```
# =====================================
#
# This script analyzes TCRB sequencing data as returned by
Adaptive Technologies and made available online
#
# - Each tab-delimited file is a single sample: blood, CPL or
synovium of a patient. File naming convention is
Patient_X_Blood|CPL|Synovium_TCRB.tsv
# - Each row is a nucleotide sequence (column "nucleotide"),
with its in-silico translation (column "aminoAcid"), in-frame
information (column "sequenceStatus"), T cell genome count
(column "estimatedNumberGenomes") and V Family Name (column
"vFamilyName")
#
# =====================================

# === Analysis options - BEGIN
seq.type <- "aminoAcid"  # set whether analysis is on
"aminoAcid" or "nucleotide" sequences
sampling.replace <- F  # set whether sampling is with (T) or
without (F) replacement
n.subs <- 200  # set number of subsamples
n.intervals <- 200  # set number of sample size intervals for
analysis of TCR repertoire overlap
# === Analysis options - END

# === Load required packages - BEGIN
library(plyr)
library(dplyr)
library(vegan)
library(ggplot2)
library(Biobase)
library(foreach)
library(doMC)
library(iterators)
# === Load required packages - END

registerDoMC(cores=detectCores())
mcoptions <- list(preschedule=T, set.seed=F)

samples <- c("Blood", "CPL", "Synovium")
chart.colors <- c("blue", "orange", "black")

# === Load data - BEGIN
```

```
dir.create(file.path(".", "results"), showWarnings=F)
index <- list.files(path=".", pattern="tsv")  # assumes this
script runs from a folder containing all data files
cat("Analysis in progress...\n")
all.data <- llply(index, read.delim)
names(all.data) <- gsub("_TCRB.tsv", "", index)
all.data <- lapply(all.data, tbl_df)
all.data <- lapply(all.data, function(x) dplyr::filter(x,
sequenceStatus == "In"))  # retain productive TCR
rearrangements
all.data <- lapply(all.data, function(x) dplyr::select(x,
nucleotide, aminoAcid, estimatedNumberGenomes, vFamilyName))
# === Load data - END

# === Vb family analysis - BEGIN
v.counts <- lapply(all.data, function(x)
aggregate(estimatedNumberGenomes ~ vFamilyName, FUN=sum,
data=x))
v.counts <- lapply(v.counts, function(x) x[x$vFamilyName != ""
& x$vFamilyName != "TCRBVA", ])
for(i in 1:length(v.counts)) v.counts[[i]] <-
cbind(v.counts[[i]], dataset=names(v.counts[i]))
v.tot <- sapply(v.counts, function(x)
sum(x$estimatedNumberGenomes))
for(i in 1:length(v.counts))
v.counts[[i]]$estimatedNumberGenomes <-
v.counts[[i]]$estimatedNumberGenomes / v.tot[i]
all.v <- data.frame()
for(i in 1:length(v.counts)) all.v <- rbind(all.v,
v.counts[[i]])
all.v <- droplevels(all.v)
all.v$vFamilyName <- factor(all.v$vFamilyName,
levels=levels(all.v$vFamilyName)
[order(levels(all.v$vFamilyName))])
levels(all.v$vFamilyName) <- sub("TCRB", "",
levels(all.v$vFamilyName))
all.v$Group <- substr(all.v$dataset, 1, 9)
all.v <- all.v[!is.na(all.v$Group), ]
all.v$Group <- factor(all.v$Group)
all.v$dataset <- as.character(all.v$dataset); all.v$dataset <-
substr(all.v$dataset, 11, nchar(all.v$dataset))
all.v$dataset <- factor(all.v$dataset)

# plot results
v.chart <- qplot(data=all.v, geom="bar", position="stack",
x=dataset, y=estimatedNumberGenomes, fill=vFamilyName,
stat="identity") + facet_wrap(~ Group, drop=T, scales="free_x",
ncol=3) + theme_bw() + scale_fill_hue(l=58, c=90) +
theme(legend.position="right", legend.title=element_blank(),
axis.title.x=element_blank(), axis.title.y=element_blank(),
axis.ticks.y=element_blank())
ggsave(file.path(".", "results", "V_Family.pdf"), v.chart,
width=10, height=8)
rm(list=c("index", "all.v", "v.counts", "v.tot", "v.chart",
"i"))
# === Vb family analysis - END

# === Reshape data - BEGIN
subset.data <- lapply(all.data, function(x) x[, c(seq.type,
"estimatedNumberGenomes")])
for(z in 1:length(subset.data)) colnames(subset.data[[z]]) <-
c("seq", "n")
subset.data <- lapply(subset.data, function(x) x %>%
dplyr::filter(seq != "") %>% group_by(seq) %>%
dplyr::summarise(n=sum(n)))
for(z in 1:length(subset.data)) subset.data[[z]]$seq <-
as.character(subset.data[[z]]$seq)
patients <- unique(substr(names(subset.data), 1, 9))
patient.data <- vector("list", length(patients))
names(patient.data) <- patients; rm(patients)
for(i in 1:length(patient.data)) {patient.data[[i]] <-
vector("list", 3); names(patient.data[[i]]) <-
paste(names(patient.data[i]), samples, sep="_")}
for(i in 1:length(patient.data))
    for(z in 1:length(samples))
        patient.data[[i]][[z]] <-
subset.data[[which(grepl(names(patient.data[[i]][z]),
names(subset.data)))]]
rm(list=c("all.data", "subset.data", "seq.type", "i", "z"))
# === Reshape data - END

foreach(p=iter(patient.data, .inorder=F)) %dopar% {
    sink(file.path(".", "results", paste(substr(names(p[1]), 1,
9), ".txt", sep="")))

    # === Chao distance - BEGIN
    chao.data <- Map(function(x, i) setNames(x, ifelse(names(x)
%in% c("seq"), names(x), sprintf('%s.%d', names(x), i))), p,
seq_along(p))
```

```
    chao.data <- Reduce(function(x, y) merge(x, y, sort=F,
all=T, by="seq"), chao.data, accumulate=F)
    chao.data[is.na(chao.data)] <- 0
    chao.data <- t(chao.data[, -1])
    rownames(chao.data) <- samples

    # calculate the index
    chao.dist <- vegdist(chao.data, method="chao")

    # plot results
    pdf(file.path(".", "results", paste(substr(names(p[1]), 1,
9), "ChaoDistance.pdf", sep="_")), height=8, width=5,
onefile=T)
    par(lwd=4); par(cex.axis=2.5)
    plot(hclust(chao.dist, method="single"), main="", xlab="",
sub="")
    plot(as.dendrogram(hclust(chao.dist, method="single")),
main="", ylim=c(0.8, 1))
    dev.off()

    cat("\nChao Distance:\n"); print(chao.dist)  # outputs the
Chao distances to allow calculating summary statistics for all
patients

    rm(list=c("chao.data", "chao.dist"))
    # === Chao distance - END

    # convert data to one-row-per-sequence format
    samples.data <- lapply(p, function(y)
unlist(lapply(1:nrow(y), FUN=function(x) rep(y$seq[x],
times=y$n[x]))))
    names(samples.data) <- samples

    # === Renyi index - BEGIN

    # calculate the index
    samples.renyi <- lapply(lapply(samples.data, function(y)
lapply(1:n.subs, function(x) table(sample(y,
min(sapply(samples.data, length)),
replace=sampling.replace)))), function(w)
rowMedians(simplify2array(lapply(1:n.subs, function(s)
renyi(w[[s]], scales=seq(0, 2, length.out=11))))))

    # plot results
    pdf(file.path(".", "results", paste(substr(names(p[1]), 1,
9), "RenyiIndex.pdf", sep="_")), height=8, width=8)
    plot(c(1:11), type="n", xaxt="n", xlab="alpha", ylab="Renyi
diversity index", ylim=c(min(sapply(samples.renyi, min)),
max(sapply(samples.renyi, max))), cex.axis=2, cex.lab=1.5)
    axis(1, at=1:11, labels=seq(0, 2, length.out=11),
cex.axis=2)
    for(t in 1:length(samples.renyi)) lines(samples.renyi[[t]],
col=chart.colors[t], lwd=10)
    abline(v=6, lty="dashed", col="darkgrey", lwd=6)
    for(t in 1:length(samples.renyi)) points(6,
samples.renyi[[t]][6], cex=4, pch=19, col=chart.colors[t])
    legend("bottomleft", legend=samples, col=chart.colors,
lty=1, lwd=3)
    dev.off()

    cat("\nRenyi Index at alpha=1:\n"); for(t in
1:length(samples.renyi)) cat(names(samples.renyi[t]),
samples.renyi[[t]][6], "\n")  # outputs the Renyi indexes to
allow calculating summary statistics for all patients

    rm(list=c("samples.renyi", "t"))
    # === Renyi index - END

    # === Pairwise overlap - BEGIN
    pos.x1 <- grep("Blood", samples)
    pos.x2 <- grep("CPL", samples)
    pos.ref <- grep("Synovium", samples)
    min.x1 <- min(length(samples.data[[pos.x1]]),
length(samples.data[[pos.ref]]))
    min.x2 <- min(length(samples.data[[pos.x2]]),
length(samples.data[[pos.ref]]))
    min.xs <- min(min.x1, min.x2)
    acc.x1 <- acc.x2 <- vector("numeric", length=n.intervals)

    compute.overlap <- function(pair, sample.size) {
        pair.table <- lapply(pair, function(y) lapply(1:n.subs,
function(x) data.frame(table(sample(y, sample.size,
replace=sampling.replace)))))
        return(median(simplify2array(lapply(1:n.subs,
function(y) {
            d.ref <- pair.table[[2]][[y]]
            common.set <- intersect(pair.table[[1]][[y]][, 1],
d.ref[, 1])
            return(sum(d.ref[d.ref[, 1] %in% common.set, 2]) /
sum(d.ref[, 2]))
        }))))
    }

    ovp.x1 <- compute.overlap(samples.data[c(pos.x1, pos.ref)],
min.xs) * 100
    ovp.x2 <- compute.overlap(samples.data[c(pos.x2, pos.ref)],
min.xs) * 100

    cat("\nOverlap with Synovium reference by
subsampling:\nBlood:", ovp.x1, "\nCPL:", ovp.x2)  # outputs the
overlap by subsampling to allow calculating summary statistics
for all patients

    ind.x1 <- round(seq.int(1, min.x1, length.out=n.intervals))
    for(t in 1:length(ind.x1))
        acc.x1[[t]] <- compute.overlap(samples.data[c(pos.x1,
pos.ref)], ind.x1[t])
    acc.x1 <- acc.x1 * 100

    ind.x2 <- round(seq.int(1, min.x2, length.out=n.intervals))
    for(t in 1:length(ind.x2))
        acc.x2[[t]] <- compute.overlap(samples.data[c(pos.x2,
pos.ref)], ind.x2[t])
    acc.x2 <- acc.x2 * 100

    save(ovp.x1, ovp.x2, min.xs, acc.x1, acc.x2, ind.x1,
ind.x2, file=file.path(".", "results",
paste(substr(names(p[1]), 1, 9), "OverlapBySubsampling.rda",
sep="_")))

    # plot results
    pdf(file.path(".", "results", paste(substr(names(p[1]), 1,
9), "OverlapBySubsampling.pdf", sep="_")), width=5, height=8)
    plot(ind.x2, acc.x2, xlim=c(0, max(ind.x1, ind.x2)),
ylim=c(0, max(acc.x1, acc.x2)) * 1.1, col=chart.colors[2],
xlab="# sampled T cells", ylab="Synovial Coverage [%]", cex=2,
cex.lab=1.5)
    points(ind.x1, acc.x1, col=chart.colors[1], cex=2)
    abline(v=min.xs, lty="dashed", col="darkgrey", lwd=4)
    points(min.xs, ovp.x1, pch=19, col=chart.colors[1], cex=4)
    points(min.xs, ovp.x2, pch=19, col=chart.colors[2], cex=4)
    legend("topright", legend=c("Blood ", "CPL "),
col=c(chart.colors[1], chart.colors[2]), lty=c(1, 1), lwd=4,
pch=c(NA, NA))
```

```
    dev.off()

    rm(list=c("pos.x1", "pos.x2", "pos.ref", "min.x1",
"min.x2", "min.xs", "ovp.x1", "ovp.x2", "t", "ind.x1",
"ind.x2", "acc.x1", "acc.x2", "compute.overlap"))
    # === Pairwise overlap - END

    sink()
    return(paste(substr(names(p[1]), 1, 9), "completed!", sep="
"))
}
```