Memory persistence and positively selected transition to antibody-secreting subsets in longitudinal lGH repertoires

IGH subset repertoires analysis

Load libraries

```
library(data.table)
library(tidyverse)
library(magrittr)
library(ggpubr)
library(ggbeeswarm)
library(rstatix)
library(cowplot)
library(grid)
library(msa)
library(seqinr)
library(edgeR)
library(ComplexHeatmap)
library(scales)
first=dplyr::first
slice=dplyr::slice
count=dplyr::count
rename=dplyr::rename
options(dplyr.summarise.inform = FALSE)
```

Package versions and R

R 4.0.3 was used to run this code. Package versions:

```
packages_amikelov <- read.delim("package_versions.tsv")
packages_amikelov
```

```
##           package   version
## 1          seqinr  04.02.08
## 2             msa    1.20.1
## 3      Biostrings    2.56.0
## 4         XVector    0.28.0
## 5         IRanges    2.22.2
## 6       S4Vectors    0.26.1
## 7    BiocGenerics    0.34.0
## 8          scales  01.01.01
## 9  ComplexHeatmap  02.04.03
## 10          edgeR    3.30.3
## 11          limma    3.44.3
## 12        cowplot  01.01.01
## 13        rstatix     0.7.0
```

```
## 14     ggbeeswarm     0.6.0
## 15         ggpubr 0.4.0.999
## 16       magrittr     2.0.1
## 17        forcats     0.5.1
## 18        stringr  01.04.00
## 19          dplyr     1.0.6
## 20          purrr     0.3.4
## 21          readr  01.04.00
## 22          tidyr  01.01.03
## 23         tibble  03.01.01
## 24        ggplot2  03.03.05
## 25      tidyverse  01.03.00
## 26     data.table    1.13.4
```

Define helper functions

```
`%+%` <- function(a, b) paste0(a, b)

extract_id<- function(x) str_remove_all(x,".*/*.*/|\\..*")

freq<-function(db){
  db %>% mutate(freq=n/sum(n))
}
```

Load data - original, in silico generated and naive from Gidoni et al.'19

```
subset_db<-read_tsv("IGH_subsets_db_preprocessed.tsv")
naive_db<-read_tsv("Gidoni19_naive.tsv")
olga_db_5k<-read_tsv("OLGA_insilico_5K.tsv")
```

Fig 1B - isotype frequencies

```
  isotype_freq<-subset_db %>%
 # bind_rows(pbmc_db) %>%
    group_by(sampleId,donor,subpop,repnum,isotype) %>%
    summarize(nClonesIsotype=n(),nUMIsIsotype=sum(cloneCount)) %>%
    group_by(sampleId) %>%
    mutate(freqIsotype=nClonesIsotype/sum(nClonesIsotype),
           freqCountIsotype=nUMIsIsotype/sum(nUMIsIsotype)) %>%
    group_by(subpop,isotype) %>%
    summarise(meanFreqIsotype=mean(freqIsotype),
              meanFreqIsotypeByCount=mean(freqCountIsotype)) %>%
  pivot_longer(c(meanFreqIsotype,meanFreqIsotypeByCount),
               names_to="freqType",
               values_to="IsotypeFrequency")
```

```r
# ggplot unfortunately didn't know how to draw such a plot,
# so I had to do a little trick with adding dummy rows

Fig1B<-isotype_freq %>% mutate(subpop=case_when(subpop=="Bmem" ~"x1",
                                                subpop=="PBL" ~"x2",
                                                subpop=="PL" ~"x3")) %>%
  bind_rows(isotype_freq ) %>%
  mutate(isotype=factor(isotype,
                        levels=c(IgA="IGHA",IgE="IGHE",IgG="IGHG",IgD="IGHD",IgM="IGHM")),
        subpop=factor(subpop,
                      levels=c("Bmem","x1","PBL","x2","PL","x3")),
        color_label=ifelse(grepl("x",subpop),"black",NA ),
        isotype_subpop=factor(paste(str_replace(subpop,"x.","x"),isotype,sep="_"),
                              levels=outer(X =unique(str_replace(subpop,"x.","x")),
                                           Y = c("IGHA","IGHE","IGHG","IGHD","IGHM"),
                                           FUN = paste, sep= "_" ) %>% t() %>% c())
                                     ) %>%
  ggplot(aes(x=subpop,
             y=IsotypeFrequency,
             fill=isotype_subpop))+
  geom_bar(stat="identity",width=0.7)+
  theme(legend.position = "right",
        axis.title.x = element_blank(),
        axis.title.y = element_text(size = 9),
        strip.text.x = element_text(size = 9,margin = margin(0.08,0,0.08,0, "cm")),
        legend.text=element_text(size=8),
        legend.title = element_text(size=8),
        axis.ticks.x=element_blank())+
  labs(y="Mean Isotype Frequency", y = "Subset",fill="Isotype")+
  facet_wrap(~freqType,
             labeller = labeller(freqType=c(meanFreqIsotype="By unique clonotypes",
                                            meanFreqIsotypeByCount="By unique cDNA molecules")))+
  geom_text(aes(label=percent(IsotypeFrequency,0.1),
                color=color_label),
            position = position_stack(vjust = 0.5),
            size=3)+
  scale_color_manual(values=c("black",NA),guide='none',na.value = NA)+
  scale_alpha_continuous( guide='none')+
  scale_fill_manual(values = c("peachpuff3","darkred","cyan4","gray72","royalblue4",
                               "peachpuff3","darkred","cyan4","gray72","royalblue4",
                               "peachpuff3","darkred","cyan4","gray72","royalblue4",
                               NA,NA,NA,NA,NA),
                    labels=c("IgA","IgE","IgG","IgD","IgM"),
                    guide='none',
                    na.value = NA)+
  scale_x_discrete(labels=c("Bmem" = "    Bmem","x1"="",
                            "PBL"="     PBL","x2"="",
                            "PL"="      PL","x3"=""))

fig1b_leg <- tibble(x1 = 1:5,
                    y1 = 1:5,
                    Isotype = factor(
```

```
                      c("IgA", "IgE", "IgG", "IgD", "IgM"),
                      levels = c("IgA", "IgE", "IgG", "IgD", "IgM")
                    ))

fig1b_leg <- ggplot(fig1b_leg, aes(x = as.numeric(x1), y = as.numeric(y1), fill = Isotype)) +
  geom_point( alpha = 0.8, size = 5, shape = 22, color = "grey20") +
  scale_fill_manual("Isotype",
                    values = c("IgA"="peachpuff3",
                               "IgE"="darkred",
                               "IgG"="cyan4",
                               "IgD"="gray72",
                               "IgM"="royalblue4")) +
  theme(legend.position = "right",
                  legend.text = element_text(size=9),
                  legend.title = element_text(size=9))
fig1b_leg <- cowplot::get_legend(fig1b_leg )

Fig1B<-plot_grid(Fig1B,fig1b_leg,rel_widths = c(1,0.08))
Fig1B
```

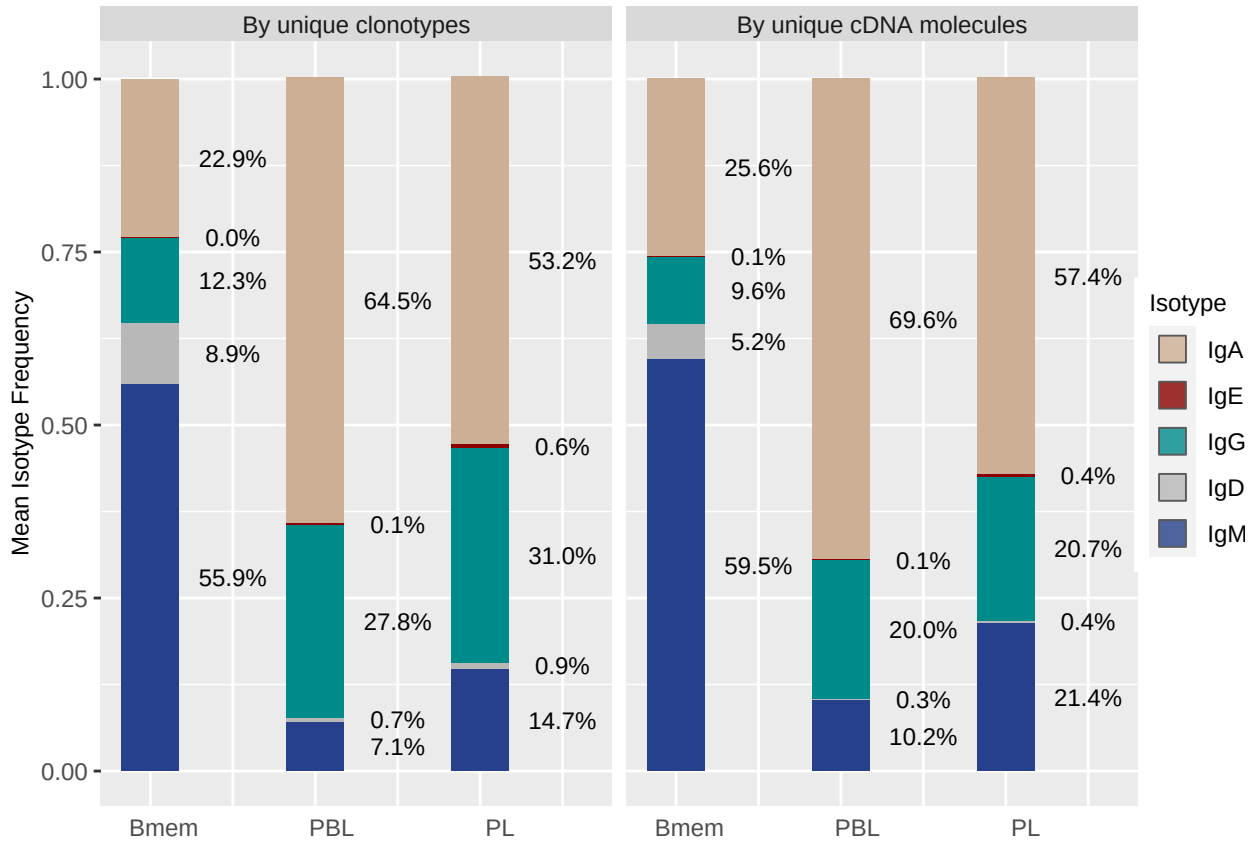


Fig 1C - SHM rate

```
# auxillary functions for statistics calculation on plot
give.n <- function(y){
```

```r
    return(data.frame(
             y = -3,
             label =  paste0('N=', length(y))))
}


give.median <- function(y){
    return(data.frame(
             y = -7,
             label =  paste0('med=', round(median(y),2))))
}

subpop_comparisons<-list(combn(c("Bmem","PBL","PL"),2,c,F))
subpop_comparisons %<>% unlist(recursive = F)

## SHM for all clonotypes - each point is a clonotype
Fig1C<-subset_db %>%
    mutate(isotype=factor(isotype,levels=c("IGHM","IGHD","IGHG","IGHE","IGHA"),
                          labels=c("IgM","IgD", "IgG", "IgE", "IgA")),
           SHM=1-vBestIdentityPercent) %>%
    ggplot(aes(x=subpop, y= SHM*100,fill=subpop))+

    geom_quasirandom(width=0.3, size = 0.5, color = "grey20", alpha = 0.5)+
    geom_boxplot(alpha = 0.8, outlier.colour = NA)+
    facet_wrap(~isotype,nrow=1)+
    theme_bw()+
    scale_fill_manual(values = c("Bmem" = "#273046", "PBL" = "#F8AFA8", "PL" = "#CB2314"),
                      labels = c("B" = "Bmem", "P" = "PBL", "L" =   "PL"),
                      name = "Cell fraction")+
    labs(y="Rate of SHM, # per 100 bp")+
    stat_summary(fun.data = give.n, geom = "text",size=2)+
    stat_summary(fun.data = give.median, geom = "text",size=2)+
    stat_compare_means(comparisons = subpop_comparisons,
                       label = "p.signif",
                       label.y = c(46,52,59))+
    theme(legend.position = "none",
          axis.title.x =  element_blank(),
          axis.title.y = element_text(size = 9),
          strip.text.x = element_text(size = 9,margin = margin(0.08,0,0.08,0, "cm")))+
    ylim(-7,70)

Fig1C
```

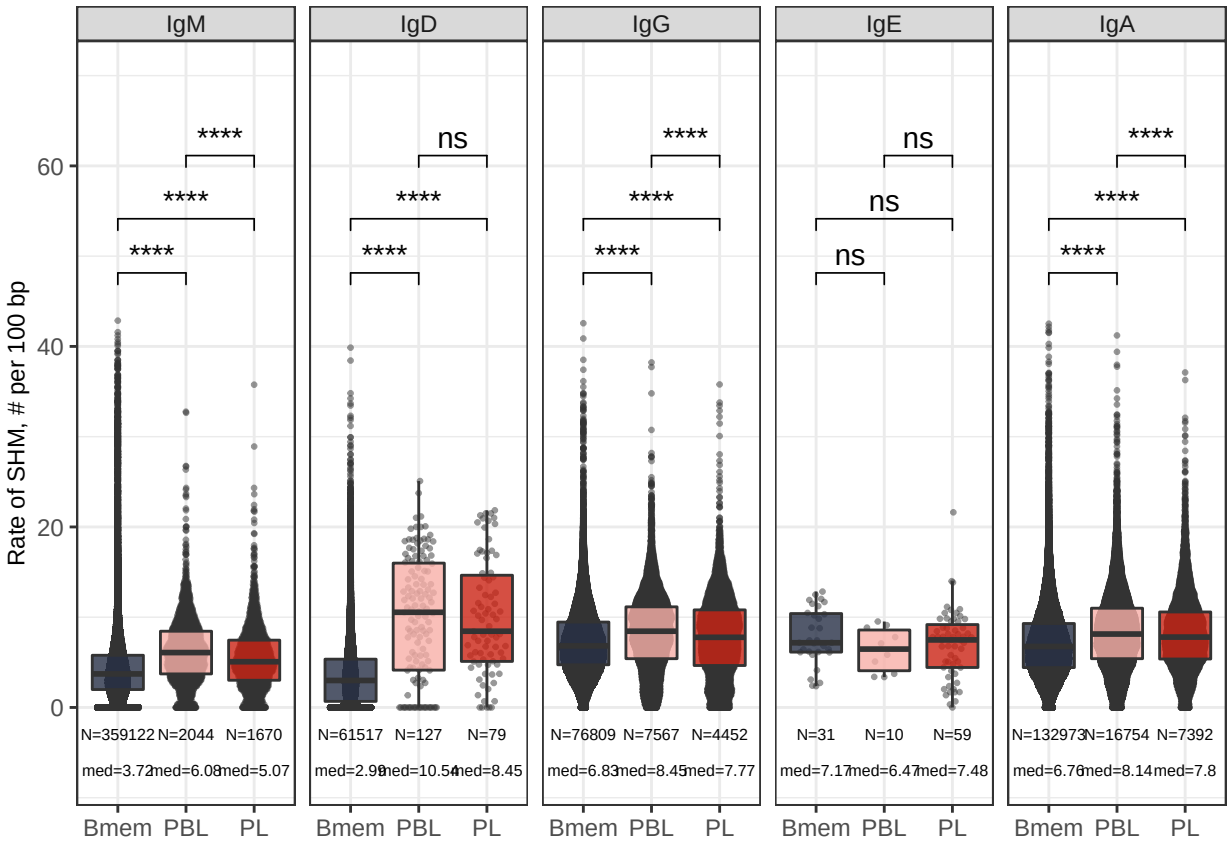
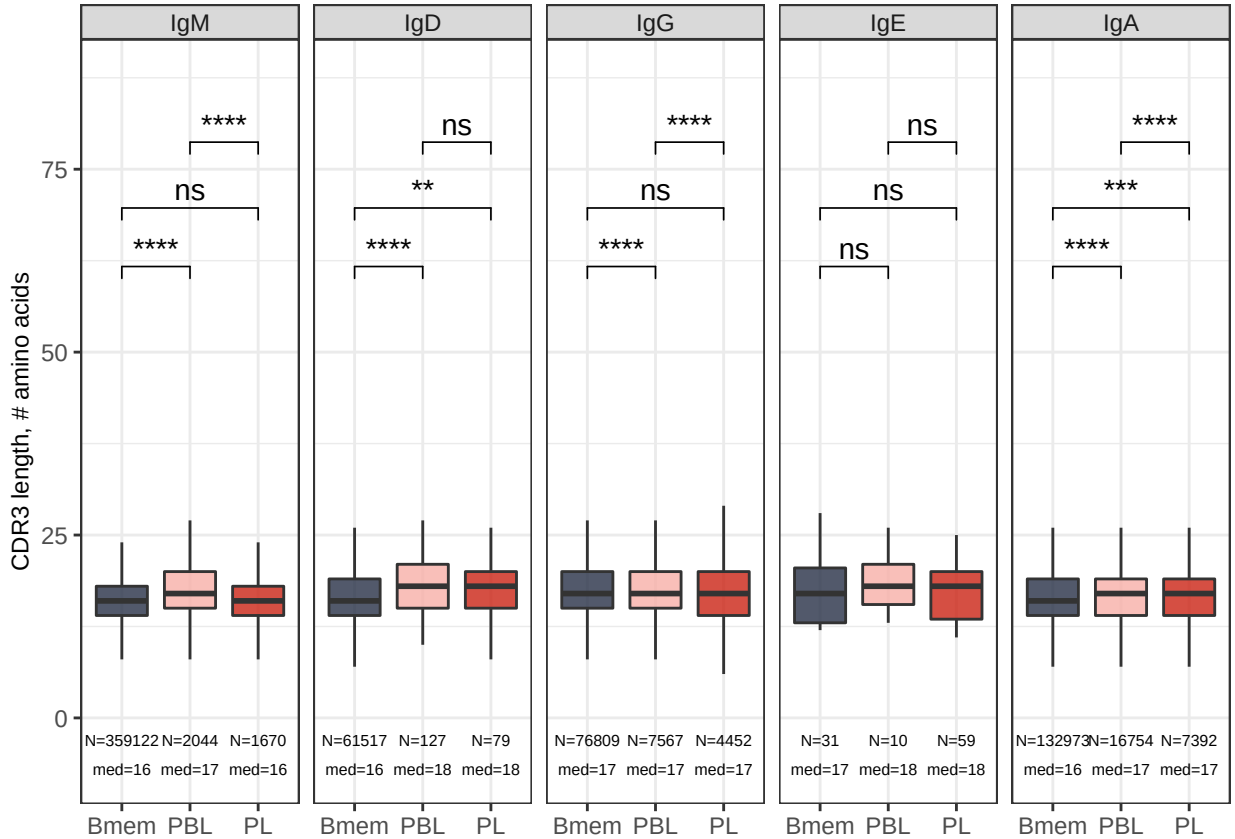


Fig 1D - CDR3 length, each point is a clonotype

```
# auxillary functions for statistics calculation on plot
give.n <- function(y){
    return(data.frame(
             y = -3,
             label = paste0('N=', length(y))))
}

give.median <- function(y){
    return(data.frame(
             y = -7,
             label =  paste0('med=', round(median(y),2))))
}

# points with geom_quasirandom are heavy, so uncomment if you want
# to wait for some time and see figure exactly as it appears in the paper
Fig1D<-subset_db %>%
    mutate(isotype=factor(isotype,levels=c("IGHM","IGHD","IGHG","IGHE","IGHA"),
                          labels=c("IgM","IgD", "IgG", "IgE", "IgA")),
           SHM=1-vBestIdentityPercent,
           len_ntCDR3=nchar(nSeqCDR3)) %>%
    ggplot(aes(x=subpop, y= len_aaCDR3,fill=subpop))+
    #geom_quasirandom(width=0.3, size = 0.5, color = "grey20", alpha = 0.5)+
```

```
    geom_boxplot(alpha = 0.8, outlier.colour = NA)+
    facet_wrap(~isotype,nrow=1)+theme_bw()+
    scale_fill_manual(values = c("Bmem" = "#273046", "PBL" = "#F8AFA8", "PL" = "#CB2314"),
                      labels = c("B" = "Bmem", "P" = "PBL", "L" =   "PL"),
                      name = "Cell fraction")+
    labs(y="CDR3 length, # amino acids")+
    stat_summary(fun.data = give.n, geom = "text",size=2)+
    stat_summary(fun.data = give.median, geom = "text",size=2)+
    stat_compare_means(comparisons = subpop_comparisons,
                       label = "p.signif",
                       label.y = c(59,67,76))+
    theme(legend.position = "none",
          axis.title.x =  element_blank(),
          axis.title.y = element_text(size = 9),
          strip.text.x = element_text(size = 9,margin = margin(0.08,0,0.08,0, "cm")),)+
    ylim(-7,88)
```

Fig1D



Calculate IGHV usage tables

```
vusage<-subset_db %>%
  group_by(sampleId) %>%
  mutate(totalNClones=n(), totalNGroups=n_distinct(clonalGroupId)) %>%
```

```
  ungroup() %>%
  group_by(sampleId,V,repnum,timepoint,donor,subpop) %>%
  summarise(NClones=n(),
            NClones_normalized=n()/first(totalNClones),
            nGroups=n_distinct(clonalGroupId),
            nGroups_normalized= nGroups/first(totalNGroups),
            freq=sum(cloneFraction),
            count=sum(cloneCount))

vusage_naive<- naive_db %>%
  group_by(sampleId) %>% mutate(totalNClones=n()) %>%
  ungroup() %>%
  filter(totalNClones>5000) %>%
  group_by(sampleId,V,repnum,timepoint,donor,subpop) %>%
  summarise(NClones=n(),
            NClones_normalized=n()/first(totalNClones),
            freq=sum(cloneFraction),
            count=sum(cloneCount)) %>% mutate(repnum=as.integer(repnum))


vusage_all<- bind_rows(vusage,vusage_naive)

vusage_all_wide<-vusage_all %>%
  as.data.table() %>%
  dcast(V~donor+subpop+timepoint+repnum,value.var="NClones",fill=0)
de_matrix<-vusage_all_wide%>% select(-V) %>% as.matrix()
rownames(de_matrix)<-vusage_all_wide %>% pull(V)
```

Calculate FDR for Vsegment enrichment in Bmem/PBL/PL vs Naive

```
de_matrix<-vusage_all_wide%>% select(-V) %>% as.matrix()
rownames(de_matrix)<-vusage_all_wide %>% pull(V)

de_matrix <- de_matrix[ rowMeans(de_matrix) >2, ]

      if(nrow(de_matrix)==0) {
        warning(paste(x,"No clones in matrix after filtering"))
        return()
      }

      groups<-colnames(vusage_all_wide)[-1] %>% str_extract("naive|Bmem|PBL|PL")
      dge<-DGEList(counts = de_matrix,
                   group=groups)

      dge<-calcNormFactors(dge)
      dge<-estimateDisp(dge)
```

```
## Design matrix not provided. Switch to the classic mode.
```

```
      all_pairs<-list(c("naive","Bmem"),c("naive","PBL"),c("naive","PL"))
```

```
    de_list<-lapply(all_pairs,function(pair){
      de<-exactTest(dge,pair = pair,dispersion = "trended" )
      de_table<-de$table %>%
        select(logFC,PValue)%>%
        mutate(V=row.names(de$table),
               comparison=paste(pair,collapse="_"),
               FDR=p.adjust(PValue,method = "BH"))
      return(de_table)}) %>% bind_rows()


v_de_naive_pvalue_pooled <- de_list %>%
  separate(comparison,into = c("subpop.x","subpop.y"))  %>%
  mutate(logFC=ifelse(FDR<0.01,logFC,0)) %>%
  dcast(subpop.y~V,value.var="logFC")

v_de_naive_pvalue_pooled %<>%
  mutate_at(vars(starts_with("IGHV")), function(x){
    case_when(x< -1 ~ "log2FC<-1",
              x >= -1 & x< 0 ~ "-1<log2FC<0",
              x<=1 & x>0 ~ "0<log2FC<1",
              x>1 ~ "log2FC>1",
              x==0 ~ "FDR>0.01")
  })
```

Cluster IGH Vgenes by their nt sequence similarity

```
 Vs<-vusage_naive %>%
    group_by(V) %>%
    summarise(freq=mean(freq)) %>%
    ungroup() %>%
    filter(V %in% rownames(de_matrix)) %>%
    pull(V)


refV<-readDNAStringSet(filepath = "IMGT_IGHV.fasta")
names(refV)<-names(refV) %>% str_remove("\\*.*")
Valignments<-msa(refV[Vs]) %>%  msaConvert( type="seqinr::alignment")
d <- dist.alignment(Valignments, "identity")

V_clusters<-hclust(d)
```

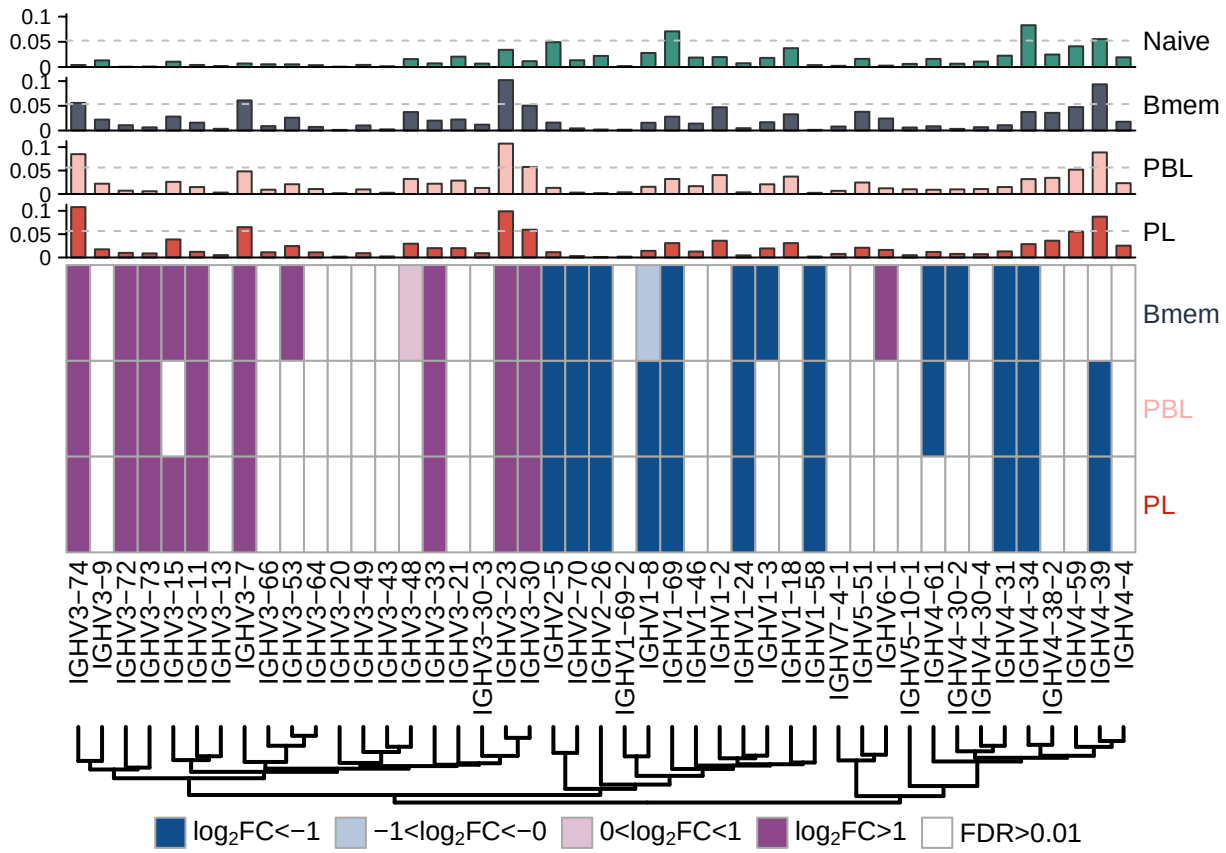
Fig 1E - IGHV gene segment frequencies and enrichment against naive B cells heatmap

```
colors = structure(
  c("dodgerblue4", "#B6C7DD", "#E0C0D3", "orchid4", "white"),
  names = c(
```

```
    "log2FC<-1",
    "-1<log2FC<0",
    "0<log2FC<1",
    "log2FC>1",
    "FDR>0.01"
  )
)


lgd = Legend(labels = c(expression(log[2]*FC*"<-1"),
                        expression("-1<"*log[2]*FC*"<-0"),
                        expression("0<"*log[2]*FC*"<1"),
                        expression(log[2]*FC*">1"),
                        "FDR>0.01"),
             legend_gp = gpar(fill = c("dodgerblue4",
                                       "#B6C7DD",
                                       "#E0C0D3",
                                       "orchid4",
                                       "white")),
             labels_gp = gpar(size=8),
             nrow = 1,
             title_position = "leftcenter",

             border="darkgrey")

naive_vusage<-HeatmapAnnotation(`Naive` = anno_barplot(vusage_naive %>%
                                                       group_by(V) %>%
                                                       summarise(freq=mean(freq)) %>%
                                                       filter(V %in% rownames(de_matrix)) %>%
                                                       pull(freq),
                                                       gp = gpar(fill = "#0B775E",alpha=0.8),
                                                       ylim = c(0,0.1),
                                                       height = unit(7, "mm"),
                                                       border = F),
                                `Bmem` = anno_barplot(vusage %>%
                                                       filter(subpop=="Bmem") %>%
                                                       group_by(V) %>%
                                                       summarise(freq=mean(freq)) %>%
                                                       filter(V %in% rownames(de_matrix)) %>%
                                                       pull(freq),
                                                       gp = gpar(fill = "#273046",alpha=0.8),
                                                       height = unit(7, "mm"),
                                                       border = F),

                                `PBL` = anno_barplot(vusage %>%
                                                       filter(subpop=="PBL") %>%
                                                       group_by(V) %>%
                                                       summarise(freq=mean(freq)) %>%
                                                       filter(V %in% rownames(de_matrix)) %>%
                                                       pull(freq),
                                                       gp = gpar(fill = "#F8AFA8",alpha=0.8),
                                                       height = unit(7, "mm"),
                                                       border = F),
```

```
                                `PL` = anno_barplot(vusage %>%
                                                filter(subpop=="PL") %>%
                                                group_by(V) %>%
                                                summarise(freq=mean(freq)) %>%
                                                filter(V %in% rownames(de_matrix)) %>%
                                                pull(freq),
                                                gp = gpar(fill = "#CB2314",alpha=0.8),
                                                height = unit(7, "mm"),
                                                border = F),

                                annotation_name_gp= gpar(fontsize = 10),
                                gap = unit(1.4, "mm"))



hm<-Heatmap( v_de_naive_pvalue_pooled[,-1] %>%
           set_rownames(c("Bmem","PBL","PL")) %>%
           as.matrix(),
         col=colors,
         rect_gp = gpar(col = "darkgrey", lwd = 1),
         top_annotation  =naive_vusage,
         cluster_columns = V_clusters,
         column_dend_side = "bottom",
         column_dend_gp = gpar(lwd=2),
         column_dend_height = unit(10, "mm"),
         show_heatmap_legend = FALSE,

        row_names_gp =    gpar(col = c("#273046", "#F8AFA8","#CB2314"),fontsize = 10),
          column_split =4 ,
          column_names_gp =    gpar(fontsize = 10),
          column_gap = unit(rep(0,4), "mm"),
          column_title = NULL
         )

draw(hm,heatmap_legend_side = "bottom",heatmap_legend_list = lgd )


decorate_annotation("Naive", {

    pushViewport(viewport(xscale = c(0.5, 10.5), yscale = c(0, 1)))
    grid.lines(c(0.5, 23), c(0, 0), gp = gpar(lty = 1),
               default.units = "native")

    grid.lines(c(0.5, 23), c(0.5, 0.5), gp = gpar(lty = 2,col="grey"),
               default.units = "native")

    popViewport()
})

decorate_annotation("Bmem", {

    pushViewport(viewport(xscale = c(0.5, 10.5), yscale = c(0, 1)))
    grid.lines(c(0.5, 23), c(0, 0), gp = gpar(lty = 1),
```

```
               default.units = "native")

    grid.lines(c(0.5, 23), c(0.5, 0.5), gp = gpar(lty = 2,col="grey"),
               default.units = "native")

    popViewport()
})

decorate_annotation("PBL", {

    pushViewport(viewport(xscale = c(0.5, 10.5), yscale = c(0, 1)))
    grid.lines(c(0.5, 23), c(0, 0), gp = gpar(lty = 1),
               default.units = "native")

    grid.lines(c(0.5, 23), c(0.5, 0.5), gp = gpar(lty = 2,col="grey"),
               default.units = "native")

    popViewport()
})

decorate_annotation("PL", {

    pushViewport(viewport(xscale = c(0.5, 10.5), yscale = c(0, 1)))
    grid.lines(c(0.5, 23), c(0, 0), gp = gpar(lty = 1),
               default.units = "native")

    grid.lines(c(0.5, 23), c(0.5, 0.5), gp = gpar(lty = 2,col="grey"),
               default.units = "native")

    popViewport()
})
```



Calculate Jensen-Shannon divergence between all pairs of repertoires

```
#define a function for calculation of JS between repertoires of top n clonotypes

calcJSVusage<- function(db,nClonesThreshold){

vusage<-db %>%
  group_by(sampleId) %>%
  mutate(totalNClones=n()) %>%
  group_by(sampleId,V,repnum,timepoint,donor,subpop) %>%
  filter(totalNClones>=nClonesThreshold) %>%
  arrange(cloneRank) %>%
  slice(1:nClonesThreshold) %>%
  summarise(NClones=n(), NClones_normalized=n()/first(totalNClones))


  ids<-vusage %>% pull(sampleId) %>% unique()
  all_pairs<-combn(ids,2,c,F)

  js_vusage<-combn(ids,2,c)%>% t()  %>% set_colnames(c('id.x','id.y'))%>% as_tibble()

  js_vusage$js_dist <- sapply(all_pairs , function(pair, pseudocount= T) {

    df <- merge(vusage %>% filter(sampleId==pair[[1]]) ,
```

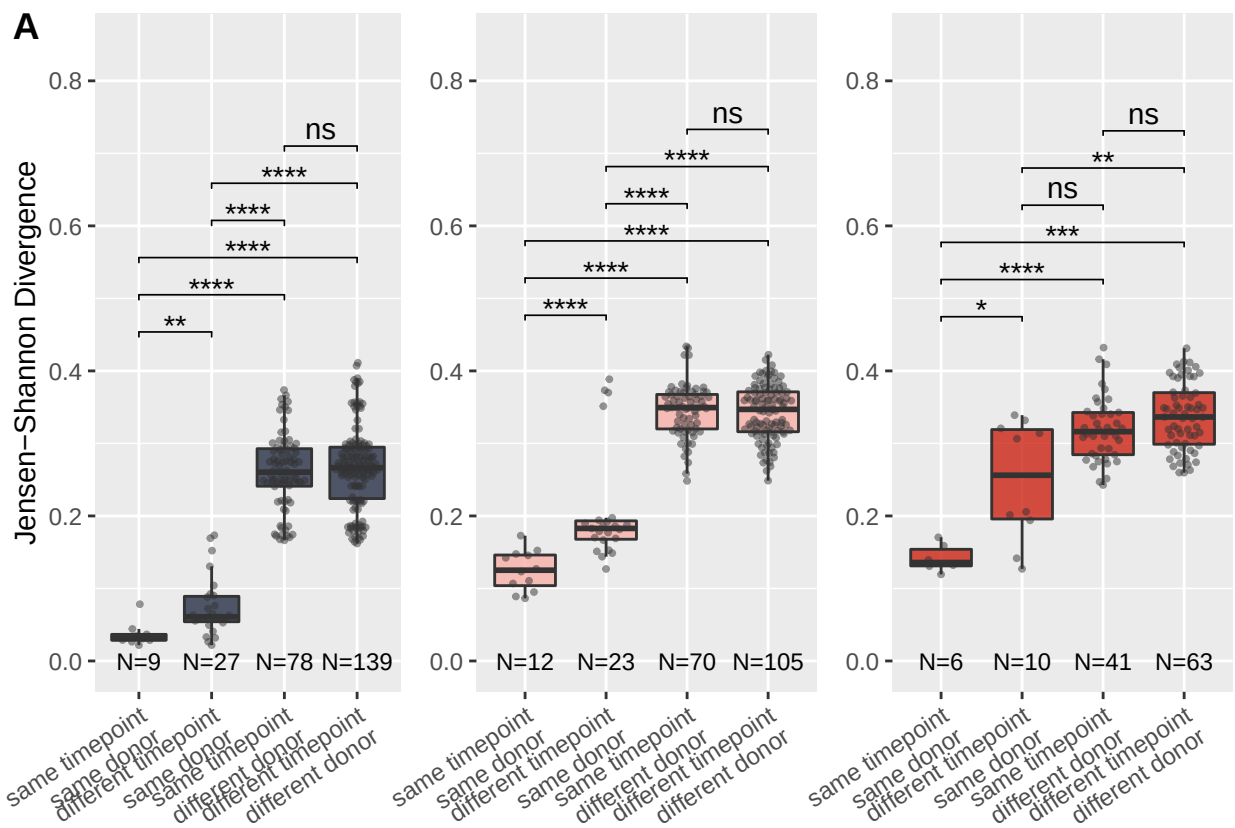
```
                    vusage %>% filter(sampleId==pair[[2]]),
                    by = "V",all = pseudocount )%>%
          replace_na(list(freq.x=0,freq.y=0,
                          NClones_normalized.x=0,NClones_normalized.y=0,
                          nGroups_normalized.x=0,nGroups_normalized.y=0))

        if (pseudocount) {

          pseudocount.x = df %>% filter(NClones_normalized.x >0) %>%
            pull(NClones_normalized.x) %>% min()
          pseudocount.x = pseudocount.x/10

          pseudocount.y = df %>% filter(NClones_normalized.y >0) %>%
            pull(NClones_normalized.y) %>% min()
          pseudocount.y = pseudocount.y/10

          df%<>%mutate(NClones_normalized.x=ifelse(NClones_normalized.x==0,
                                                   yes=pseudocount.x,
                                                   no=NClones_normalized.x))%>%
            mutate(NClones_normalized.y=ifelse(NClones_normalized.y==0,
                                               yes=pseudocount.y,
                                               no=NClones_normalized.y))
        }

        df %<>% filter(NClones_normalized.x!=0 & NClones_normalized.y!=0) %>%
          mutate(p.x=NClones_normalized.x/sum(NClones_normalized.x),
                 p.y=NClones_normalized.y/sum(NClones_normalized.y))

        m <- 0.5 * (df$p.x + df$p.y)
        JS <- 0.5 * (sum(df$p.x * log2(df$p.x / m)) + sum(df$p.y * log2(df$p.y / m)))
        return(sqrt(JS))
      })
  js_vusage %<>% separate(
    id.x,
    into = c("donor.x", "timepoint.x", "subpop.x", "repnum.x"),
    remove = F
  ) %>%
    separate(
      id.y,
      into = c("donor.y", "timepoint.y", "subpop.y", "repnum.y"),
      remove = F
    )
  return(js_vusage)
}

# Calculate JS for top 14K Bmem, 600 PBL, 300 PL
js_vusage<-
  calcJSVusage(subset_db %>% filter(subpop=="Bmem"),14000) %>%
  bind_rows(calcJSVusage(subset_db %>% filter(subpop=="PBL"),600)) %>%
  bind_rows(calcJSVusage(subset_db %>% filter(subpop=="PL"),300)) %>%
  mutate(type=case_when(
   timepoint.x==timepoint.y & donor.x==donor.y ~ "same timepoint / same donor",
   timepoint.x==timepoint.y & donor.x!=donor.y ~ "same timepoint / different donor",
```

```
   timepoint.x!=timepoint.y & donor.x==donor.y ~ "different timepoint / same donor",
   timepoint.x!=timepoint.y & donor.x!=donor.y ~ "different timepoint / different donor"),
    type=factor(type,levels=c("same timepoint / same donor",
                              "different timepoint / same donor",
                              "same timepoint / different donor",
                              "different timepoint / different donor")))
```

Fig 2A - Jensen-Shannon divergence of IGH repertoires

```
mycomparisons = list(combn(c("same timepoint / same donor",
                             "different timepoint / same donor",
                             "same timepoint / different donor",
                             "different timepoint / different donor"),2,c,F))
mycomparisons %<>% unlist(recursive = F)

stat.test <- js_vusage %>%
  filter(subpop.x==subpop.y) %>%
  group_by(subpop.x) %>%
  wilcox_test(js_dist~type,comparisons = mycomparisons)%>% add_xy_position(x = "type")

#function to calculate N of observations for plots

give.n <- function(y){
    return(data.frame(
             y = 0,
             label = paste0('N=', length(y))))
}


plot_js<-function(subp){
  js_vusage %>%
  filter(subpop.x==subpop.y,subpop.x==subp) %>%
  ggplot(aes(y=js_dist,x=type,fill=subpop.x))+
    geom_boxplot(fill=case_when(subp=="Bmem" ~"#273046",
                                subp=="PBL" ~ "#F8AFA8",
                                subp=="PL" ~ "#CB2314"), alpha = 0.8, outlier.colour = NA)+
    ylim(0,0.85)+
  labs(y="Jensen-Shannon Divergence")+
  scale_x_discrete(
    labels = c(
      "same timepoint / same donor" = "same timepoint\nsame donor",
      "same timepoint / different donor" = "same timepoint\ndifferent donor",
      "different timepoint / same donor" = "different timepoint\nsame donor",
      "different timepoint / different donor" = "different timepoint\ndifferent donor"
    )) +
    theme(legend.position = "none",
        axis.title.x = element_blank(),
        plot.title = element_text(size=10,hjust = 0.5),
        axis.text.x=element_text(angle = 33, hjust = 1)
       )+
```

```
  stat_summary(fun.data = give.n, geom = "text",size=3)+
    stat_pvalue_manual(stat.test %>% filter(subpop.x == subp),
                       label = "p.adj.signif",
                       tip.length = 0.01) +
    geom_quasirandom(
      width = 0.3,
      size = 0.7,
      color = "grey20",
      alpha = 0.5
    )
}


Fig2A_Bmem<- plot_js("Bmem")
Fig2A_PBL<- plot_js("PBL") + theme(axis.title.y = element_blank())
Fig2A_PL<-plot_js("PL") + theme(axis.title.y = element_blank())

cowplot::plot_grid(
  Fig2A_Bmem,
  Fig2A_PBL,
  Fig2A_PL,
  ncol = 3,
  rel_widths = c(1, 1, 1),
  labels = c("A", "", ""),
  label_size = 14
)
```



Number of shared clonotypes function definition (by aaCDR3+V+isotype)

```
overlap<-function(db,nClonesThreshold,selectedSubpop){
db<-db %>% filter(totalNClones>=nClonesThreshold,subpop%in%selectedSubpop)

ids<-db %>% pull(sampleId) %>% unique()
all_pairs<-combn(ids,2,c,F)

overlapTable<-combn(ids,2,c)%>% t()  %>% set_colnames(c('id.x','id.y'))%>% as_tibble()

overlaps<-
  lapply(all_pairs , function(pair) {

  df1=db %>% filter(sampleId==pair[[1]]) %>% arrange(cloneRank) %>%
    {if(nClonesThreshold!=0) slice(.,1:nClonesThreshold) else . } %>%
    group_by(donor,timepoint,subpop,repnum,aaSeqCDR3,V,isotype) %>% summarise(N=n())
  df2=db %>% filter(sampleId==pair[[2]]) %>% arrange(cloneRank) %>%
    {if(nClonesThreshold!=0) slice(.,1:nClonesThreshold) else . } %>%
    group_by(donor,timepoint,subpop,repnum,aaSeqCDR3,V,isotype) %>% summarise(N=n())


  Noverlap<-merge(df1, df2, by = c("aaSeqCDR3","V","isotype"))

  nCommonClones<-Noverlap %>% nrow()
  NoverlapNorm<-nCommonClones/(as.double(nrow(df1))*as.double(nrow(df2)))

  return(tibble(nCommonClones=nCommonClones, normNCommonClones=NoverlapNorm))

})%>% bind_rows()

overlapTable<-cbind(overlapTable,overlaps) %>%
  separate(id.x, into=c("donor.x","timepoint.x","subpop.x","repnum.x"),remove=F ) %>%
  separate(id.y, into=c("donor.y","timepoint.y","subpop.y","repnum.y"),remove=F )
return(overlapTable)
}
```

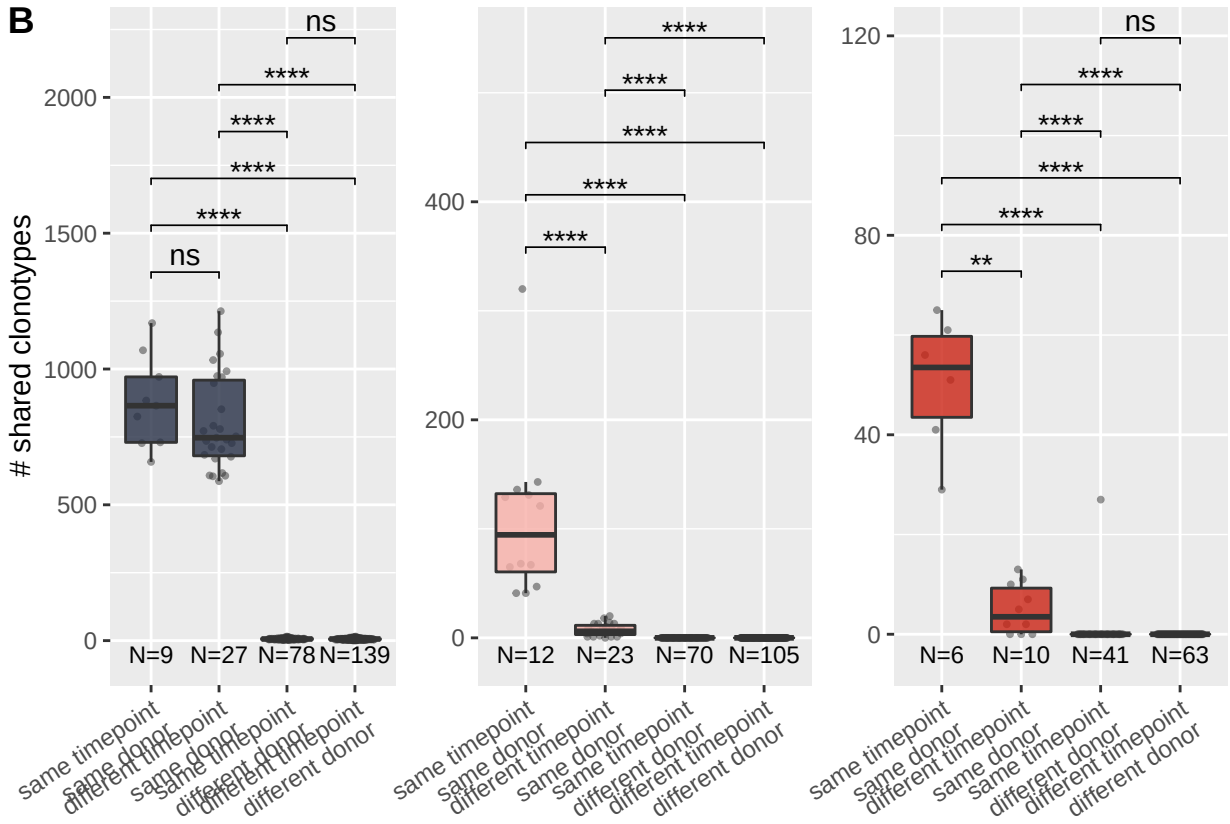
Calculate overlap for top 14000 Bmem, 600 PBL, 300 PL clonotypes (overlap by aaCDR3 +isotype)

```
overlapTable_Bmem<-overlap(subset_db,14000,"Bmem") %>% mutate(subpop="Bmem")
overlapTable_PBL<-overlap(subset_db,600,"PBL") %>% mutate(subpop="PBL")
overlapTable_PL<-overlap(subset_db,300,"PL") %>% mutate(subpop="PL")

overlapTable<-
  overlapTable_Bmem %>%
  bind_rows(overlapTable_PBL) %>%
  bind_rows(overlapTable_PL)
```

Fig 2B - Number of shared clones for fixed # of top clonotypes in each subset

```
mycomparisons = list(combn(c("same timepoint / same donor",
                             "different timepoint / same donor",
                             "same timepoint / different donor",
                             "different timepoint / different donor"),2,c,F))
mycomparisons %<>% unlist(recursive = F)

overlapTable %<>%
mutate(type=case_when(
 timepoint.x==timepoint.y & donor.x==donor.y ~ "same timepoint / same donor",
 timepoint.x==timepoint.y & donor.x!=donor.y ~ "same timepoint / different donor",
 timepoint.x!=timepoint.y & donor.x==donor.y ~ "different timepoint / same donor",
 timepoint.x!=timepoint.y & donor.x!=donor.y ~ "different timepoint / different donor"),
 type=factor(type,levels=c("same timepoint / same donor",
                           "different timepoint / same donor",
                           "same timepoint / different donor",
                           "different timepoint / different donor")))


plot_overlaps<- function(subp){


stat.test <- overlapTable %>%
  filter(subpop.x==subp) %>%
  wilcox_test(nCommonClones~type,
              comparisons = if(subp=="PBL") mycomparisons[1:5] else mycomparisons) %>%
  add_xy_position(x = "type")

#function to calculate N of observations for plots
#(that's why I am not using facet_wrap here)

give.n <- function(y){
    return(data.frame(
             y = case_when(subp=="Bmem"~ -50,
                           subp=="PBL" ~ -15,
                           subp=="PL" ~ -4),
             label = paste0('N=', length(y))))
}

overlapTable %>%
  filter(subpop.x==subp) %>%
  ggplot(aes(x=type,y=nCommonClones))+
    labs(y="# shared clonotypes")+
    scale_x_discrete(
      labels = c(
        "same timepoint / same donor" = "same timepoint\nsame donor",
        "same timepoint / different donor" = "same timepoint\ndifferent donor",
        "different timepoint / same donor" = "different timepoint\nsame donor",
        "different timepoint / different donor" = "different timepoint\ndifferent donor"
      ))+
  stat_summary(fun.data = give.n, geom = "text",size=3)+
  stat_pvalue_manual(stat.test, label = "p.adj.signif", tip.length = 0.01)+
```

```
  geom_quasirandom(width=0.3, size = 0.7, color = "grey20", alpha = 0.5)+
  geom_boxplot(fill=case_when(subp=="Bmem" ~"#273046",
                              subp=="PBL" ~ "#F8AFA8",
                              subp=="PL" ~ "#CB2314"),
               alpha = 0.8,
               outlier.colour = NA)+
  theme(axis.title.x = element_blank(),
          plot.title = element_text(size=10,hjust = 0.5),
        axis.text.x=element_text(angle = 33, hjust = 1))
}

Fig2B_Bmem<-plot_overlaps("Bmem")
Fig2B_PBL<-plot_overlaps("PBL") +theme(axis.title.y = element_blank())
Fig2B_PL<-plot_overlaps("PL") +theme(axis.title.y = element_blank())

cowplot::plot_grid(
  Fig2B_Bmem,
  Fig2B_PBL,
  Fig2B_PL,
  ncol = 3,
  rel_widths = c(1, 1, 1),
  labels = c("B", "", ""),
  label_size = 14
)
```



Calculate overlaps of 5K clones - Bmem, Naive, Olga between donors

```
overlapTable_Bmem_5K<-overlap(subset_db,5000,"Bmem")
overlapTable_Bmem_random5K<-overlap(subset_db %>%
                                    filter(subpop=="Bmem") %>%
                                    sample_n(5000),5000,"Bmem")
overlapTable_naive<-overlap(naive_db,5000,"naive")
overlapTable_olga<-overlap(olga_db_5k,5000,"synth")
```

Fig 2C Overlaps of 5K clones, for each pair of our donors overlaps are averaged

```
crossDonor<-bind_rows(overlapTable_Bmem_5K %>% filter(donor.x!=donor.y) %>%
                        rowwise() %>%
                        mutate(donors=paste(sort(c(donor.x,donor.y)),
                                            collapse="_")) %>%
                        ungroup() %>%
                        group_by(donors,subpop.x) %>%
                        summarise(nCommonClones=mean(nCommonClones),
                                  normMCommonClones=mean(nCommonClones)) ,
                      overlapTable_Bmem_random5K %>%
                        filter(donor.x!=donor.y) %>%
                        rowwise() %>%
                        mutate(donors=paste(sort(c(donor.x,donor.y)),
                                            collapse="_")) %>%
                        ungroup() %>%
                        group_by(donors) %>%
                        summarise(nCommonClones=mean(nCommonClones),
                                  normMCommonClones=mean(nCommonClones)) %>%
                        mutate(subpop.x="Bmem_random"),
                      overlapTable_naive %>% mutate(subpop.x="naive"),
                      overlapTable_olga %>% mutate(subpop.x="synth")) %>%
  ungroup()


crosscomparisons<-list(combn(c("Bmem","Bmem_random","naive","synth"),2,c,F))
crosscomparisons %<>% unlist(recursive = F)

stat.test <- crossDonor %>%
  wilcox_test(nCommonClones~subpop.x,comparisons = crosscomparisons) %>%
  add_xy_position(x = "subpop.x")


give.n <- function(y){
    return(data.frame(
             y = -1,
             label = paste0('N=', length(y))))
}


Fig2C<-crossDonor %>%
```

```
  ggplot(aes(y=nCommonClones,x=subpop.x))+

  scale_x_discrete(labels=c("Bmem" = "Bmem\ntop", "Bmem_random" = "Bmem\nrandom",
                            "naive"="Naive\ntop", "synth"="In silico"))+
  labs(y="# shared clonotypes")+
   scale_fill_manual(values = c("Bmem" = "#273046", "Bmem_random" = "#273046",
                            "naive"="seagreen", "synth"="grey40")) +
    theme(axis.title.x = element_blank(),
          axis.text.x=element_text(size = 11))+
  stat_summary(fun.data = give.n, geom = "text",size=3)+
  stat_pvalue_manual(stat.test, label = "p.adj.signif", tip.length = 0.01)+
    geom_quasirandom(width=0.4, size = 0.3, color = "grey20", alpha = 0.5)+
geom_boxplot(fill="#273046",alpha = 0.8, outlier.colour = NA)

Fig2C
```

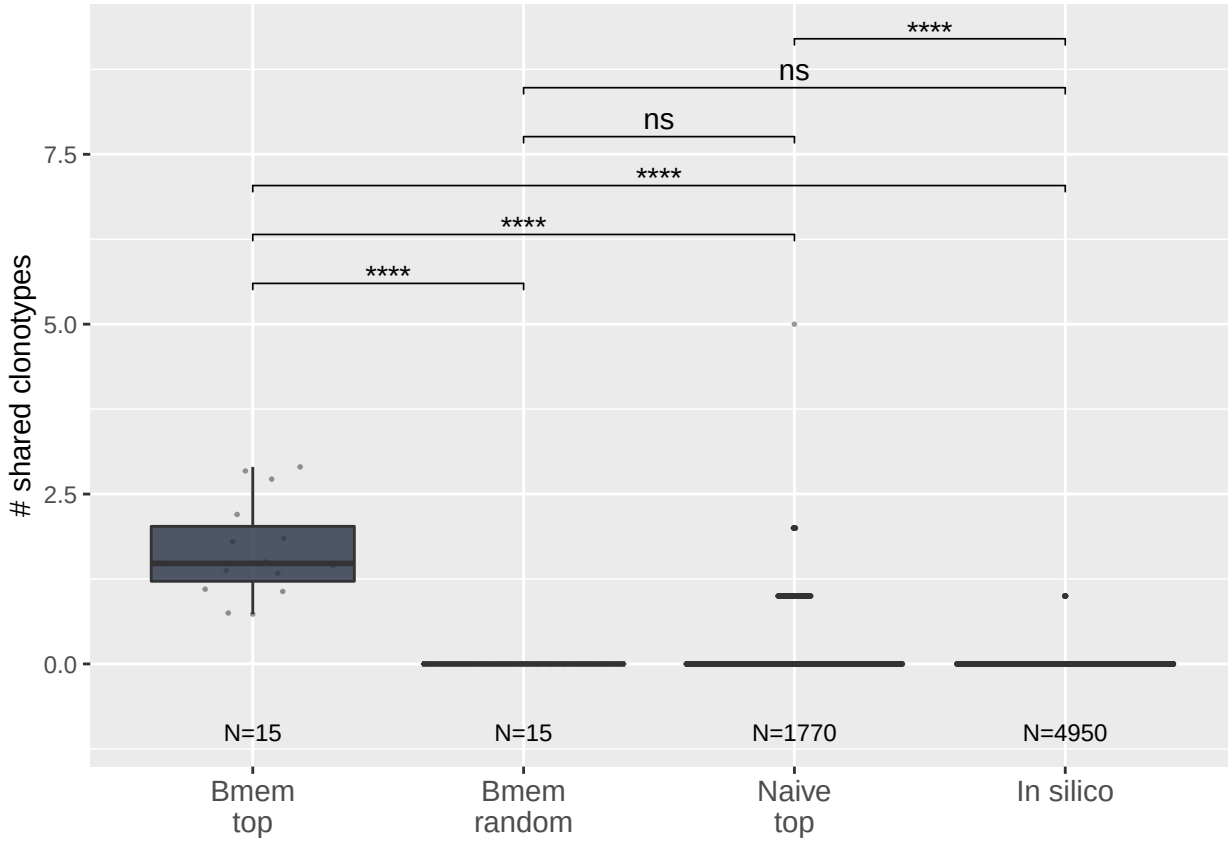
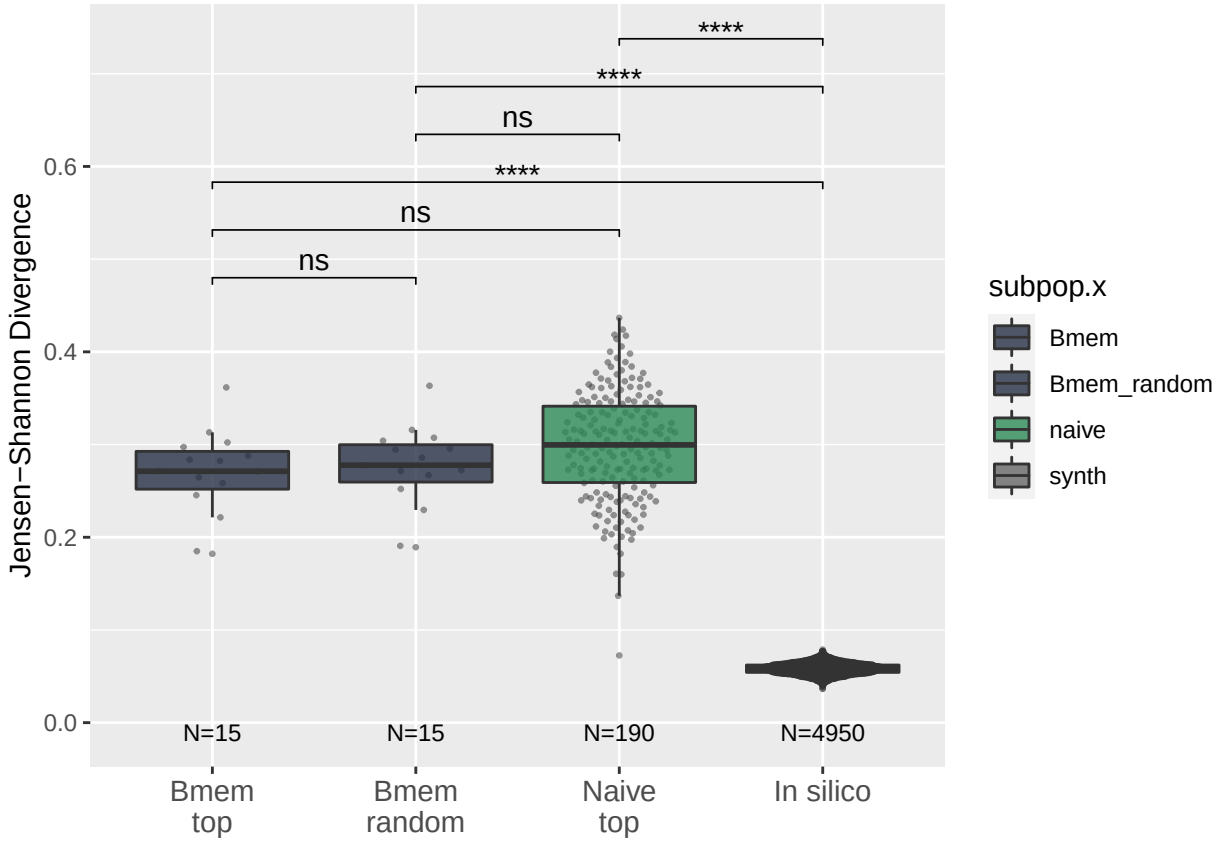


Fig 2D Vusage crossdonor

```
vusageTable_Bmem_5K<-subset_db %>%
  ungroup() %>%
  filter(subpop=="Bmem") %>%
  calcJSVusage(5000)
```

```r
vusageTable_Bmem_random5K<-subset_db %>%
  filter(subpop=="Bmem",totalNClones>5000) %>%
  group_by(sampleId) %>%
  sample_n(5000) %>%
  calcJSVusage(5000)



vusageTable_naive<-naive_db %>%
  calcJSVusage(5000)

vusageTable_olga<-olga_db_5k %>%
  calcJSVusage(5000)


crossDonor_vusage<-
  bind_rows(
    vusageTable_Bmem_5K %>% filter(donor.x != donor.y) %>%
      rowwise() %>%
      mutate(donors = paste(sort(c(donor.x, donor.y)),
      collapse = "_")) %>%
      ungroup() %>%
      group_by(donors, donor.x, donor.y, subpop.x) %>%
      summarise(js_dist = mean(js_dist)),
    vusageTable_Bmem_random5K %>% filter(donor.x != donor.y) %>%
      rowwise() %>%
      mutate(donors = paste(sort(c(donor.x, donor.y)),
      collapse = "_")) %>%
      ungroup() %>%
      group_by(donors, donor.x, donor.y, subpop.x) %>%
      summarise(js_dist = mean(js_dist)) %>%
      mutate(subpop.x = "Bmem_random"),
    vusageTable_naive %>% mutate(subpop.x = "naive"),
    vusageTable_olga %>% mutate(subpop.x = "synth")
  ) %>%
  ungroup()

crosscomparisons<-list(combn(c("Bmem","Bmem_random","naive","synth"),2,c,F))
crosscomparisons %<>% unlist(recursive = F)

stat.test <- crossDonor_vusage %>%
  wilcox_test(js_dist~subpop.x,comparisons = crosscomparisons) %>%
  add_xy_position(x = "subpop.x")


give.n <- function(y){
    return(data.frame(
             y = -0.01,
             label = paste0('N=', length(y))))
}


Fig2D<-crossDonor_vusage %>%  filter(subpop.x!="PBL",subpop.x!="PL") %>%
```

```
  ggplot(aes(y=js_dist,x=subpop.x))+
  scale_fill_manual(values = c("Bmem" = "#273046", "Bmem_random" = "#273046",
                               "naive"="seagreen", "synth"="grey40")) +
     scale_x_discrete(labels=c("Bmem" = "Bmem\ntop",
                                 "Bmem_random" = "Bmem\nrandom",
                               "naive"="Naive\ntop",
                               "synth"="In silico"))+
  stat_summary(fun.data = give.n, geom = "text",size=3)+
  stat_pvalue_manual(stat.test, label = "p.adj.signif", tip.length = 0.01)+
  labs(y="Jensen-Shannon Divergence")+
  theme(axis.title.x = element_blank(),
        axis.text.x=element_text(size = 11))+
   geom_quasirandom(width=0.3, size = 0.5, color = "grey20", alpha = 0.5)+
  geom_boxplot(aes(fill=subpop.x),alpha = 0.8, outlier.colour = NA)
Fig2D
```



Function to select only public clonotypes

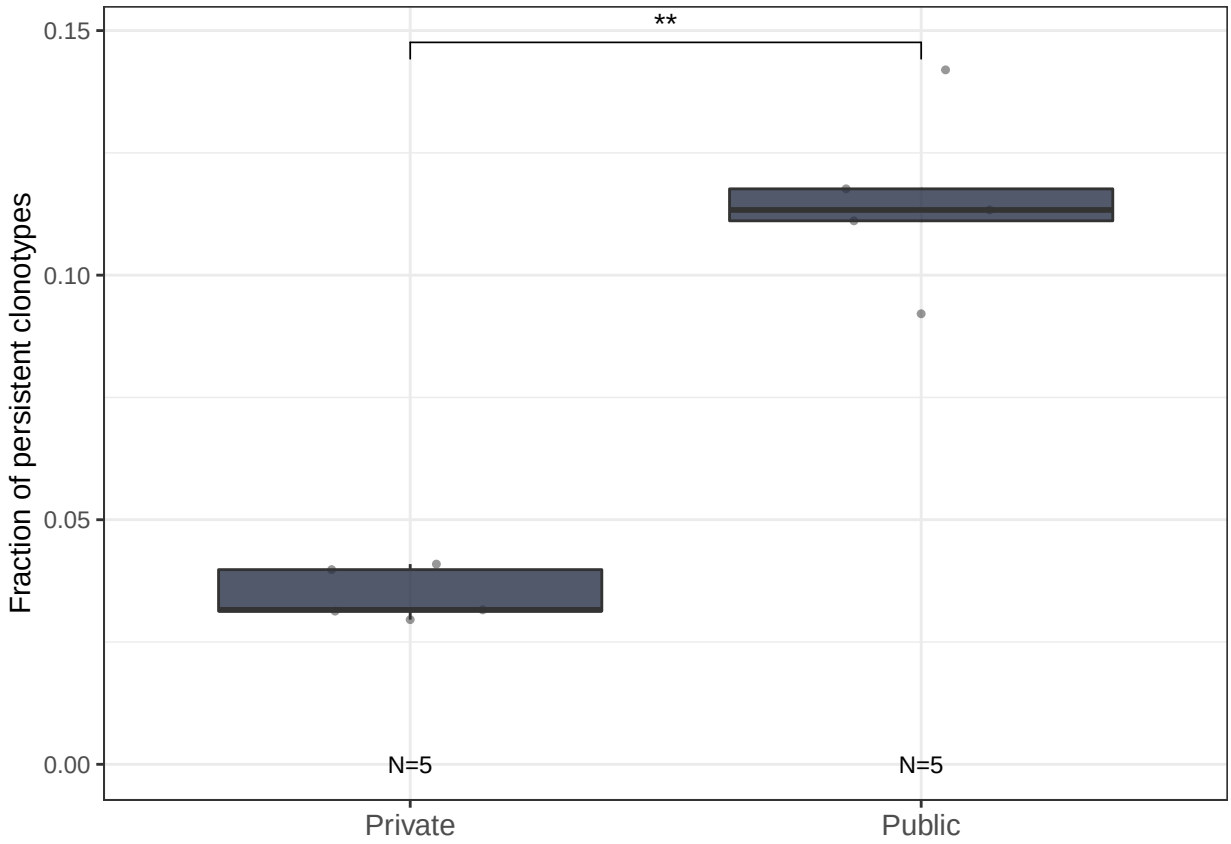
```
overlapClones<-function(db,nClonesThreshold,selectedSubpop){
db<-db %>% filter(totalNClones>=nClonesThreshold,subpop%in%selectedSubpop)

ids<-db %>% pull(sampleId) %>% unique()
all_pairs<-combn(ids,2,c,F)
```

```
overlaps<-
  lapply(all_pairs , function(pair) {

  df1=db %>% filter(sampleId==pair[[1]]) %>%
    arrange(cloneRank) %>% slice(1:nClonesThreshold)  %>%
    group_by(sampleId,donor,timepoint,subpop,repnum,aaSeqCDR3,V,isotype) %>%
    summarise(N=n(),seqId=c(sequenceId))  #,cloneCount=sum(cloneCount)
  df2=db %>% filter(sampleId==pair[[2]]) %>%
    arrange(cloneRank) %>% slice(1:nClonesThreshold) %>%
    group_by(sampleId,donor,timepoint,subpop,repnum,aaSeqCDR3,V,isotype) %>%
    summarise(N=n(),seqId=c(sequenceId)) #,cloneCount=sum(cloneCount)


  Noverlap<-merge(df1, df2, by = c("aaSeqCDR3","V","isotype"))


  return(Noverlap)

})%>% bind_rows()


return(overlaps)
}

# selecting public clonotypes in top 5000 Bmem
overlapped_clones<-overlapClones(subset_db,5000,"Bmem")

overlapped_clones<-overlapped_clones %>%
  filter(donor.x!=donor.y) %>%
  select(seqId.x) %>%
  rename(sequenceId=seqId.x) %>%
  rbind( overlapped_clones %>%
           filter(donor.x!=donor.y) %>%
           select(seqId.y) %>%
           rename(sequenceId=seqId.y)
) %>% unique() %>% pull(sequenceId)
```

Fig 2E - Public vs Persistent

```
freq<-function(db){
  db %>% mutate(freq=n/sum(n))
}


give.n <- function(y){
  return(data.frame(
    y = 0,
    label = paste0('N=', length(y))))
}
```

```r
overlapped_reproducibility<-subset_db %>%
  mutate(overlapped=ifelse(sequenceId %in% overlapped_clones,TRUE,FALSE)) %>%
  group_by(overlapped,donor,aaSeqCDR3,V,isotype) %>%
  summarise(nTimepoints=n_distinct(timepoint),
            timepoints=paste(unique(timepoint),collapse=","),
            nSubpops=n_distinct(subpop),
            subpops=paste(unique(subpop),collapse=",")) %>%
  ungroup()




Fig2E <-overlapped_reproducibility %>%
  ungroup() %>%
  dplyr::count(donor,nTimepoints,overlapped) %>%
  group_by(donor,overlapped) %>%
  freq() %>%
  filter(donor!="MT") %>%
  mutate(reproduced=ifelse(nTimepoints>1, "reproduced","not"),
         overlapped=ifelse(overlapped,"Public","Private")) %>%
  filter(reproduced=="reproduced") %>%
  ungroup() %>%
  group_by(donor,overlapped) %>%
  summarise(n=sum(n),freq=mean(freq)) %>%
  ggplot(aes(x=overlapped,y=freq))+
  theme_bw()+
  stat_compare_means(comparisons = list(c("Public","Private")),
                     label = "p.signif") +
  labs( y="Fraction of persistent clonotypes") +
  theme(axis.title.x = element_blank(),legend.position = "bottom",
        axis.text.x=element_text(size = 11)) +
  geom_quasirandom(width=0.3, size = 0.9, color = "grey20", alpha = 0.5)+
  geom_boxplot(fill="#273046",alpha = 0.8, outlier.colour = NA)+
  stat_summary(fun.data = give.n, geom = "text",size=3)

Fig2E
```


**
0.15
0.10
0.05
0.00
Fraction of persistent clonotypes
N=5
N=5
Private
Public