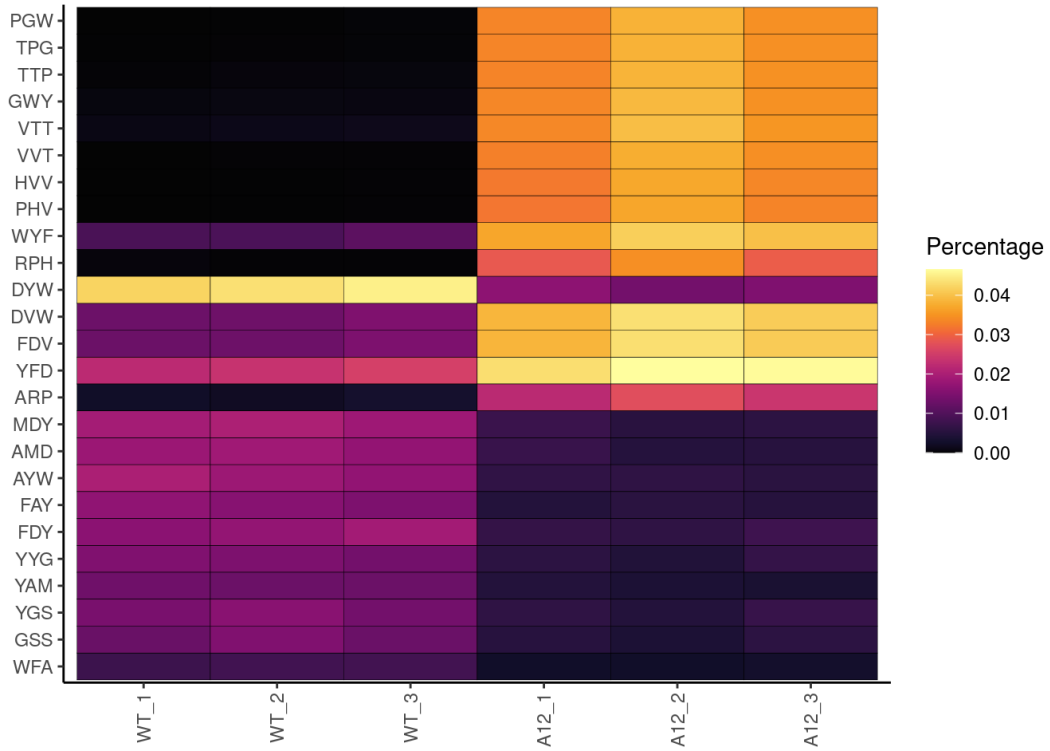



```
df.genes <- percentGenes(combined_BCR_SP,
                         chain = "IGL",
                         gene = "Vgene",
                         exportTable = TRUE)

#Performing PCA
pc <- prcomp(df.genes)

#Getting data frame to plot from
df <- as.data.frame(cbind(pc$x[,1:2], rownames(df.genes)))
df$PC1 <- as.numeric(df$PC1)
df$PC2 <- as.numeric(df$PC2)

#Plotting
ggplot(df, aes(x = PC1, y = PC2)) +
  geom_point(aes(fill =df[,3]), shape = 21, size = 5) +
  guides(fill=guide_legend(title="Samples")) +
  scale_fill_manual(values = hcl.colors(nrow(df), "inferno")) +
  theme_classic()
```



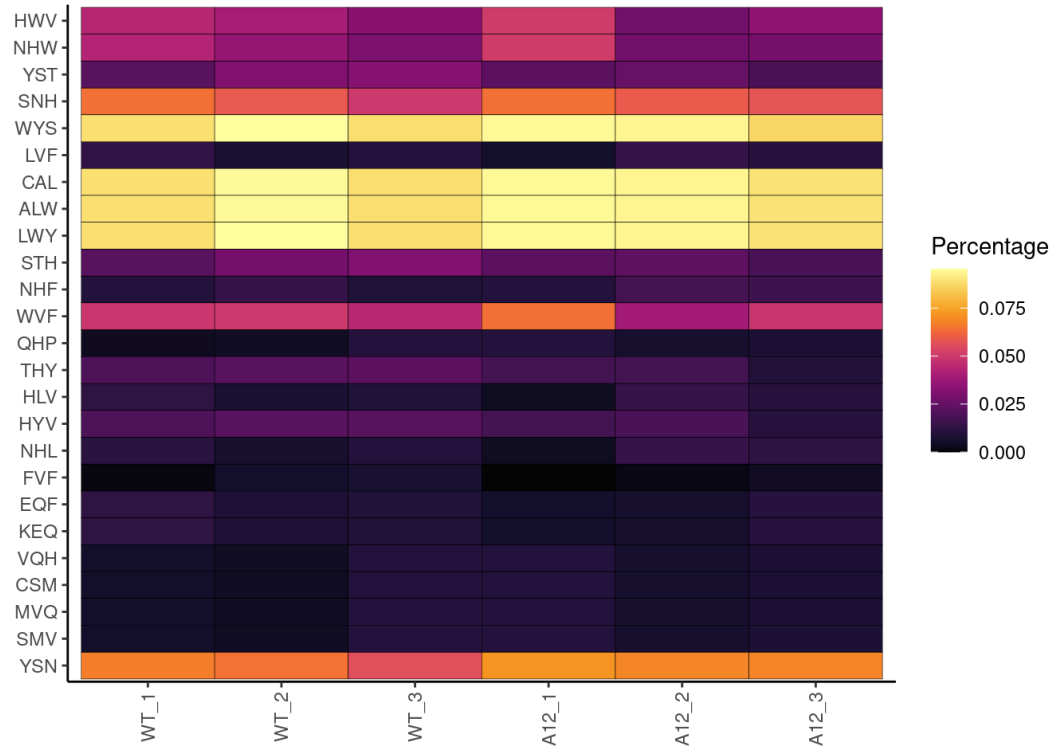
Searching for motifs in CDR3

```
percentKmer(combined_BCR_BM,
            cloneCall = "aa",
            chain = "IGH",
            motif.length = 3,
            top.motifs = 25)
```



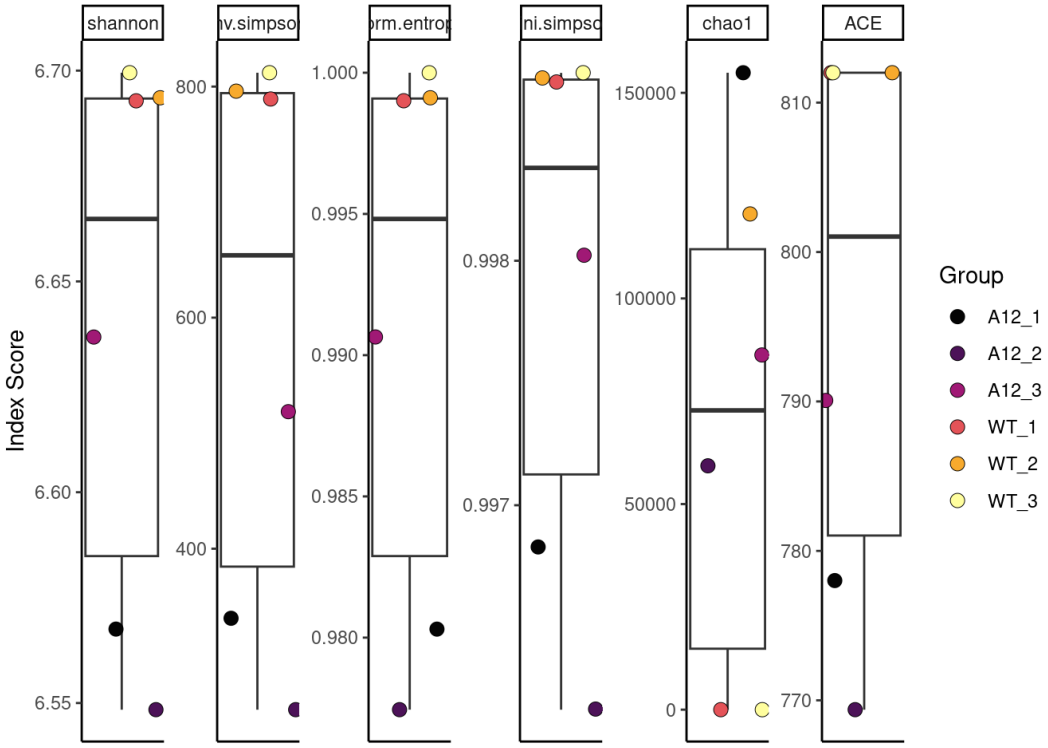

```
percentKmer(combined_BCR_BM,
            cloneCall = "aa",
            chain = "IGL",
```

```
            motif.length = 3,
            top.motifs = 25)
```



Clonal diversity and overlap

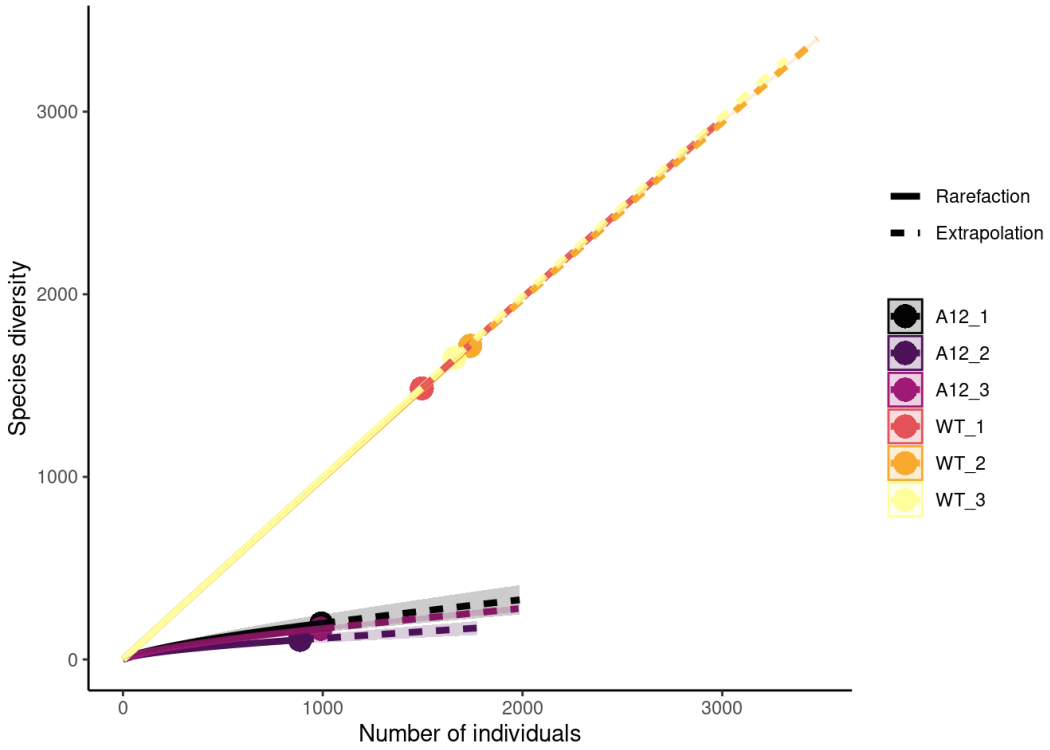
```
clonalDiversity(combined_BCR_SP,
                cloneCall = "aa",
                chain = "IGH",
                n.boots = 20)
```



Rarefaction

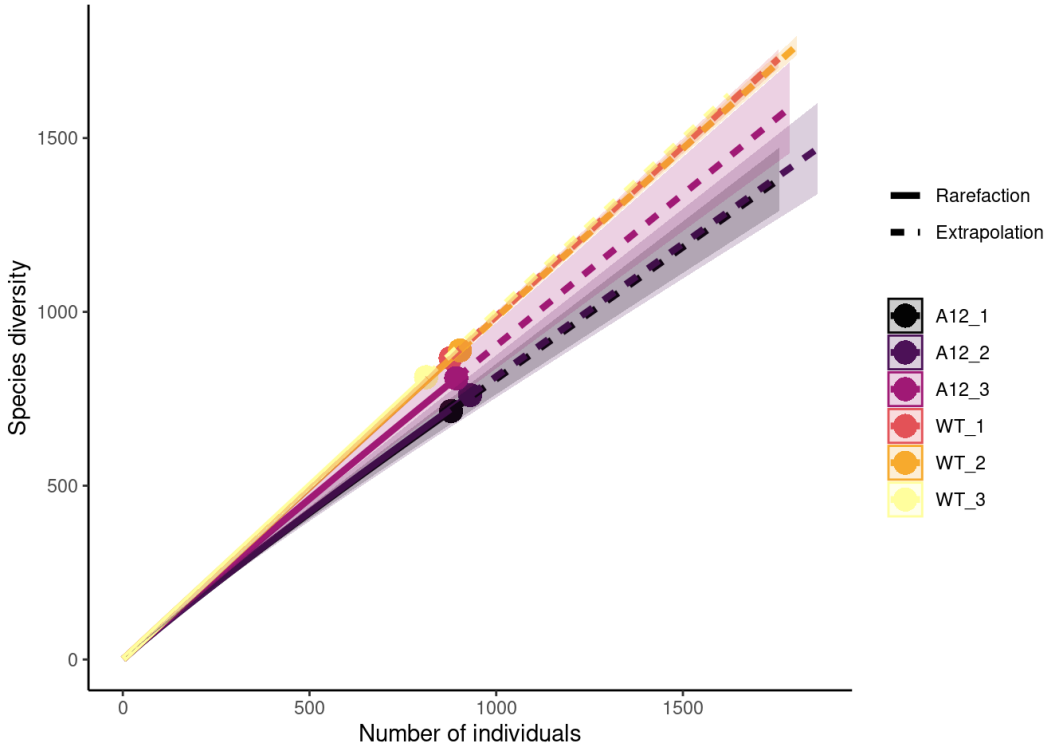
Rarefaction is a technique used in ecology and biodiversity studies to standardize or compare species richness (i.e., the number of different species) across different samples or environments, especially when those samples differ in size. It allows researchers to estimate how many species they would expect to find in smaller or larger samples, helping to account for the fact that larger samples naturally tend to contain more species simply because they cover more ground or involve more individuals.

```
clonalRarefaction(combined_BCR_BM,
                  cloneCall = "aa",
                  chain = "IGH",
                  plot.type = 1,
                  hill.numbers = 1,
                  n.boots = 2)
```




```
clonalRarefaction(combined_BCR_SP,
                  cloneCall = "aa",
                  chain = "IGH",
                  plot.type = 1,
                  hill.numbers = 1,
                  n.boots = 2)
```

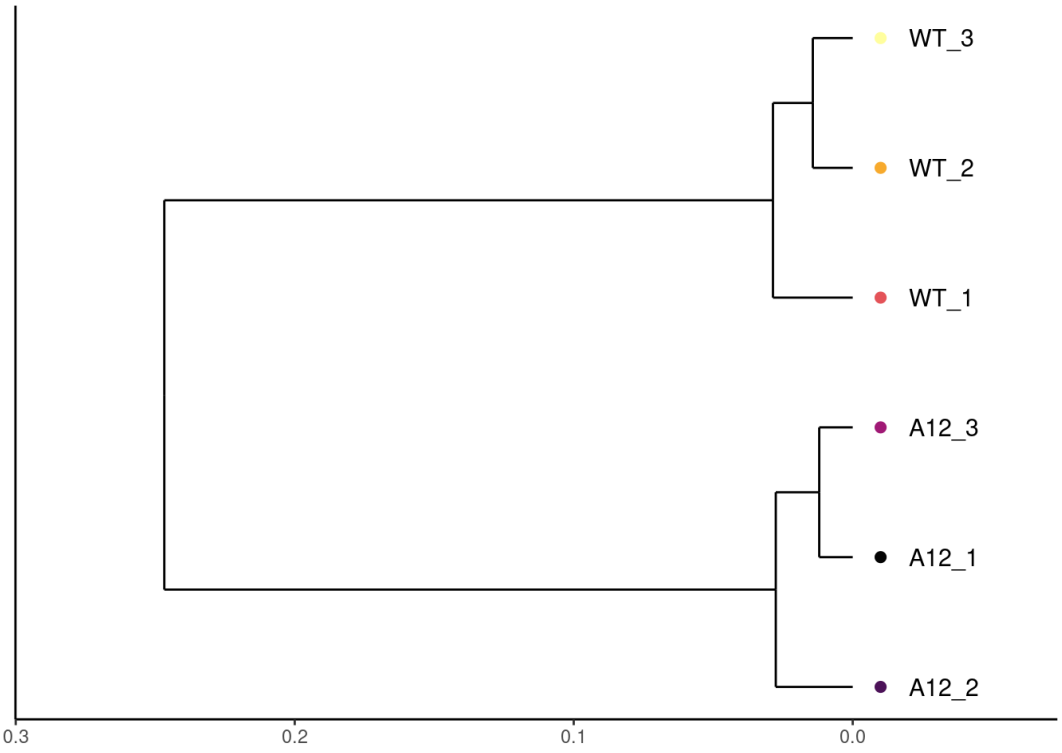
```
## Warning in matrix(apply(Abun.Mat, 2, function(x) invChat.Ind(x, q, goalSC)$qD),
## : la longitud de los datos [48] no es un submúltiplo o múltiplo del número de
## filas [22] en la matriz
```



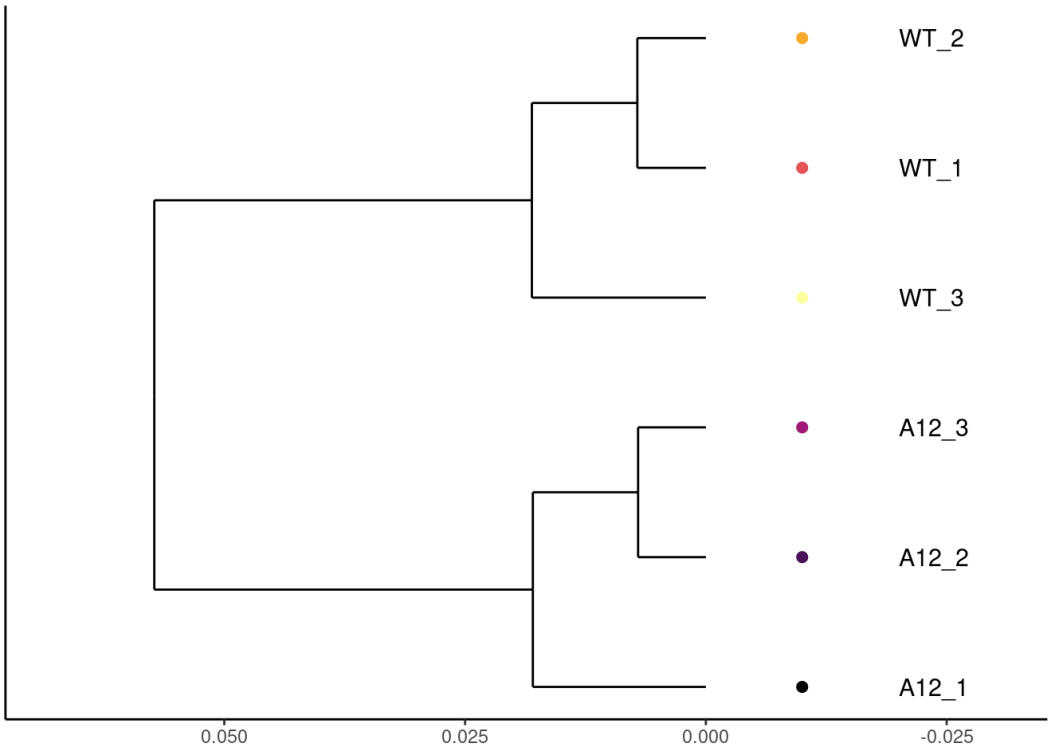
Clonal Size distribution

Another method for modeling the repertoire distribution is a discrete gamma-GPD spliced threshold model, proposed by Koch et al. The spliced model models the repertoire and allows for the application of a power law distribution for larger clonal-expanded sequences and a Poisson distribution for smaller clones. After fitting the models, repertoires can be compared using Euclidean distance.

```
clonalSizeDistribution(combined_BCR_BM,
                       cloneCall = "aa",
                       chain = "IGH",
                       method= "ward.D2")
```

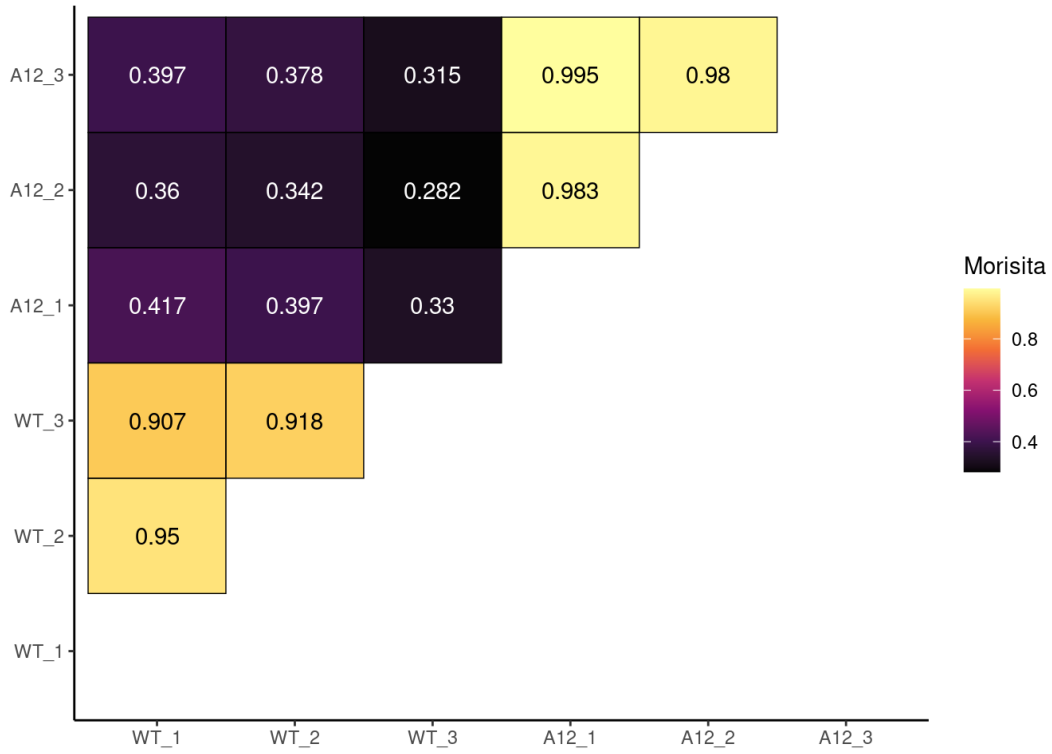


```
clonalSizeDistribution(combined_BCR_SP,
                       cloneCall = "strict",
                       chain = "IGH",
                       method= "ward.D2")
```

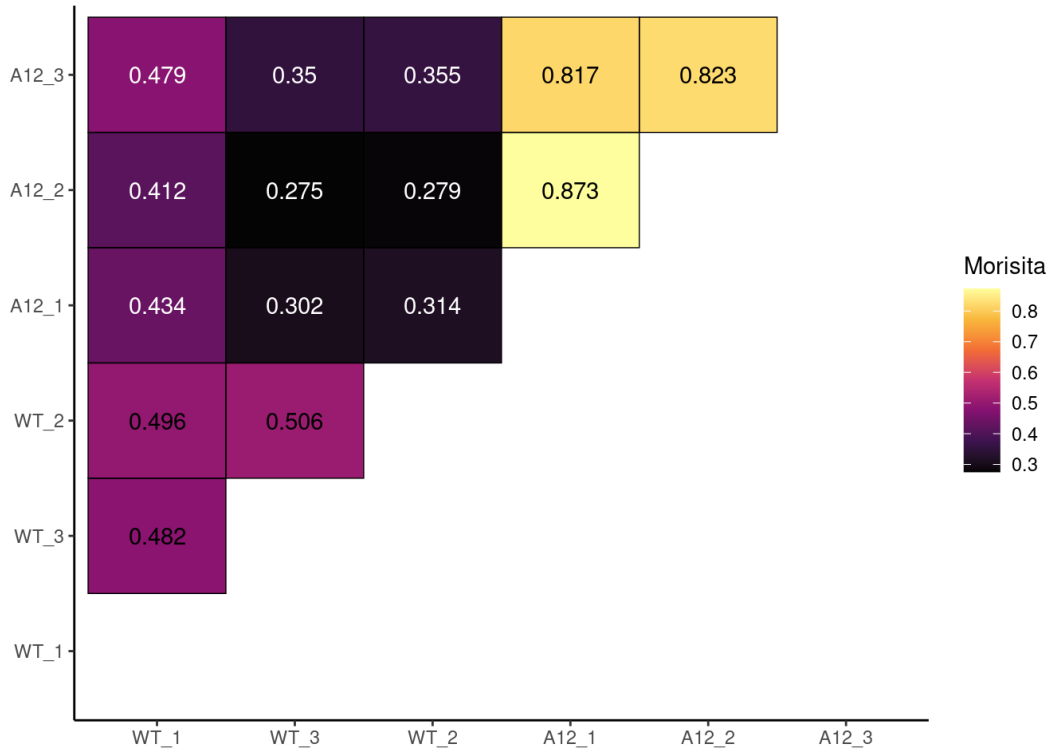


Clonal overlap

```
clonalOverlap(combined_BCR_BM,
              cloneCall = "strict",
              chain = "IGH",
              method = "morisita")
```



```
clonalOverlap(combined_BCR_SP,
              cloneCall = "strict",
              chain = "IGH",
              method = "morisita")
```

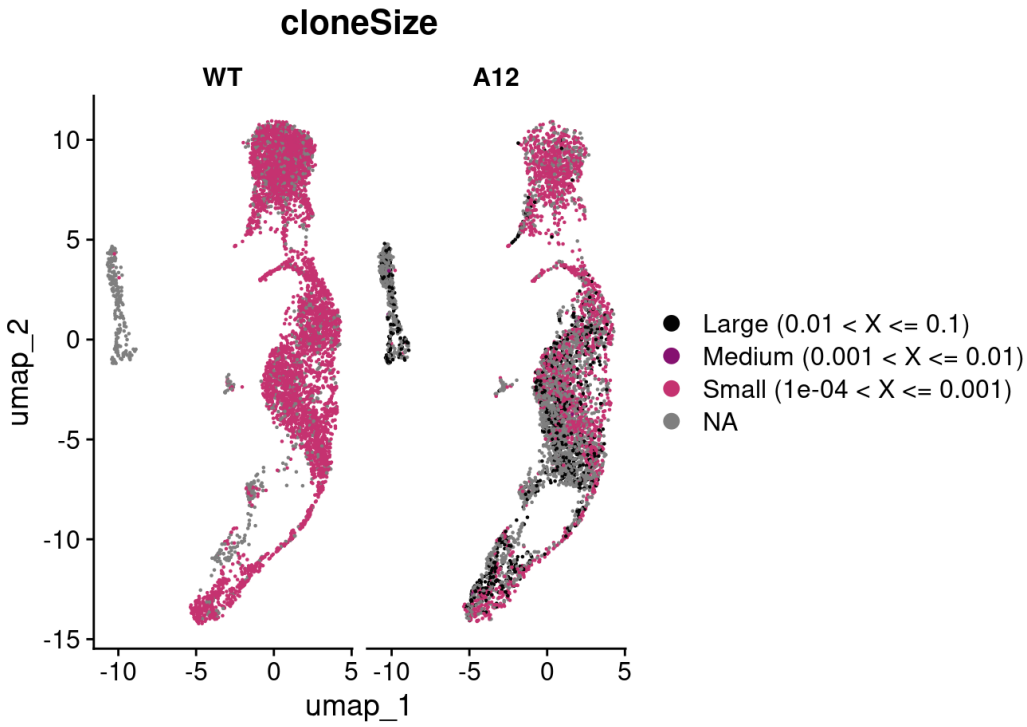


Calculating cloneSize

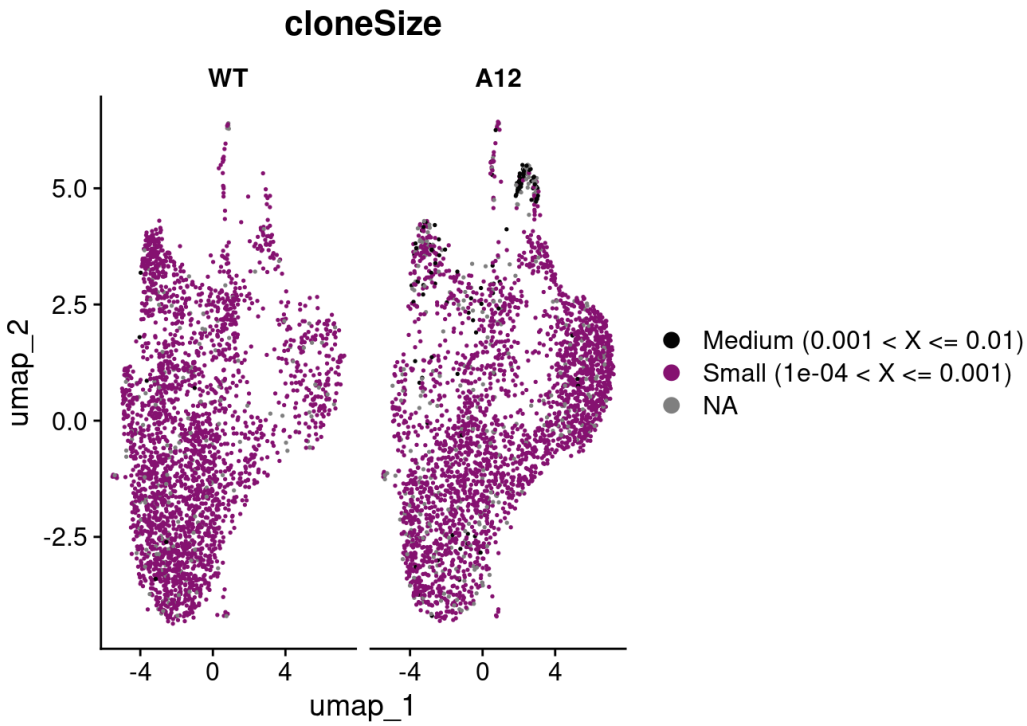
Part of combineExpression() is calculating the clonal frequency and proportion, placing each clone into groups called cloneSize. The default cloneSize argument uses the following bins: c(Rare = 1e-4, Small = 0.001, Medium = 0.01, Large = 0.1, Hyperexpanded = 1)

Clonal frequency and proportion is dependent on the repertoires being compared, which we can modify the calculation using the group.by parameter. If group.by is not set, combineExpression() will calculate clonal frequency, proportion, and cloneSize as a function of individual sequencing runs. In addition, cloneSize can use the frequency of clones when proportion = FALSE.

```
colorblind_vector <- hcl.colors(n=7, palette = "inferno", fixup = TRUE)

DimPlot(sce_BM, reduction = "umap", group.by = "cloneSize", split.by = "genotype") +
    scale_color_manual(values=rev(colorblind_vector[c(5,4,3,1)]))
```




```
colorblind_vector <- hcl.colors(n=7, palette = "inferno", fixup = TRUE)

DimPlot(sce_SP, reduction = "umap", group.by = "cloneSize", split.by = "genotype") +
    scale_color_manual(values=rev(colorblind_vector[c(5,4,3,1)]))
```

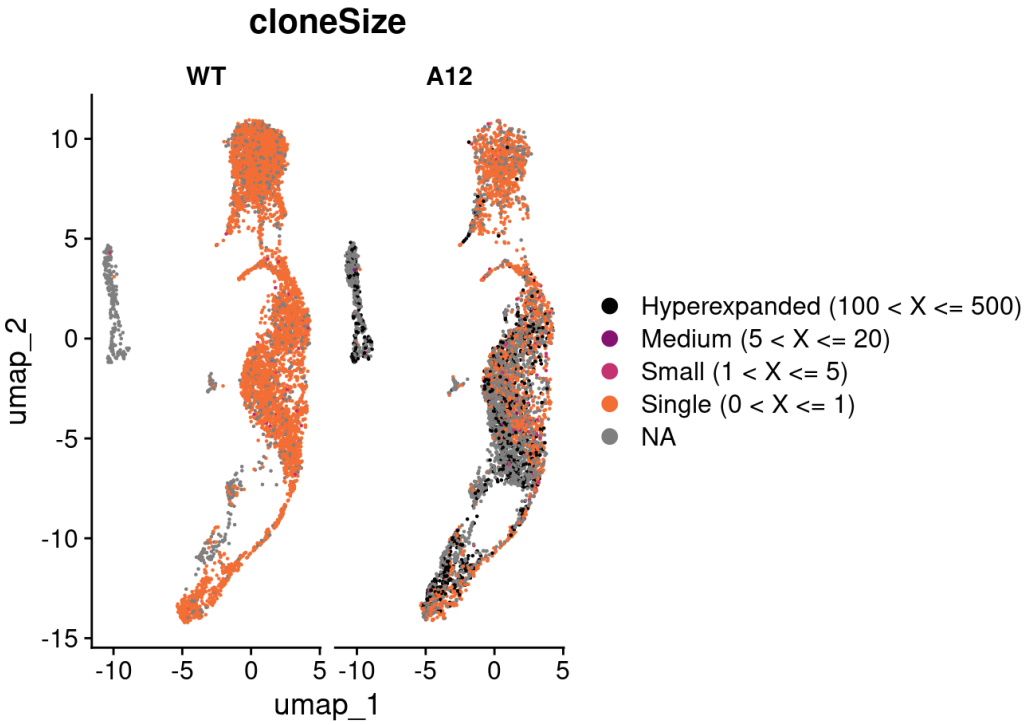


Alternatively, if we want cloneSize to be based on the frequency of the clone, we can set proportion = FALSE and we will need to change the cloneSize bins to integers. If we have not inspected our clone data, setting the upper limit of the clonal frequency might be difficult - combineExpression() will automatically adjust the upper limit to fit the distribution of the frequencies.

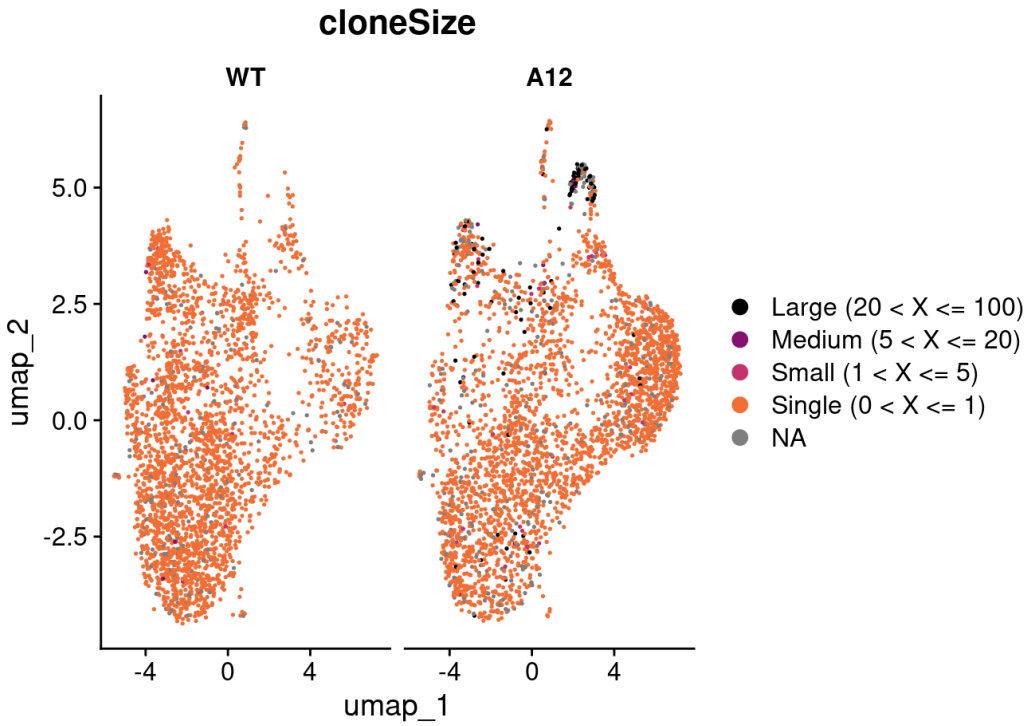
```
sce_BM <- combineExpression(combined_BCR_BM,
                        Seurat_BM_2,
                        cloneCall="aa",
```

```
                           chain = "IGH",
                           group.by = "sample",
                           filterNA = FALSE,
                           proportion = FALSE,
                           cloneSize=c(Single=1, Small=5, Medium=20, Large=100, Hyperexpanded=500))



sce_SP <- combineExpression(combined_BCR_SP,
                           Seurat_SP_2,
                           cloneCall="aa",
                           chain = "IGH",
                           group.by = "sample",
                           filterNA = FALSE,
                           proportion = FALSE,
                           cloneSize=c(Single=1, Small=5, Medium=20, Large=100, Hyperexpanded=500))
```

```
colorblind_vector <- hcl.colors(n=7, palette = "inferno", fixup = TRUE)

DimPlot(sce_BM, reduction = "umap", group.by = "cloneSize", split.by = "genotype") +
    scale_color_manual(values=rev(colorblind_vector[c(5,4,3,1)]))
```

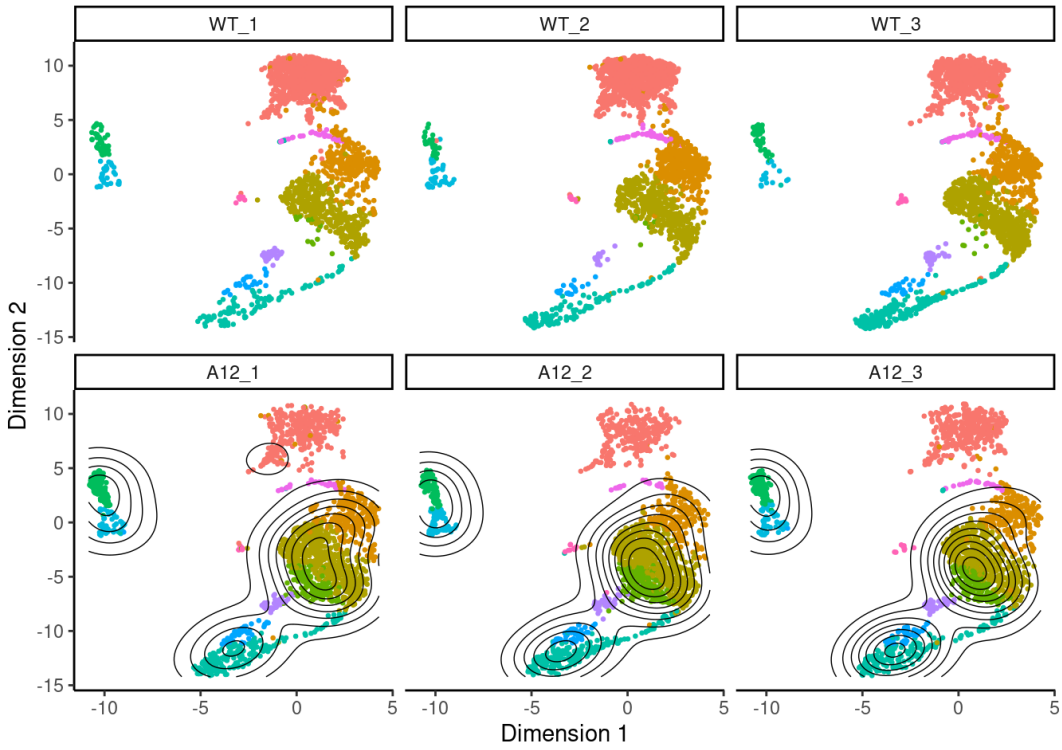



```
colorblind_vector <- hcl.colors(n=7, palette = "inferno", fixup = TRUE)

DimPlot(sce_SP, reduction = "umap", group.by = "cloneSize", split.by = "genotype") +
    scale_color_manual(values=rev(colorblind_vector[c(5,4,3,1)]))
```



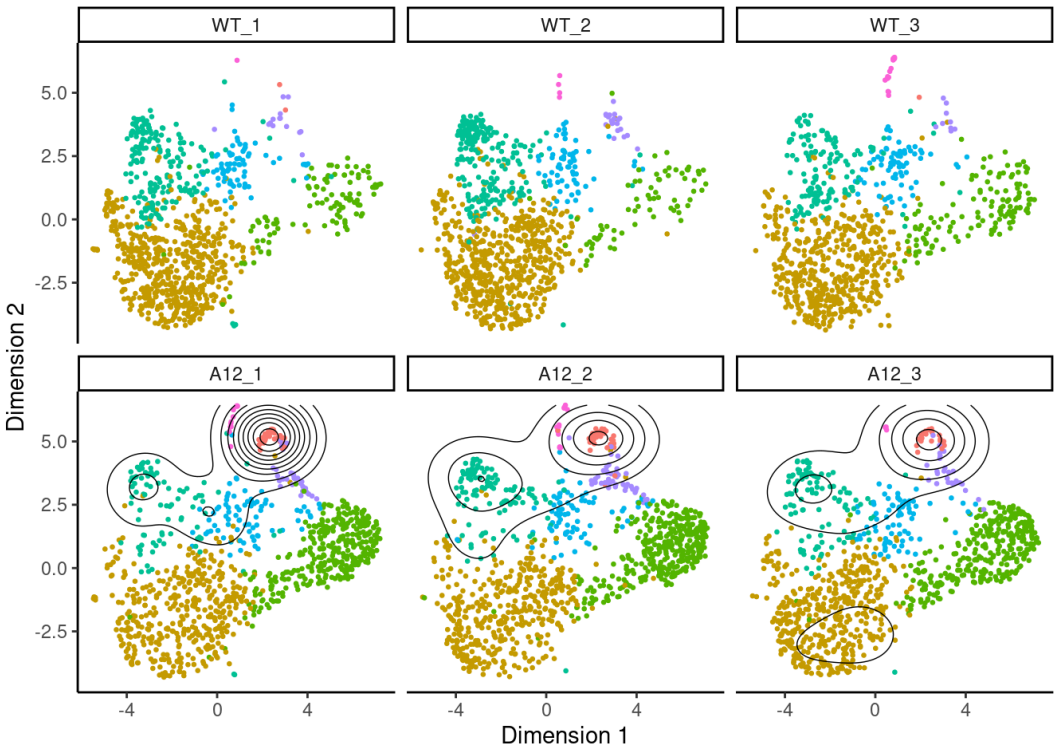
Clonal Overlay

```
clonalOverlay(sce_BM,
              reduction = "umap",
              cut.category = "clonalFrequency",
              cutpoint = 100,
              bins = 10,
              facet.by = "mice") +
              guides(color = "none")
```




```
clonalOverlay(sce_SP,
```

```
                reduction = "umap",
                cut.category = "clonalFrequency",
                cutpoint = 15,
                bins = 10,
                facet.by = "mice") +
                guides(color = "none")
```



Clonal Network

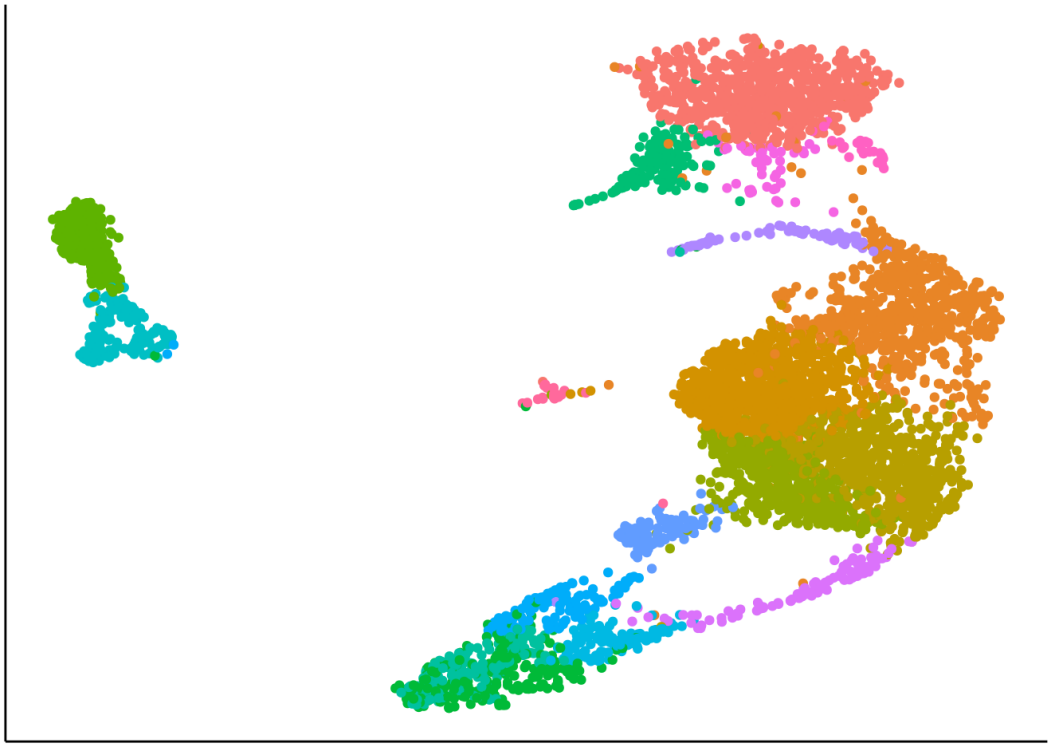
```
sce_BM_A12 <- subset(sce_BM, genotype == "A12")
sce_BM_WT <- subset(sce_BM, genotype == "WT")

sce_SP_A12 <- subset(sce_SP, genotype == "A12")
sce_SP_WT <- subset(sce_SP, genotype == "WT")
```

```
#ggraph needs to be loaded due to issues with ggplot
library(ggraph)
```

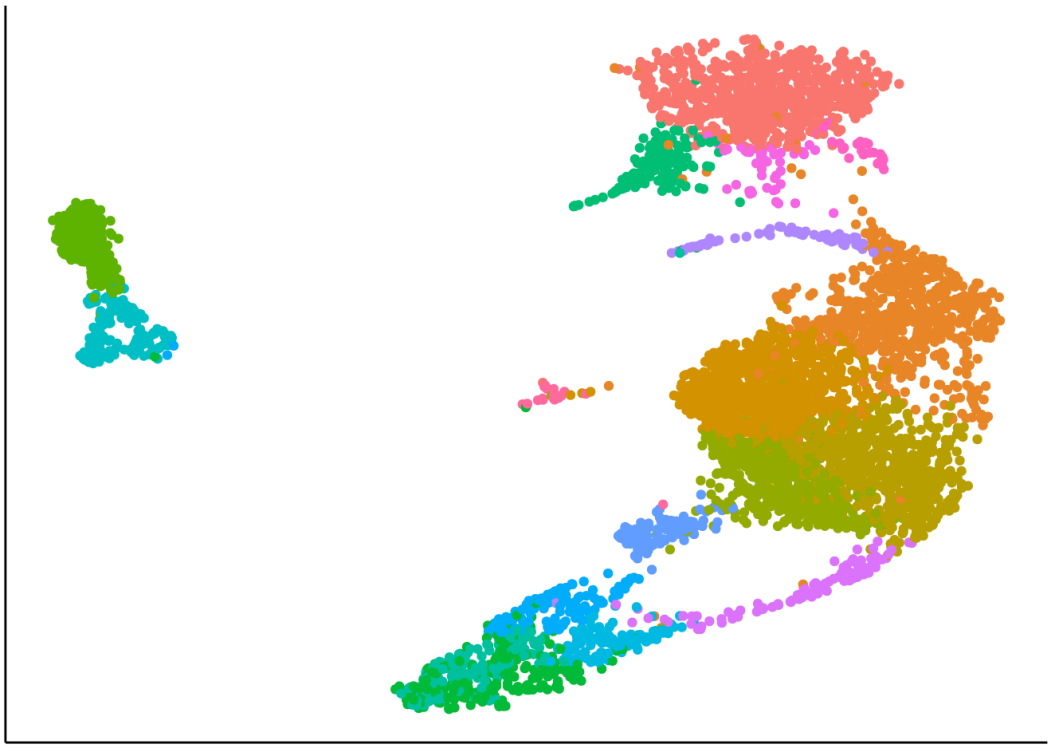
```
##
## Attaching package: 'ggraph'
```

```
## The following object is masked from 'package:sp':
##
##     geometry
```

```
colnames(sce_BM@meta.data) <- make.unique(colnames(sce_BM@meta.data))
#No Identity filter
clonalNetwork(sce_BM_A12,
              reduction = "umap",
              group.by = "seurat_clusters",
              filter.clones = 20,
              filter.identity = "FrE",
              cloneCall = "aa")
```

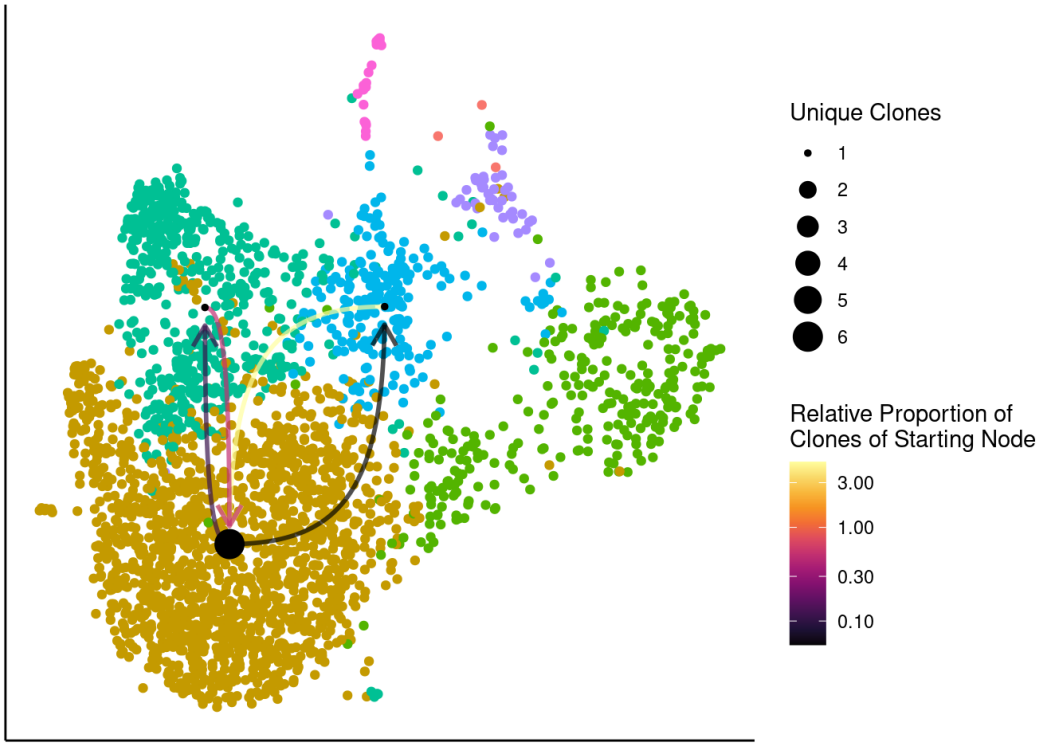


```
clonalNetwork(sce_BM_A12,
              reduction = "umap",
              group.by = "seurat_clusters",
              filter.clones = 20,
              filter.identity = "FrE",
              cloneCall = "aa")
```

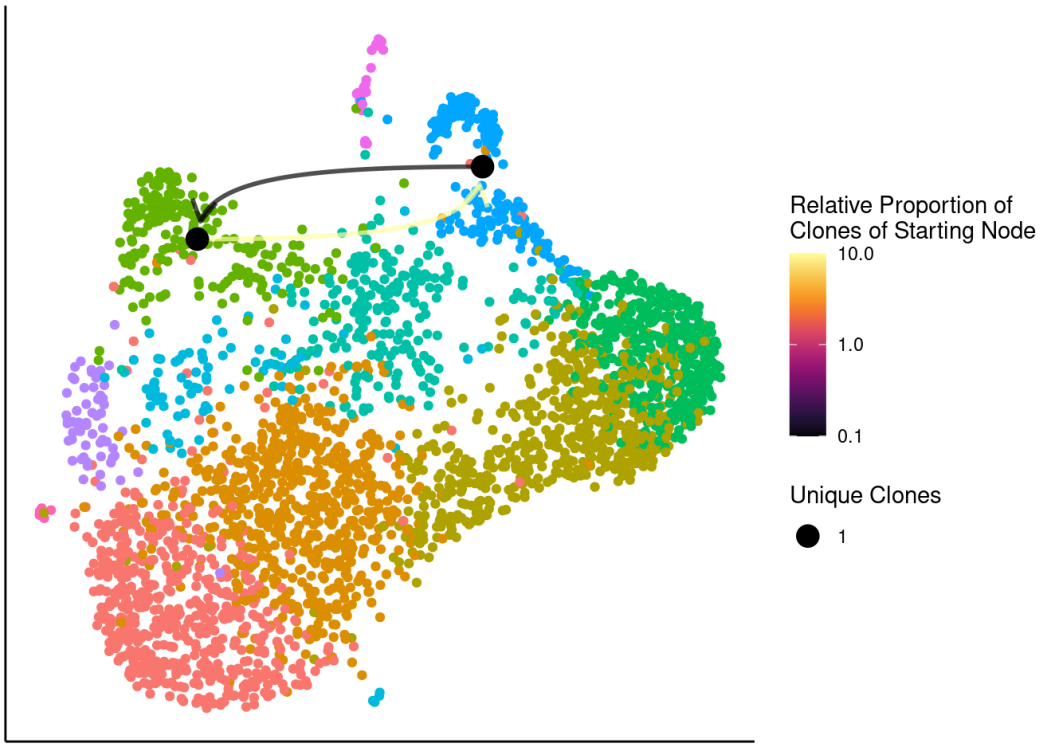


```
#ggraph needs to be loaded due to issues with ggplot
library(ggraph)

#No Identity filter
clonalNetwork(sce_SP_WT,
```

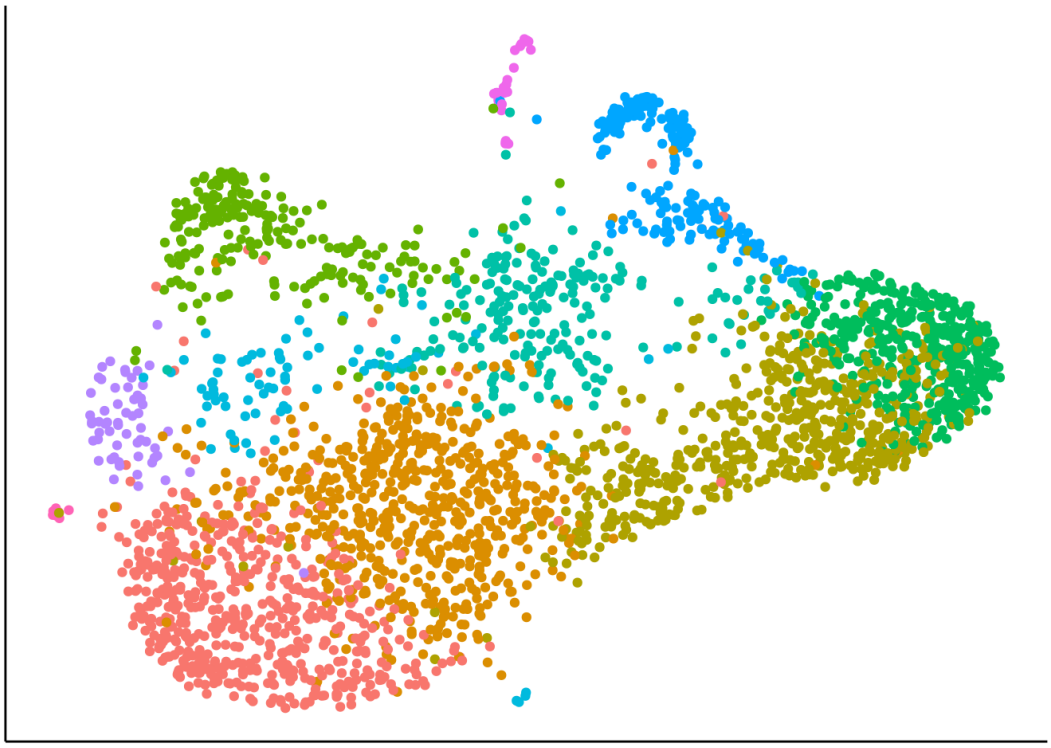
```
              reduction = "umap",
              filter.clones = 20,
              filter.identity = NULL,
              cloneCall = "strict",
              palette= "inferno")
```




```
clonalNetwork(sce_SP_A12,
              reduction = "umap",
              group.by = "seurat_clusters",
              filter.clones = 20,
              filter.identity = NULL,
              cloneCall = "aa")
```



```
clonalNetwork(sce_SP_A12,
              reduction = "umap",
              group.by = "seurat_clusters",
              filter.clones = 15,
              filter.identity = "A12 uniq",
              cloneCall = "aa")
```



```
shared.clones <- clonalNetwork(sce_BM_WT,
              reduction = "umap",
              group.by = "seurat_clusters",
              filter.clones = 200,
              cloneCall = "aa",
              exportClones = TRUE)
head(shared.clones)
```

clone <chr>	sum <int>
NA_CALWYSNHWVF	9
NA_CLQSDNLPLTF	6
NA_CQQHYSTPWTF	6
NA_CQQWSSNPPTF	6
NA_CQQYSSYPLTF	6
NA_CALWYSTHYVF;NA	5

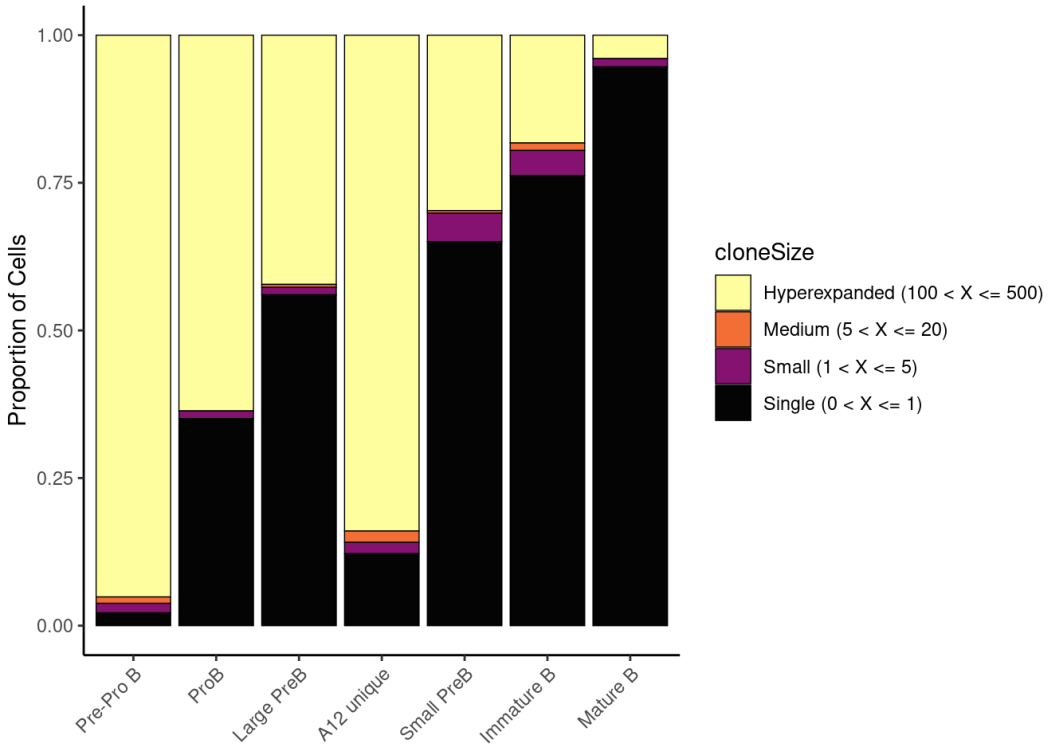
6 rows

```
shared.clones <- clonalNetwork(sce_BM_A12,
              reduction = "umap",
              group.by = "seurat_clusters",
              filter.clones = 200,
              cloneCall = "aa",
              exportClones = TRUE)
head(shared.clones)
```

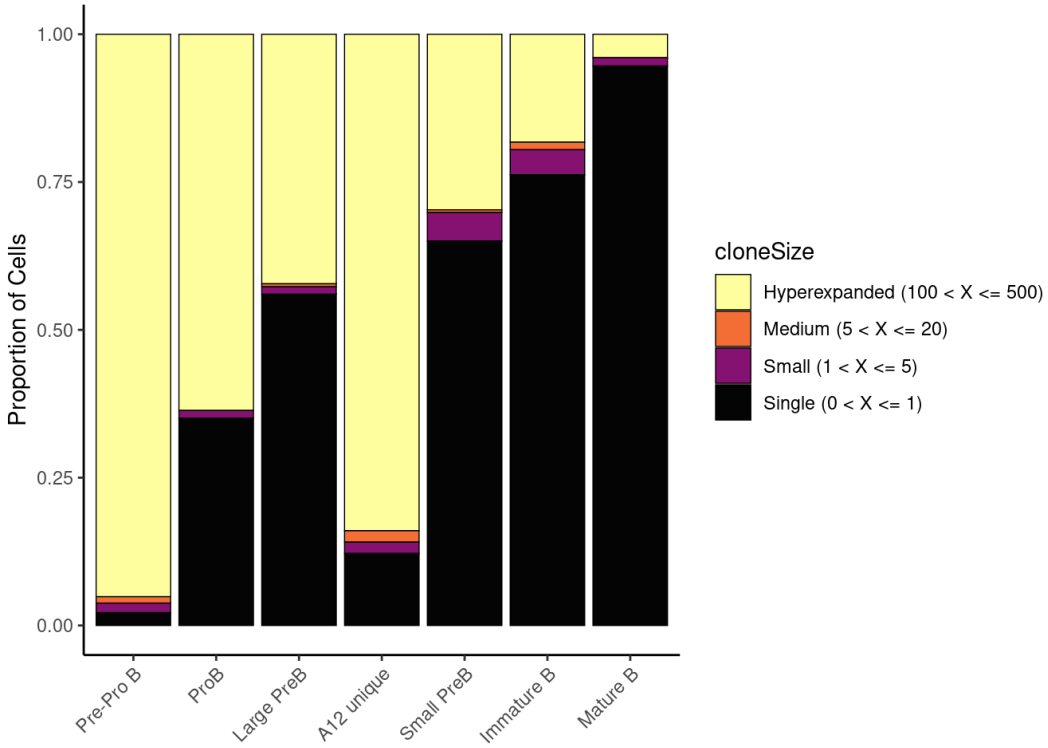
clone	sum
<chr>	<int>
CARPHVVTTPGWYFDVW_NA	675

1 row

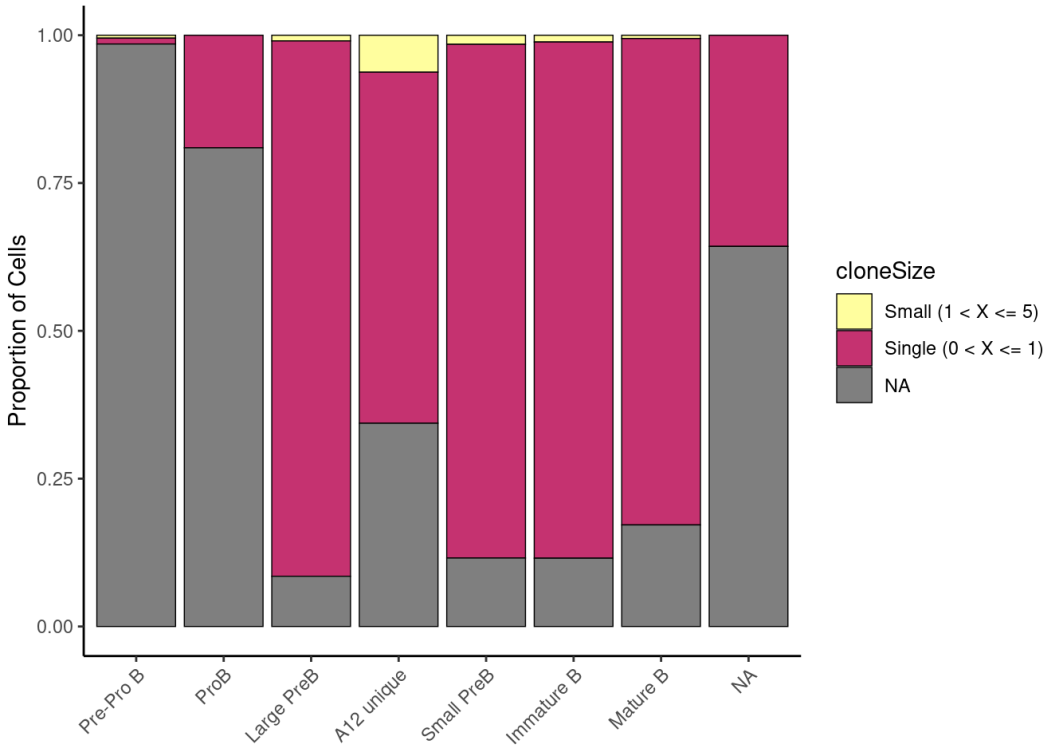
```
clonalOccupy(sce_BM_A12,
                     x.axis = "simplified_clusters",
                     proportion = TRUE,
                     na.include = FALSE,
                     label = FALSE) +
                    ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
```




```
clonalOccupy(sce_BM_A12,
                     x.axis = "simplified_clusters",
                     proportion = TRUE,
                     label = FALSE) +
                    ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
```

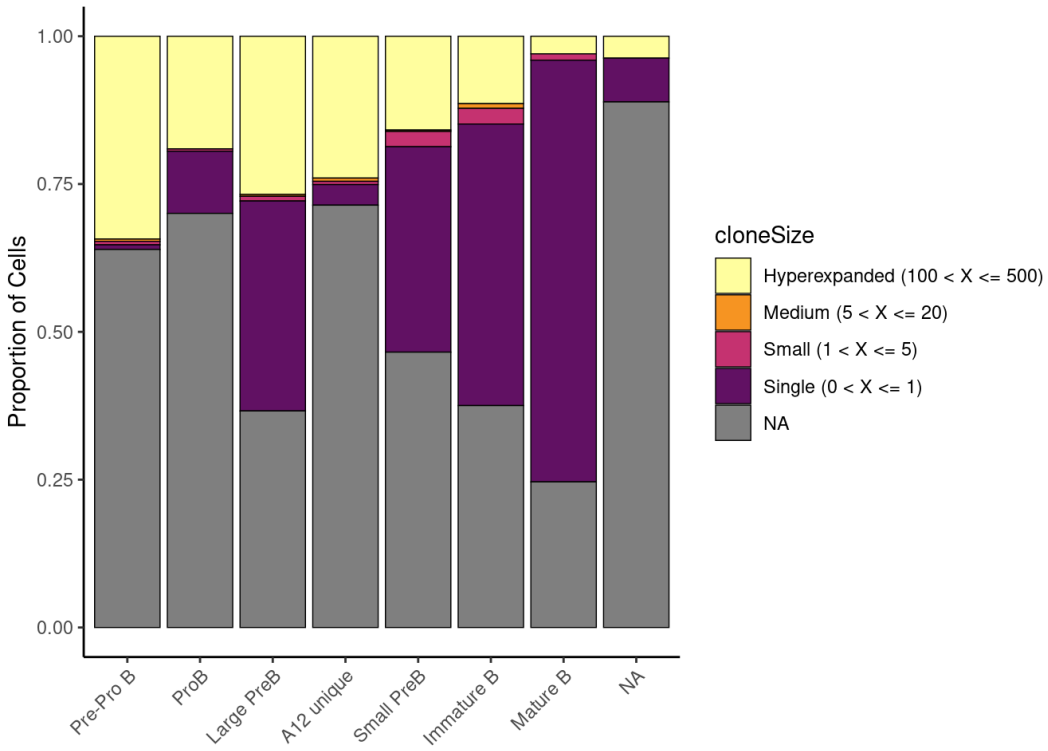



```
clonalOccupy(sce_BM_WT,
                     x.axis = "simplified_clusters",
                     proportion = TRUE,
                     label = FALSE,
                     na.include = TRUE) +
                    ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
```

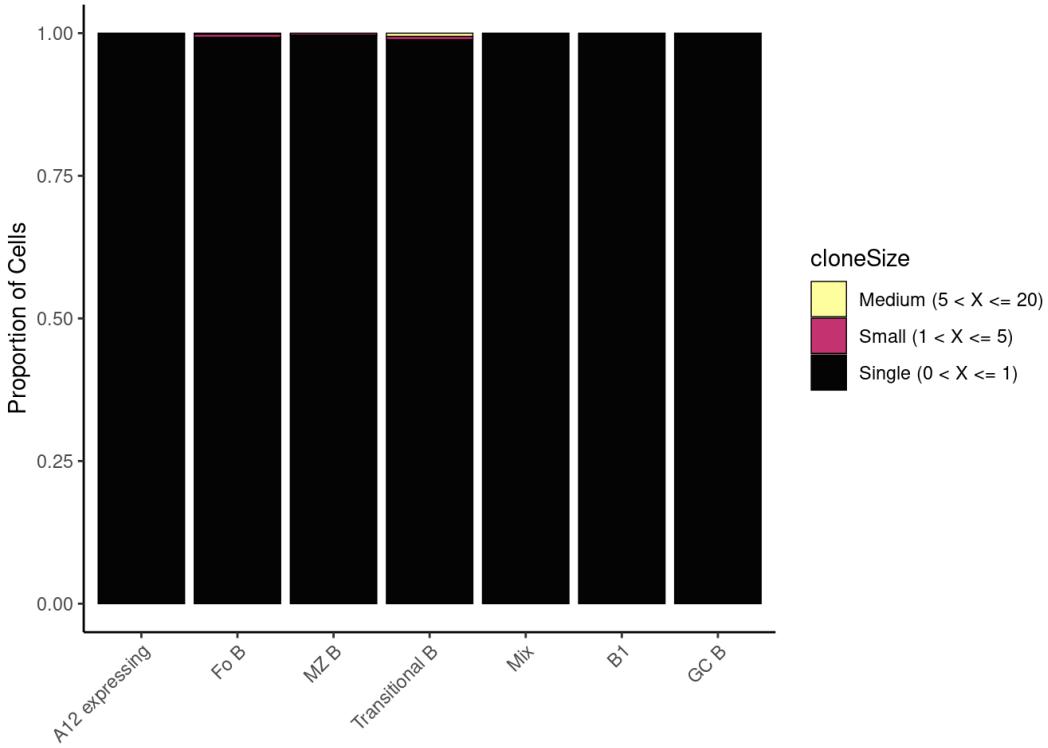



```
clonalOccupy(sce_BM_A12,
                     x.axis = "simplified_clusters",
                     proportion = TRUE,
                     label = FALSE,
                     na.include = TRUE) +
```

```
                    ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
```

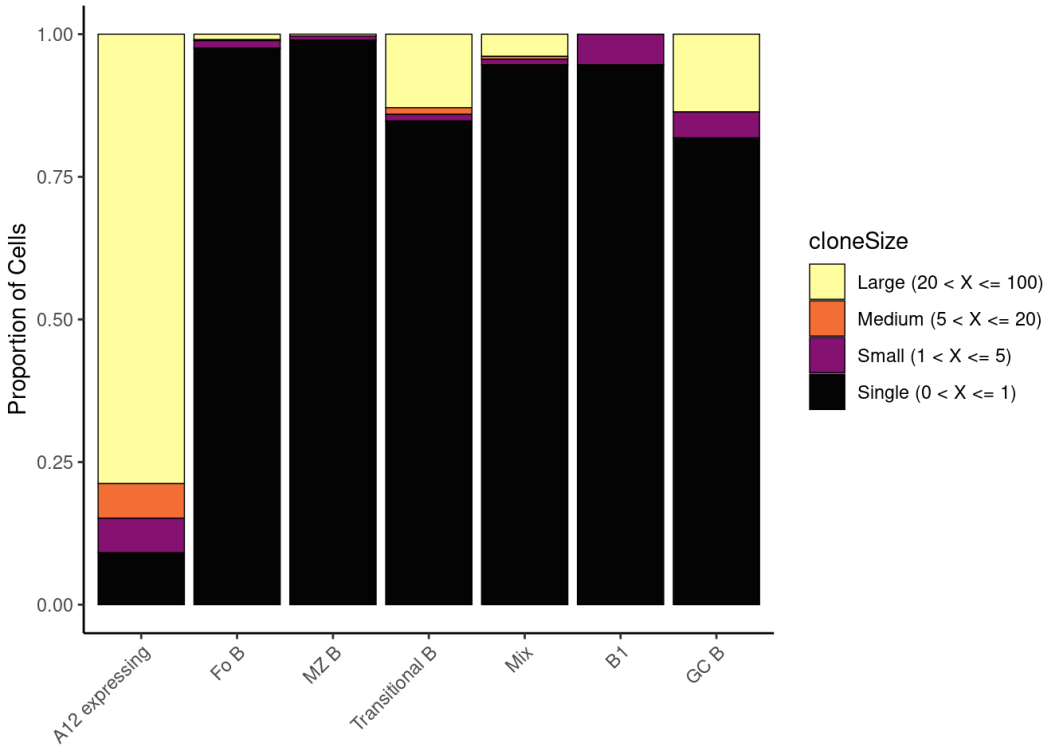



```
clonalOccupy(sce_SP_WT,
                     x.axis = "seurat_clusters_new_fin",
                     proportion = TRUE,
                     label = FALSE) +
                    ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
```

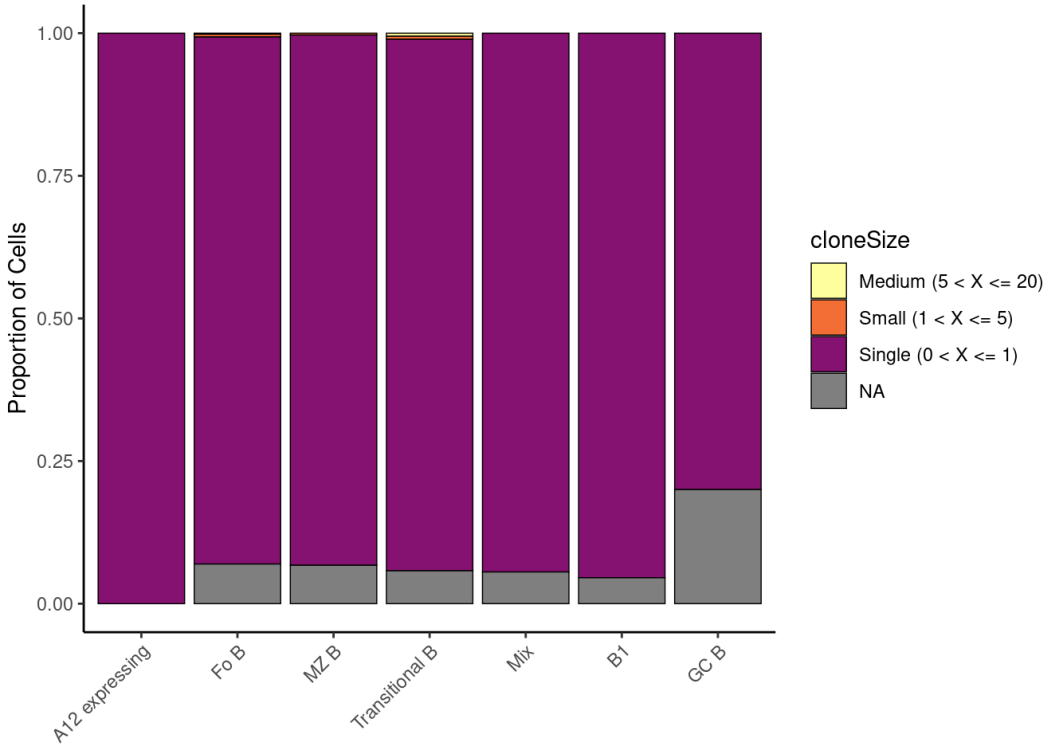



```
clonalOccupy(sce_SP_A12,
                     x.axis = "seurat_clusters_new_fin",
                     proportion = TRUE,
                     label = FALSE) +
```

```
                    ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
```




```
clonalOccupy(sce_SP_WT,
                     x.axis = "seurat_clusters_new_fin",
                     proportion = TRUE,
                     label = FALSE,
                     na.include = TRUE) +
                    ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
```




```
clonalOccupy(sce_SP_A12,
                     x.axis = "seurat_clusters_new_fin",
                     proportion = TRUE,
```

```
                    label = FALSE,
                    na.include = TRUE) +
                  ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 45, hjust = 1))
```