scRepertoire v2 Analysis

Nick Borcherding

2024-12-23

Loading Libraries

```
# Define required libraries
packages <- c(
  "Azimuth", "BiocParallel", "celldex", "dplyr", "ggplot2",
  "ggthemes", "harmony", "igraph", "patchwork", "RColorBrewer",
  "scDblFinder", "scGate", "scRepertoire", "scran", "Seurat",
  "SeuratData", "SingleR", "stringr", "viridis"
)

# Suppress startup messages
suppressPackageStartupMessages(
  lapply(packages, library, character.only = TRUE)
)

# Load annotation references if not already present
if (!exists("HPCA")) {
  HPCA <- celldex::HumanPrimaryCellAtlasData()
}
if (!exists("Monaco")) {
  Monaco <- celldex::MonacoImmuneData()
}

# Set options
options(future.globals.maxSize = 8000 * 1024^2)

# Define custom negate function
"%!in%" <- Negate("%in%")
```

Directory Setup

```
# Define directories
qc_dir <- "./qc"
input_dir <- "./inputs/data/GSE169440"
output_dir <- "./output"
processed_dir <- "./inputs/data/processedData"

# Create directories if they do not exist
dir.create(qc_dir, showWarnings = FALSE)
dir.create(output_dir, showWarnings = FALSE)
dir.create(processed_dir, showWarnings = FALSE)
```

Helper Functions

```
# Save plots with consistent formatting
save_plot <- function(plot, filename, height = 3, width = 3, dpi = 300) {
  ggsave(filename = filename, plot = plot, height = height, width = width,
         dpi = dpi)
}

# Calculate QC metrics
calculate_qc_metrics <- function(seurat_obj) {
  seurat_obj$nCount_RNA <- colSums(seurat_obj@assays$RNA@layers$counts)
  seurat_obj$nFeature_RNA <- colSums(seurat_obj@assays$RNA@layers$counts != 0)
  seurat_obj[["mito.genes"]] <- PercentageFeatureSet(seurat_obj,
                                                     pattern = "^MT-")
  seurat_obj[["ribo.genes"]] <- PercentageFeatureSet(seurat_obj,
                                                     pattern = "^RPS|RPL-")
  return(seurat_obj)
}

# Plot QC metrics
plot_qc_metrics <- function(seurat_obj, file_name, qc_dir) {
  p <- VlnPlot(
    object = seurat_obj,
    features = c("nCount_RNA", "nFeature_RNA", "mito.genes", "ribo.genes"),
    pt.size = 0,
    cols = "grey"
  ) +
```

```
    theme_minimal() +
    theme(legend.position = "none") +
    plot_layout(ncol = 2)

  save_plot(p,
            file.path(qc_dir, paste0(file_name, "_metrics.pdf")),
            height = 8,
            width = 8)
}

# Filter cells based on QC metrics
filter_cells <- function(seurat_obj, file_name, qc_dir) {
  standev <- sd(log(seurat_obj$nFeature_RNA)) * 2
  mean_val <- mean(log(seurat_obj$nFeature_RNA))
  cut <- round(exp(standev + mean_val))

  p <- FeatureScatter(seurat_obj,
                      feature1 = "nCount_RNA",
                      feature2 = "nFeature_RNA") +
    geom_hline(yintercept = cut)

  save_plot(p, file.path(qc_dir,
                         paste0(file_name, "_cutpoint.pdf")),
            height = 8,
            width = 8)

  seurat_obj <- subset(seurat_obj,
                       subset = mito.genes < 10 & nFeature_RNA < cut)
  return(seurat_obj)
}
```

Plot Standardization

```
custom_theme <- theme_minimal(base_size = 8) +
  theme(
    axis.title = element_text(size = 8, color = "black"),
    axis.text = element_text(size = 6, color = "black"),
    legend.text = element_text(size = 4, color = "black"),
    legend.key.size = unit(0.1, "cm"),
    legend.title = element_text(size = 4, color = "black"),
```

```
    legend.position = c(1, 1),
    legend.justification = c("right", "top"),
    legend.box.just = "right",
    legend.margin = margin(6, 6, 6, 6),
    legend.spacing.y = unit(2, "cm")
  )
```

```
Warning: A numeric `legend.position` argument in `theme()` was deprecated in ggplot2
3.5.0.
i Please use the `legend.position.inside` argument of `theme()` instead.
```

Processing Sequencing Runs

```
# Setup
dir.create("./qc", showWarnings = FALSE)
file_list <- list.files("./inputs/data/GSE169440", full.names = FALSE)

# Helper functions
calculate_qc_metrics <- function(seurat_obj) {
  seurat_obj$nCount_RNA <- colSums(seurat_obj@assays$RNA@layers$counts)
  seurat_obj$nFeature_RNA <- colSums(seurat_obj@assays$RNA@layers$counts != 0)
  seurat_obj[["mito.genes"]] <- PercentageFeatureSet(seurat_obj,
                                                     pattern = "^MT-")
  seurat_obj[["ribo.genes"]] <- PercentageFeatureSet(seurat_obj,
                                                     pattern = "^RPS|RPL-")
  return(seurat_obj)
}

plot_qc_metrics <- function(seurat_obj, file_name) {
  p <- VlnPlot(
    object = seurat_obj,
    features = c("nCount_RNA", "nFeature_RNA", "mito.genes", "ribo.genes"),
    pt.size = 0,
    cols = "grey"
  ) +
    theme(legend.position = "none") +
    plot_layout(ncol = 2)
  ggsave(paste0("./qc/", file_name, "_metrics.pdf"),
         plot = p,
         height = 8,
```

```
         width = 8)
}

filter_cells <- function(seurat_obj, file_name) {
  standev <- sd(log(seurat_obj$nFeature_RNA)) * 2
  mean_val <- mean(log(seurat_obj$nFeature_RNA))
  cut <- round(exp(standev + mean_val))

  p <- FeatureScatter(seurat_obj,
                      feature1 = "nCount_RNA",
                      feature2 = "nFeature_RNA") +
    geom_hline(yintercept = cut)
  ggsave(paste0("./qc/", file_name, "_cutpoint.pdf"),
         plot = p,
         height = 8,
         width = 8)

  seurat_obj <- subset(seurat_obj,
                       subset = mito.genes < 10 & nFeature_RNA < cut)
  return(seurat_obj)
}

annotate_with_singler <- function(sce, ref, ref_name) {
  com.res <- SingleR(sce,
                     ref = ref,
                     labels = ref$label.fine,
                     assay.type.test = 1)
  df <- data.frame("labels" = com.res$labels,
                   "pruned.labels" = com.res$pruned.labels)
  rownames(df) <- rownames(com.res)
  colnames(df) <- paste0(ref_name, ".", colnames(df))
  return(df)
}

# Download and prepare reference datasets if not already present
if (!exists("HPCA")) {
  HPCA <- celldex::HumanPrimaryCellAtlasData()
}
if (!exists("Monaco")) {
  Monaco <- celldex::MonacoImmuneData()
}
```

```r
# Main processing loop
for (file in file_list) {
  message("Processing ", file)

  # 1. Read Data and Create Seurat Object
  tmp <- Read10X(paste0("./inputs/data/GSE169440/",
                        file, "/filtered_feature_bc_matrix/"))

  SeuratObj <- CreateSeuratObject(counts = tmp,
                                  assay = "RNA",
                                  project = file) %>%
    subset(subset = nFeature_RNA > 100) # Filter out low feature cells early

  # Remove tmp to free up memory
  rm(tmp)

  # 2. Calculate QC Metrics and Rename Cells
  SeuratObj <- calculate_qc_metrics(SeuratObj)
  SeuratObj <- RenameCells(SeuratObj, new.names =
                             paste0(file, "_", colnames(SeuratObj)))

  # 3. Plot QC Metrics
  plot_qc_metrics(SeuratObj, file)

  # 4. Filter Cells
  SeuratObj <- filter_cells(SeuratObj, file)

  # 5. Estimate Doublets
  sce <- as.SingleCellExperiment(SeuratObj)
  sce <- scDblFinder(sce)
  doublets <- data.frame(db.class = sce$scDblFinder.class,
                         db.score = sce$scDblFinder.score)
  rownames(doublets) <- rownames(sce@colData)
  SeuratObj <- AddMetaData(SeuratObj, doublets)

  # 6. Azimuth Annotation
  SeuratObj <- RunAzimuth(SeuratObj, reference = "pbmcref", verbose = FALSE)

  # 7. SingleR Annotation
  SeuratObj <- AddMetaData(SeuratObj,
                           annotate_with_singler(sce, HPCA, "HPCA"))
  SeuratObj <- AddMetaData(SeuratObj,
```

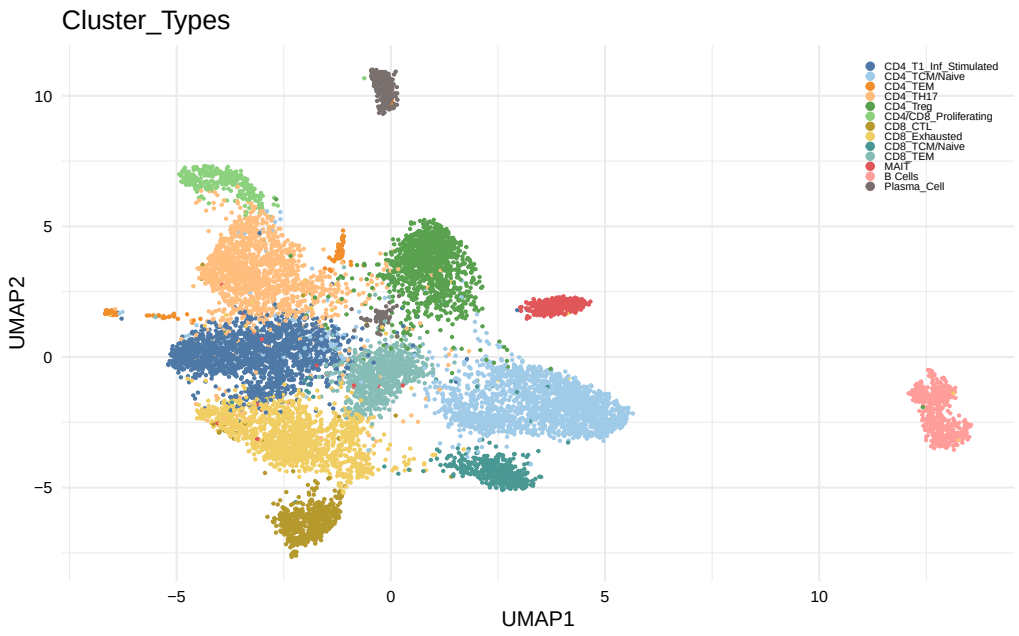
```
                          annotate_with_singler(sce, Monaco, "Monaco"))

  rm(sce)

  # 8. Add Clonal Information
  TCR.file <- list.files(paste0("./inputs/data/GSE169440/",
                                file, "/TCR"), pattern = "annotations")[1]
  TCR.file <- read.csv(paste0("./inputs/data/GSE169440/",
                              file, "/TCR/", TCR.file))
  combinedTCR <- combineTCR(TCR.file, samples = file, filterMulti = TRUE)

  BCR.file <- list.files(paste0("./inputs/data/GSE169440/",
                                file, "/BCR"), pattern = "annotations")[1]
  BCR.file <- read.csv(paste0("./inputs/data/GSE169440/",
                              file, "/BCR/", BCR.file))
  combinedBCR <- combineBCR(BCR.file, samples = file)

  SeuratObj <- combineExpression(c(combinedTCR, combinedBCR),
                                 SeuratObj,
                                 cloneCall = "strict",
                                 proportion = TRUE)

  # 9. scGate Filtering
  suppressWarnings({
    scGateModelDb <- get_scGateDB("data/scGateDB")
  })
  SeuratObj <- scGate(SeuratObj, scGateModelDb$human$generic)

  cells.to.keep <- which(SeuratObj$is.pure_Tcell == "Pure" |
                           SeuratObj$is.pure_Bcell == "Pure" |
                           SeuratObj$is.pure_PlasmaCell == "Pure")
  clones.recovered <- which(!is.na(SeuratObj$CTaa))
  cells.to.keep <- intersect(cells.to.keep, clones.recovered)
  SeuratObj <- subset(SeuratObj, cells = colnames(SeuratObj)[cells.to.keep])

  # 10. Save Preliminary Seurat Object
  saveRDS(SeuratObj, paste0("./inputs/data/processedData/", file, ".rds"))
  rm(SeuratObj)
  gc()
}
```

Integrate Cohort

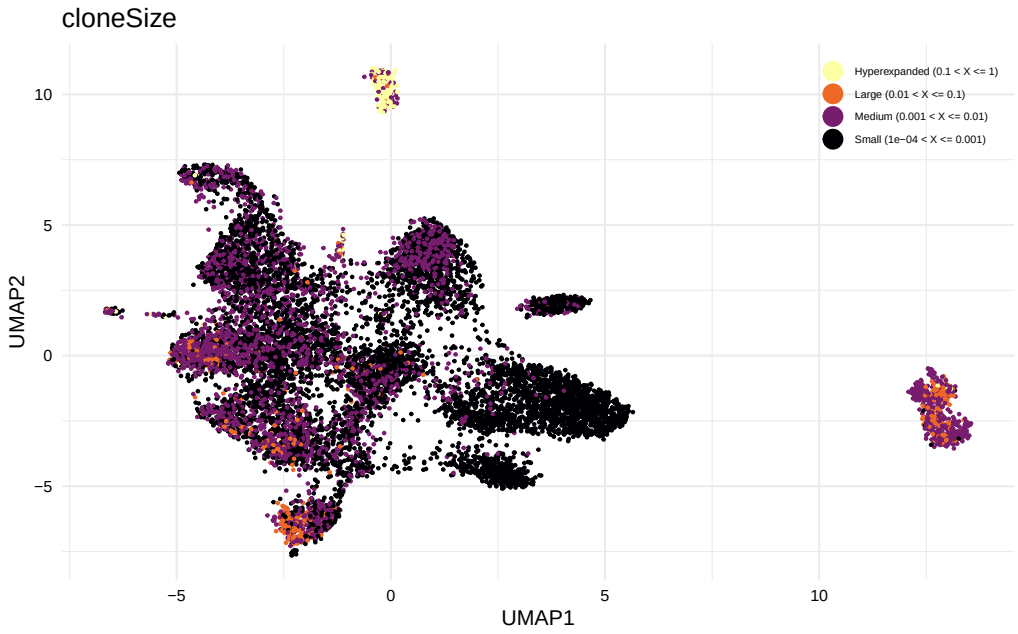
```
dir.create("./output", showWarnings = FALSE)
# 1. Read and merge data
files <- list.files("./inputs/data/processedData/", full.names = TRUE)

object.list <- lapply(files, function(file) {
  readRDS(file)
})

# Use reduce and merge for a more efficient merge
object.merge <- Reduce(function(x, y) merge(x, y), object.list)

# 2. Preprocessing and Integration
object.merge <- object.merge %>%
  NormalizeData(verbose = FALSE) %>%
  FindVariableFeatures(nfeatures = 2500, verbose = FALSE) %>%
  Ibex::quietBCRgenes() %>% # Corrected function call using quietly
  Trex::quietTCRgenes() %>% # Corrected function call using quietly
  ScaleData(verbose = FALSE,
            vars.to.regress = c("mito.genes")) %>%
  RunPCA(verbose = FALSE) %>%
  RunHarmony("orig.ident", verbose = FALSE)

# 3. Clustering and Visualization
object.merge <- object.merge %>%
  RunUMAP(reduction = "harmony",
          dims = 1:30,
          reduction.name = "umap.harmony",
          verbose = FALSE) %>%
  FindNeighbors(dims = 1:30,
                reduction = "harmony",
                verbose = FALSE) %>%
  FindClusters(algorithm = 4,
               resolution = 0.4,
               verbose = FALSE)

# 4. Defining Cell Types By Cluster
cluster.types <- c("CD4_TCM/Naive",
                   "CD4_T1_Inf_Stimulated",
                   "CD4_TH17",
                   "CD8_Exhausted",
```

```
                    "CD4_Treg",
                    "CD8_TEM",
                    "CD8_CTL",
                    "B Cells",
                    "CD8_TCM/Naive",
                    "CD4/CD8_Proliferating",
                    "MAIT",
                    "Plasma_Cell",
                    "CD4_TEM",
                    "Plasma_Cell",
                    "CD4_TEM")
cluster.types <- setNames(cluster.types, as.character(1:15))
object.merge$Cluster_Types <- recode(as.character(object.merge$seurat_clusters),
                                     !!!cluster.types)

#5. Add patient meta data
object.merge$Patient <- str_remove_all(object.merge$orig.ident, "SKN")
object.merge$Patient <- str_remove_all(object.merge$Patient, "SKL")


#6. Plotting UMAP by Cell type and Clonal Expansion
celltype.palette <- tableau_color_pal(palette = "Tableau 20")(length(unique(object.merge$Clus
object.merge$Cluster_Types <- factor(object.merge$Cluster_Types,
                                     sort(unique(object.merge$Cluster_Types))[c(2:12,1,13)])

plot1 <- DimPlot(object.merge, group.by = "Cluster_Types", pt.size = 0.1) +
                  scale_color_manual(values = celltype.palette) +
                  theme_minimal() +
                  custom_theme +
                  guides(colour = guide_legend(override.aes = list(size=1))) +
                  ylab("UMAP2") +
                  xlab("UMAP1")

plot2 <- DimPlot(object.merge, group.by = "cloneSize") +
                  scale_color_viridis(option = "inferno",
                                      discrete = TRUE,
                                      direction = -1) +
                  theme_minimal() +
                  custom_theme +
                  ylab("UMAP2") +
                  xlab("UMAP1")
```

```
plot1
```




```
ggsave("output/Figure2A.pdf", height = 3, width = 3)

plot2
```



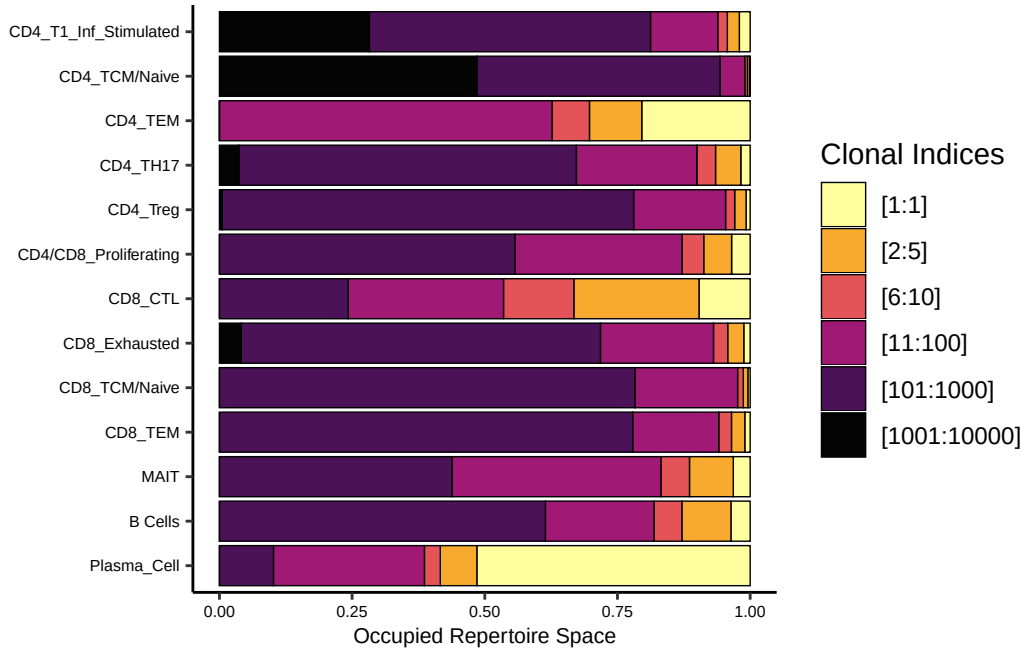

```
ggsave("output/Figure2B_P1.pdf", height = 3, width = 3, dpi = 300)

# 6. Save the integrated object
saveRDS(object.merge, "./inputs/data/IntegratedSeuratObject.rds")
```

Clonal Analysis

Clonal Proportion

```
object.merge <- readRDS("./inputs/data/IntegratedSeuratObject.rds")

plot1 <- clonalProportion(object.merge,
                          group.by = "Cluster_Types",
                  cloneCall = "strict",
                  clonalSplit = c(1, 5, 10, 100, 1000, 10000)) +
                  coord_flip() +
  scale_x_discrete(limits=rev) +
  theme(axis.title.y = element_blank(),
        axis.title = element_text(size=8, color = "black"),
        axis.text = element_text(size=6, color = "black"))
```

```
plot1
```




```
ggsave("output/Figure2B_P2.pdf", height = 2, width = 3)
```

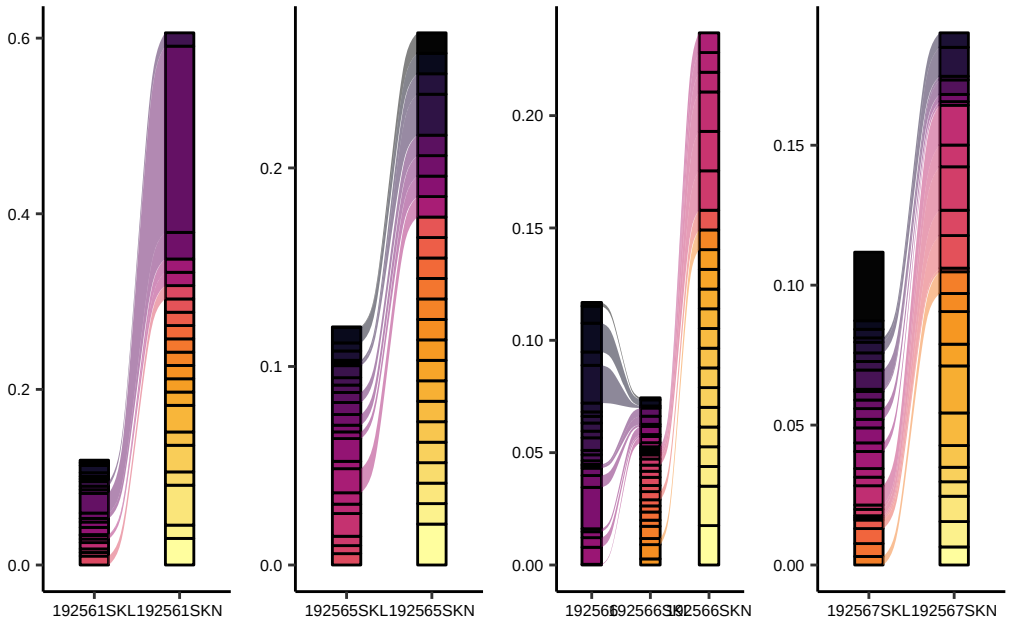
Seperating B and T Cells

```
T.types <- c("CD4_TCM/Naive", "CD4_T1_Inf_Stimulated","CD4_TH17",
             "CD8_Exhausted", "CD4_Treg", "CD8_TEM", "CD8_CTL",
             "CD8_TCM/Naive", "CD4/CD8_Proliferating", "MAIT",
             "CD4_TEM", "CD4_TEM")
object.TCells <- subset(object.merge, Cluster_Types %in% T.types)
object.BCells <- subset(object.merge, Cluster_Types %!in% T.types)
```

Comparing Clones

```
plot1 <- clonalCompare(object.TCells,
              cloneCall = "strict",
```

```
              top.clones = 20,
              samples = c("192561SKL", "192561SKN"),
              group.by = "orig.ident",
              palette = "inferno") +
  guides(fill = "none") +
  theme(axis.title = element_blank(),
        axis.text = element_text(size=6, color = "black"))

plot2 <- clonalCompare(object.TCells,
              cloneCall = "strict",
              top.clones = 20,
              samples = c("192565SKL", "192565SKN"),
              group.by = "orig.ident",
              palette = "inferno") +
  guides(fill = "none") +
  theme(axis.title = element_blank(),
        axis.text = element_text(size=6, color = "black"))

plot3 <- clonalCompare(object.TCells,
              cloneCall = "strict",
              top.clones = 20,
              samples = c("192566","192566SKL","192566SKN"),
              group.by = "orig.ident",
              palette = "inferno") +
  guides(fill = "none") +
  theme(axis.title = element_blank(),
        axis.text = element_text(size=6, color = "black"))

plot4 <- clonalCompare(object.TCells,
              cloneCall = "strict",
              top.clones = 20,
              samples = c("192567SKL", "192567SKN"),
              group.by = "orig.ident",
              palette = "inferno") +
  guides(fill = "none") +
  theme(axis.title = element_blank(),
        axis.text = element_text(size=6, color = "black"))

plot1 + plot2 + plot3 + plot4 + plot_layout(ncol = 4)
```

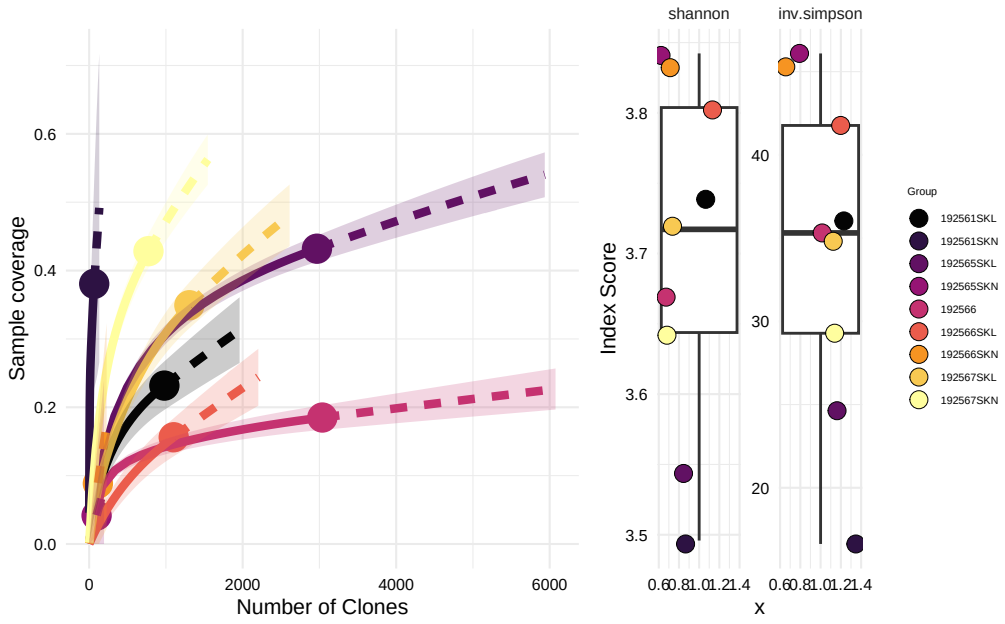



```
ggsave("output/Figure2C.pdf", height = 2, width = 4.25)
```

Clonal Rarefaction

```
plot1 <- clonalRarefaction(object.TCells,
                           group.by = "orig.ident",
                           plot.type = 2,
                           hill.numbers = 1,
                           n.boots = 10) +
                  xlab("Number of Clones") +
                  theme_minimal() +
                  custom_theme +
  guides(color = "none", fill = "none", lty = "none", shape = "none")

plot2 <- clonalDiversity(object.TCells,
                 group.by = "orig.ident",
                 metrics = c("shannon", "inv.simpson"),
                 cloneCall = "strict",
                 n.boots = 100) +
               theme_minimal() +
               custom_theme
```

```
plot1 + plot2 + plot_layout(ncol = 2, guides = "collect", widths = c(2.5, 1))
```



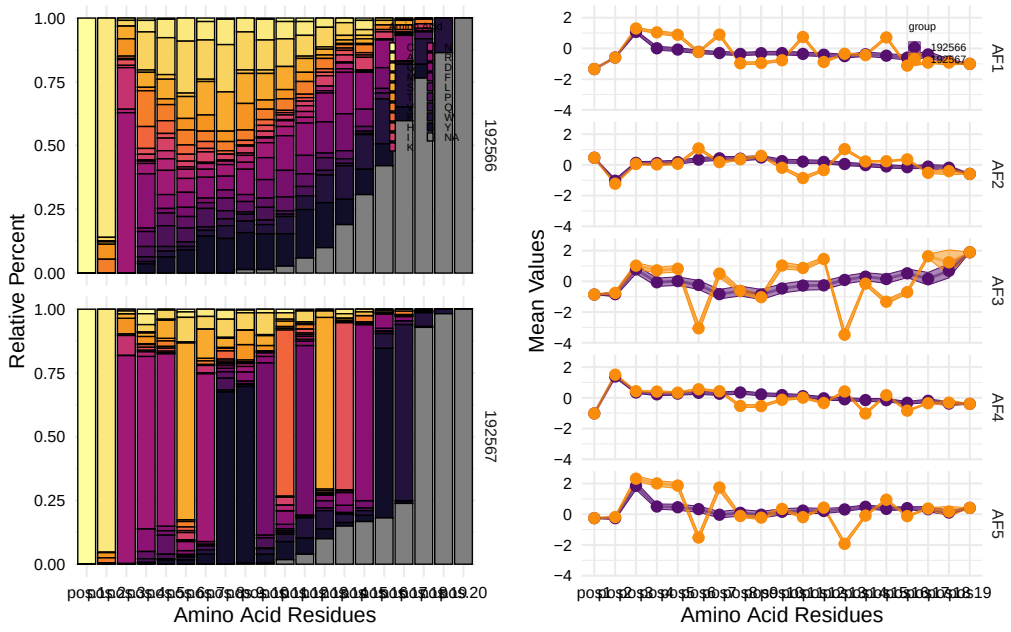

```
ggsave("output/Figure2D.pdf", height = 4, width = 8)
```

Amino Acid Summarization

```
plot1 <- percentAA(subset(object.BCells, Patient %in% c("192566", "192567")),
                        chain = "IGH",
                        group.by = "Patient") +
              custom_theme

plot2 <- positionalProperty(subset(object.BCells, Patient %in%
                                     c("192566", "192567")),
                        chain = "IGH",
                        group.by = "Patient") +
              custom_theme +
              scale_color_manual(values = viridis_pal(option = "inferno")(5)[c(2,4)]) +
              scale_fill_manual(values = viridis_pal(option = "inferno")(5)[c(2,4)])
```

```
Scale for colour is already present.
Adding another scale for colour, which will replace the existing scale.
Scale for fill is already present.
Adding another scale for fill, which will replace the existing scale.
```

```
plot1 + plot2 + plot_layout(ncol = 2)
```

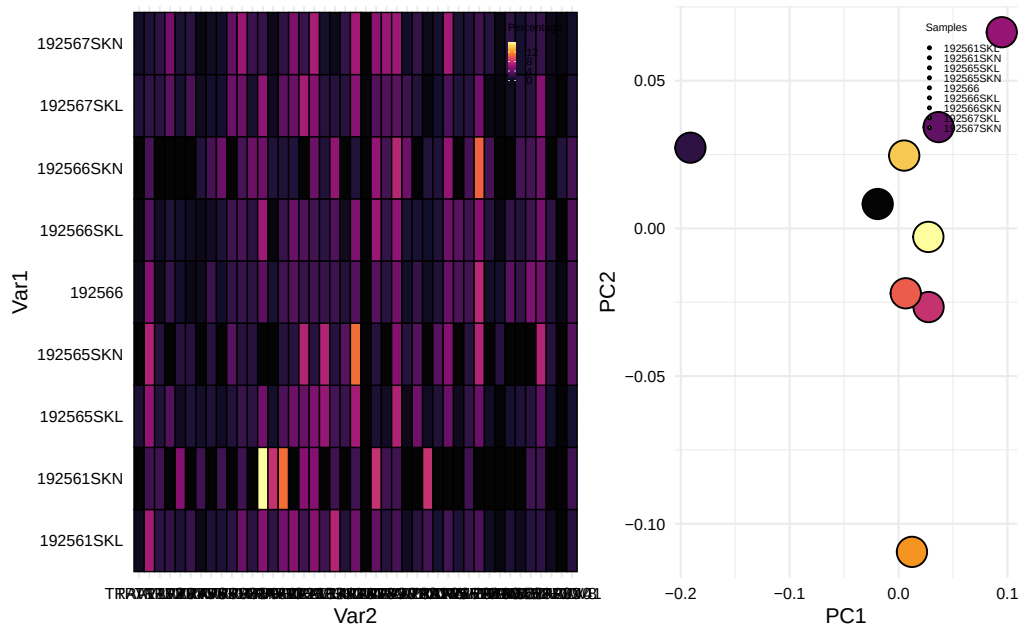



```
ggsave("output/Figure2E.pdf", height = 2, width = 8)
```

Percent Genes

```
plot1 <- percentGenes(object.TCells,
                      gene = "V",
                      chain = "TRA",
                      group.by = "orig.ident") +
             custom_theme

df.genes <- percentGenes(object.TCells,
                         gene = "V",
                         chain = "TRA",
```

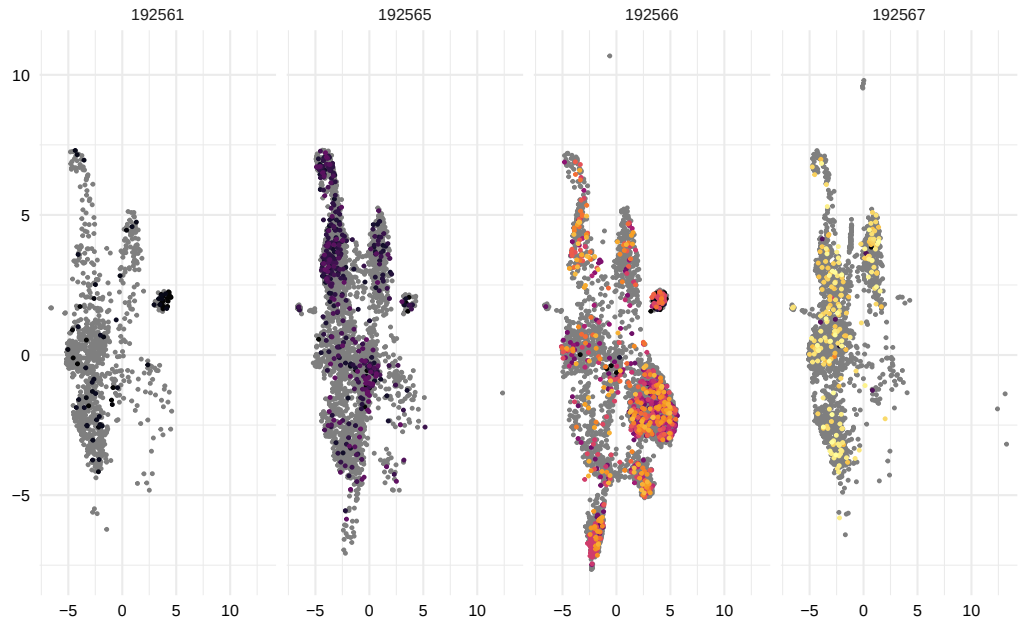
```
                          group.by = "orig.ident",
                          exportTable = TRUE)

#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
plot2 <- ggplot(df, aes(x = PC1, y = PC2)) +
                 geom_point(aes(fill =df[,3]), shape = 21, size = 5) +
                 scale_fill_manual(values = hcl.colors(nrow(df), "inferno")) +
                 theme_minimal() +
                 ylab("PC2") +
                 xlab("PC1") +
                custom_theme +
                guides(fill = guide_legend(title="Samples",
                                           override.aes = list(size = 0.2)))

plot1 + plot2 + plot_layout(ncol = 2,
                                             widths = c(1.3, 1))
```




```
ggsave("output/Figure2F.pdf", height = 4, width = 9)
```

Clustering Clones

```
object.TCells <- clonalCluster(object.TCells,
                               chain = "TRA",
                               sequence = "aa",
                               threshold = 0.85,
                               group.by = "Patient")

DimPlot(object.TCells,
        group.by = "TRA_cluster",
        pt.size = 0.2,
        split.by = "Patient",
        order = TRUE) +
    scale_color_manual(values =  hcl.colors(n=length(unique(object.TCells@meta.data[,"TRA_clu
 guides(color = "none") +
  custom_theme +
  theme(plot.title = element_blank(),
        axis.title = element_blank())
```




```
ggsave("output/Figure2G_P1.pdf", height = 3, width = 9)
```

```
# Define a function for generating and plotting the network
plot_clonal_network <- function(patient_id, color_index, global_vertex_scale) {
  igraph_object <- clonalCluster(
    subset(object.TCells, Patient == patient_id),
    chain = "TRA",
    sequence = "aa",
    group.by = "Patient",
    threshold = 0.85,
    exportGraph = TRUE
  )

  # Generate color for the vertices
  col_samples <- hcl.colors(4, "inferno")[color_index]

  # Standardize vertex sizes
  vertex_sizes <- igraph::V(igraph_object)$size * global_vertex_scale

  # Plot the network with Fruchterman-Reingold layout
  plot(
    igraph_object,
    layout          = layout_with_fr(igraph_object),
```

```
    vertex.size    = vertex_sizes,
    vertex.label   = NA,
    edge.arrow.size = 0.05,
    vertex.color   = col_samples,
    main           = paste(patient_id)
  )
}

# Define patient IDs and color indices
patient_ids <- c("192561", "192565", "192566", "192567")
color_indices <- 1:4

# Calculate a global scaling factor for vertex sizes
all_sizes <- unlist(lapply(patient_ids, function(pid) {
  graph <- clonalCluster(
    subset(object.TCells, Patient == pid),
    chain = "TRA",
    sequence = "aa",
    group.by = "Patient",
    threshold = 0.85,
    exportGraph = TRUE
  )
  sqrt(igraph::V(graph)$size)
}))
global_vertex_scale <- 5 / max(all_sizes)  # Scale so max size is 5

pdf("output/Figure2G_P2.pdf", height = 3, width = 9)
# Set up the plotting grid
par(mfrow = c(1,4), mar = c(1, 1, 2, 1))
for (i in seq_along(patient_ids)) {
  plot_clonal_network(patient_ids[i], color_indices[i], global_vertex_scale)
}
dev.off()
```

```
pdf
  2
```

Conclusion

```
sessionInfo()
```

```
R version 4.4.2 (2024-10-31)
Platform: aarch64-apple-darwin20
Running under: macOS Sonoma 14.0

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  L

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/Chicago
tzcode source: internal

attached base packages:
[1] stats4    stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] viridis_0.6.5               viridisLite_0.4.2
 [3] stringr_1.5.1               SingleR_2.8.0
 [5] pbmcref.SeuratData_1.0.0    SeuratData_0.2.2.9001
 [7] Seurat_5.1.0                SeuratObject_5.0.2
 [9] sp_2.1-4                    scran_1.34.0
[11] scuttle_1.16.0              scRepertoire_2.2.1
[13] scGate_1.6.2                scDblFinder_1.20.0
[15] SingleCellExperiment_1.28.1 RColorBrewer_1.1-3
[17] patchwork_1.3.0             igraph_2.1.2
[19] harmony_1.2.3               Rcpp_1.0.13-1
[21] ggthemes_5.1.0              ggplot2_3.5.1
[23] dplyr_1.1.4                 celldex_1.16.0
[25] SummarizedExperiment_1.36.0 Biobase_2.66.0
[27] GenomicRanges_1.58.0        GenomeInfoDb_1.42.0
[29] IRanges_2.40.0              S4Vectors_0.44.0
[31] BiocGenerics_0.52.0         MatrixGenerics_1.18.0
[33] matrixStats_1.4.1           BiocParallel_1.40.0
[35] Azimuth_0.5.0               shinyBS_0.61.1

loaded via a namespace (and not attached):
```

```
 [1] hash_2.2.6.3                ica_1.0-3
 [3] plotly_4.10.4               scater_1.34.0
 [5] zlibbioc_1.52.0             tidyselect_1.2.1
 [7] bit_4.5.0.1                 lattice_0.22-6
 [9] rjson_0.2.23                evmix_2.12
[11] blob_1.2.4                  S4Arrays_1.6.0
[13] parallel_4.4.2              seqLogo_1.68.0
[15] png_0.1-8                   cli_3.6.3
[17] ProtGenerics_1.34.0         goftest_1.2-3
[19] gargle_1.5.2                textshaping_0.4.1
[21] BiocIO_1.12.0               bluster_1.16.0
[23] purrr_1.0.2                 BiocNeighbors_2.0.0
[25] Signac_1.14.0               stringdist_0.9.14
[27] uwot_0.2.2                  curl_6.0.1
[29] mime_0.12                   evaluate_1.0.1
[31] leiden_0.4.3.1              stringi_1.8.4
[33] gsl_2.1-8                   XML_3.99-0.17
[35] httpuv_1.6.15               AnnotationDbi_1.68.0
[37] magrittr_2.0.3              rappdirs_0.3.3
[39] splines_4.4.2               RcppRoll_0.3.1
[41] ggraph_2.2.1                DT_0.33
[43] sctransform_0.4.1           ggbeeswarm_0.7.2
[45] DBI_1.2.3                   HDF5Array_1.34.0
[47] withr_3.0.2                 systemfonts_1.1.0
[49] tfruns_1.5.3                xgboost_1.7.8.1
[51] lmtest_0.9-40               tidygraph_1.3.1
[53] Trex_0.99.12                rtracklayer_1.66.0
[55] BiocManager_1.30.25         htmlwidgets_1.6.4
[57] fs_1.6.5                    biomaRt_2.58.2
[59] ggrepel_0.9.6               labeling_0.4.3
[61] SparseArray_1.6.0           cellranger_1.1.0
[63] annotate_1.84.0             reticulate_1.40.0
[65] zoo_1.8-12                  JASPAR2020_0.99.10
[67] XVector_0.46.0              knitr_1.49
[69] TFBSTools_1.44.0            UCSC.utils_1.2.0
[71] RhpcBLASctl_0.23-42         TFMPvalue_0.0.9
[73] caTools_1.18.3              grid_4.4.2
[75] data.table_1.16.4           rhdf5_2.50.0
[77] pwalign_1.2.0               quantreg_5.99.1
[79] R.oo_1.27.0                 poweRlaw_0.80.0
[81] RSpectra_0.16-2             irlba_2.3.5.1
[83] alabaster.schemas_1.6.0     fastDummies_1.7.4
[85] lazyeval_0.2.2              yaml_2.3.10
```

```
 [87] survival_3.8-3                scattermore_1.2
 [89] BiocVersion_3.20.0            crayon_1.5.3
 [91] RcppAnnoy_0.0.22              tidyr_1.3.1
 [93] progressr_0.15.1              tweenr_2.0.3
 [95] later_1.4.1                   ggridges_0.5.6
 [97] codetools_0.2-20              base64enc_0.1-3
 [99] KEGGREST_1.46.0               Rtsne_0.17
[101] limma_3.62.1                  Rsamtools_2.22.0
[103] filelock_1.0.3                pkgconfig_2.0.3
[105] xml2_1.3.6                    spatstat.univar_3.1-1
[107] GenomicAlignments_1.42.0      iNEXT_3.0.1
[109] evd_2.3-7.1                   spatstat.sparse_3.1-0
[111] BSgenome_1.74.0               alabaster.base_1.6.1
[113] xtable_1.8-4                  plyr_1.8.9
[115] httr_1.4.7                    tools_4.4.2
[117] globals_0.16.3                beeswarm_0.4.0
[119] nlme_3.1-166                  dbplyr_2.5.0
[121] hdf5r_1.3.11                  ExperimentHub_2.14.0
[123] shinyjs_2.1.0                 MatrixModels_0.5-3
[125] assertthat_0.2.1              digest_0.6.37
[127] Matrix_1.7-1                  farver_2.1.2
[129] tzdb_0.4.0                    AnnotationFilter_1.26.0
[131] reshape2_1.4.4                DirichletMultinomial_1.48.0
[133] glue_1.8.0                    cachem_1.1.0
[135] BiocFileCache_2.14.0          polyclip_1.10-7
[137] generics_0.1.3                Biostrings_2.74.0
[139] ggalluvial_0.12.5             googledrive_2.1.1
[141] presto_1.0.0                  parallelly_1.41.0
[143] statmod_1.5.0                 ragg_1.3.3
[145] RcppHNSW_0.6.0                ScaledMatrix_1.14.0
[147] pbapply_1.7-2                 httr2_1.0.7
[149] spam_2.11-0                   dqrng_0.4.1
[151] graphlayouts_1.2.1            gtools_3.9.5
[153] alabaster.se_1.6.0            gridExtra_2.3
[155] shiny_1.10.0                  GenomeInfoDbData_1.2.13
[157] R.utils_2.12.3                rhdf5filters_1.18.0
[159] RCurl_1.98-1.16               memoise_2.0.1
[161] rmarkdown_2.29                Ibex_0.99.5
[163] scales_1.3.0                  R.methodsS3_1.8.2
[165] googlesheets4_1.1.1           future_1.34.0
[167] gypsum_1.2.0                  RANN_2.6.2
[169] spatstat.data_3.1-4           rstudioapi_0.17.1
[171] cluster_2.1.8                 whisker_0.4.1
```

```
[173] spatstat.utils_3.1-1              hms_1.1.3
[175] fitdistrplus_1.2-1                munsell_0.5.1
[177] cowplot_1.1.3                     colorspace_2.1-1
[179] rlang_1.1.4                       ggdendro_0.2.0
[181] DelayedMatrixStats_1.28.0         sparseMatrixStats_1.18.0
[183] truncdist_1.0-2                   dotCall64_1.2
[185] shinydashboard_0.7.2              ggforce_0.4.2
[187] xfun_0.49                         alabaster.matrix_1.6.1
[189] CNEr_1.42.0                       keras_2.15.0
[191] abind_1.4-8                       tibble_3.2.1
[193] EnsDb.Hsapiens.v86_2.99.0         Rhdf5lib_1.28.0
[195] readr_2.1.5                       bitops_1.0-9
[197] promises_1.3.2                    RSQLite_2.3.9
[199] DelayedArray_0.32.0               GO.db_3.18.0
[201] compiler_4.4.2                    alabaster.ranges_1.6.0
[203] prettyunits_1.2.0                 beachmat_2.22.0
[205] SparseM_1.84-2                    listenv_0.9.1
[207] BSgenome.Hsapiens.UCSC.hg38_1.4.5 edgeR_4.4.0
[209] AnnotationHub_3.14.0              BiocSingular_1.22.0
[211] tensor_1.5                        MASS_7.3-61
[213] progress_1.2.3                    UCell_2.10.1
[215] cubature_2.1.1                    spatstat.random_3.3-2
[217] R6_2.5.1                          fastmap_1.2.0
[219] fastmatch_1.1-4                   vipor_0.4.7
[221] ensembldb_2.26.0                  ROCR_1.0-11
[223] SeuratDisk_0.0.0.9021             rsvd_1.0.5
[225] gtable_0.3.6                      KernSmooth_2.23-24
[227] miniUI_0.1.1.1                    deldir_2.0-4
[229] htmltools_0.5.8.1                 bit64_4.5.2
[231] spatstat.explore_3.3-3            lifecycle_1.0.4
[233] tensorflow_2.16.0                 restfulr_0.0.15
[235] vctrs_0.6.5                       zeallot_0.1.0
[237] VGAM_1.1-12                       spatstat.geom_3.3-4
[239] future.apply_1.11.3               pracma_2.4.4
[241] pillar_1.10.0                     GenomicFeatures_1.54.4
[243] metapod_1.14.0                    locfit_1.5-9.10
[245] jsonlite_1.8.9
```