#Pt14 pre-Tx MLR defined alloreactivity

This part of R script is published previously, and is available in the GitHub repository at <https://github.com/Aleksobrad/Fu-J-et-al.-LGVHR-manuscript>

#Pt14 post-Tx MLR defined alloreactivity

|  |  |
| --- | --- |
| Abbreviation | Full name |
| D4U | PreTx\_Donor\_CD4\_unstim |
| D4L | PreTx\_Donor\_CD4\_CFSElow |
| D8U | PreTx\_Donor\_CD8\_unstim |
| D8L | PreTx\_Donor\_CD8\_CFSElow |
| R4U | PreTx\_Recipient\_CD4\_unstim |
| R4L | PreTx\_Recipient\_CD4\_CFSElow |
| R8U | PreTx\_Recipient\_CD8\_unstim |
| R8L | PreTx\_Recipient\_CD8\_CFSElow |
| R'4U | PostTx\_Recipient\_CD4\_unstim |
| R'4L | PostTx\_Recipient\_CD4\_CFSElow |
| R'8U | PostTx\_Recipient\_CD8\_unstim |
| R'8L | PostTx\_Recipient\_CD8\_CFSElow |

data <- read.delim("P:/CCTI\_USERS/Jianing Fu/Adaptive samples and analysis/Pt14 iITx/Pt14 01082020.tsv")

# source('Z:/Jianing/10x scRNA-seq TCR-seq/Pt15\_MJ001, MJ004/alloreactivity\_printtofile 09132017.R')

names(data)

[1] "Rearrangement" "Sum..Templates."

[3] "Present.In" "Pt14\_Post\_PBMC\_day0030"

[5] "Pt14\_Post\_PBMC\_day0456" "Pt14\_Post\_ileum\_day0009"

[7] "Pt14\_Post\_ileum\_day0016" "Pt14\_Post\_ileum\_day0156"

[9] "Pt14\_Post\_ileum\_day0226" "Pt14\_Post\_ileum\_day0527"

[11] "Pt14\_Post\_ileum\_day0717" "Pt14\_Pre\_Donor\_CD4\_CFSElo"

[13] "Pt14\_Pre\_Donor\_CD4\_unstim" "Pt14\_Pre\_Donor\_CD8\_CFSElo"

[15] "Pt14\_Pre\_Donor\_CD8\_unstim" "Pt14\_Pre\_Recipient\_CD4\_CFSElo"

[17] "Pt14\_Pre\_Recipient\_CD4\_unstim" "Pt14\_Pre\_Recipient\_CD8\_CFSElo"

[19] "Pt14\_Pre\_Recipient\_CD8\_unstim" "Pt14\_iITx\_Bx\_POD527\_Native\_colon"

[21] "Pt14\_iITx\_Bx\_POD527\_Tx\_colon" "Pt14\_iITx\_Bx\_POD717\_Native\_colon"

[23] "Pt14\_iITx\_Bx\_POD717\_Tx\_colon" "Pt14\_iITx\_POD1764\_PBMC\_T"

[25] "Pt14\_iITx\_POD1764\_Tx\_colon\_Bx\_T" "Pt14\_iITx\_POD1764\_ileum\_Bx\_Chronic\_Rej"

[27] "Pt14\_iITx\_POD1764\_native\_colon\_Bx\_T" "Pt14\_iITx\_POD456\_PBMC\_R.4L\_D\_SP"

[29] "Pt14\_iITx\_POD456\_PBMC\_R.4U" "Pt14\_iITx\_POD456\_PBMC\_R.8L\_D\_SP"

[31] "Pt14\_iITx\_POD456\_PBMC\_R.8U"

# resolve CD4 and CD8 ambiguous clones first (ratio=5), then followed by resolve donor and recipient ambiguous clones.

#D4U vs D8U

normalize(data[,c(13,15)])

ratio=5

c1indices=which(data[,13]>0 & data[,15]>0 & data[,13]>ratio\*data[,15])

c2indices=which(data[,13]>0 & data[,15]>0 & data[,15]>ratio\*data[,13])

ambiindices=which(data[,13]>0 & data[,15]>0)

ambiindices=setdiff(setdiff(ambiindices,c1indices),c2indices)

data[c1indices,15]=0

data[c2indices,13]=0

data[ambiindices,c(13,15)]=0

#D4L vs D8L

normalize(data[,c(12,18)])

ratio=5

c1indices=which(data[,12]>0 & data[,18]>0 & data[,12]>ratio\*data[,18])

c2indices=which(data[,12]>0 & data[,18]>0 & data[,18]>ratio\*data[,12])

ambiindices=which(data[,12]>0 & data[,18]>0)

ambiindices=setdiff(setdiff(ambiindices,c1indices),c2indices)

data[c1indices,18]=0

data[c2indices,12]=0

data[ambiindices,c(12,18)]=0

#R4U vs R8U

normalize(data[,c(17,19)])

ratio=5

c1indices=which(data[,17]>0 & data[,19]>0 & data[,17]>ratio\*data[,19])

c2indices=which(data[,17]>0 & data[,19]>0 & data[,19]>ratio\*data[,17])

ambiindices=which(data[,17]>0 & data[,19]>0)

ambiindices=setdiff(setdiff(ambiindices,c1indices),c2indices)

data[c1indices,19]=0

data[c2indices,17]=0

data[ambiindices,c(17,19)]=0

#R4L vs R8L

normalize(data[,c(28,18)])

ratio=5

c1indices=which(data[,28]>0 & data[,18]>0 & data[,28]>ratio\*data[,18])

c2indices=which(data[,28]>0 & data[,18]>0 & data[,18]>ratio\*data[,28])

ambiindices=which(data[,28]>0 & data[,18]>0)

ambiindices=setdiff(setdiff(ambiindices,c1indices),c2indices)

data[c1indices,18]=0

data[c2indices,28]=0

data[ambiindices,c(28,18)]=0

rownames(data)=data[,1]

x=data

#remove donor/recipient ambiguous clones, either real shared clones, or ambiguous raised by CFSE-MLR sorting error

ambiguous=which((x[,13]>0 | x[,15]>0)&(x[,17]>0 | x[,19]>0))

ambiguous=union(ambiguous,which((x[,13]>0 | x[,15]>0)&(x[,16]>0 | x[,18]>0)))

ambiguous=union(ambiguous,which((x[,17]>0 | x[,19]>0)&(x[,12]>0 | x[,14]>0)))

x=x[setdiff(1:nrow(x),ambiguous),]

#additionally remove stim vs unstim ambiguous within recipient preTx (R4L, R4U, R8L, R8U) or donor preTx (D4L, D4U, D8L, D8U) samples; which can be considered as additional step of optimization for CD4 and CD8 sorting error.

#R4L vs R8U

normalize(x[,c(16,19)])

ratio=2

c1indices=which(x[,16]>0 & x[,19]>0 & x[,16]>ratio\*x[,19])

c2indices=which(x[,16]>0 & x[,19]>0 & x[,19]>ratio\*x[,16])

ambiindices=which(x[,16]>0 & x[,19]>0)

ambiindices=setdiff(setdiff(ambiindices,c1indices),c2indices)

x[c1indices,19]=0

x[c2indices,16]=0

x[ambiindices,c(16,19)]=0

#R8L vs R4U

normalize(x[,c(18,17)])

ratio=2

c1indices=which(x[,18]>0 & x[,17]>0 & x[,18]>ratio\*x[,17])

c2indices=which(x[,18]>0 & x[,17]>0 & x[,17]>ratio\*x[,18])

ambiindices=which(x[,18]>0 & x[,17]>0)

ambiindices=setdiff(setdiff(ambiindices,c1indices),c2indices)

x[c1indices,17]=0

x[c2indices,18]=0

x[ambiindices,c(18,17)]=0

#D4L vs D8U

normalize(x[,c(12,15)])

ratio=2

c1indices=which(x[,12]>0 & x[,15]>0 & x[,12]>ratio\*x[,15])

c2indices=which(x[,12]>0 & x[,15]>0 & x[,15]>ratio\*x[,12])

ambiindices=which(x[,12]>0 & x[,15]>0)

ambiindices=setdiff(setdiff(ambiindices,c1indices),c2indices)

x[c1indices,15]=0

x[c2indices,12]=0

x[ambiindices,c(12,15)]=0

#D8L vs D4U

normalize(x[,c(14,13)])

ratio=2

c1indices=which(x[,14]>0 & x[,13]>0 & x[,14]>ratio\*x[,13])

c2indices=which(x[,14]>0 & x[,13]>0 & x[,13]>ratio\*x[,14])

ambiindices=which(x[,14]>0 & x[,13]>0)

ambiindices=setdiff(setdiff(ambiindices,c1indices),c2indices)

x[c1indices,13]=0

x[c2indices,14]=0

x[ambiindices,c(14,13)]=0

write.table(x,file ="P:/CCTI\_USERS/Jianing Fu/Adaptive samples and analysis/Pt14 iITx/Pt14 01082020 resolve ambiguous updated.tsv",quote=F,row.names=F,col.names=T, sep="\t")

#resolve ambiguous raised by post-Tx unstim and CFSE-MLR

#post-Tx R4U (29) vs R8U (31)

normalize(x[,c(29,31)])

ratio=5

c1indices=which(x[,29]>0 & x[,31]>0 & x[,29]>ratio\*x[,31])

c2indices=which(x[,29]>0 & x[,31]>0 & x[,31]>ratio\*x[,29])

ambiindices=which(x[,29]>0 & x[,31]>0)

ambiindices=setdiff(setdiff(ambiindices,c1indices),c2indices)

x[c1indices,31]=0

x[c2indices,29]=0

x[ambiindices,c(29,31)]=0

#post-Tx R4L (28) vs R8L (30)

normalize(x[,c(28,30)])

ratio=5

c1indices=which(x[,28]>0 & x[,30]>0 & x[,28]>ratio\*x[,30])

c2indices=which(x[,28]>0 & x[,30]>0 & x[,30]>ratio\*x[,28])

ambiindices=which(x[,28]>0 & x[,30]>0)

ambiindices=setdiff(setdiff(ambiindices,c1indices),c2indices)

x[c1indices,30]=0

x[c2indices,28]=0

x[ambiindices,c(28,30)]=0

#remove donor/recipient ambiguous clones, either real shared clones, or ambiguous raised by CFSE-MLR sorting error

ambiguous=which((x[,13]>0 | x[,15]>0)&(x[,29]>0 | x[,31]>0))

ambiguous=union(ambiguous,which((x[,13]>0 | x[,15]>0)&(x[,28]>0 | x[,30]>0)))

ambiguous=union(ambiguous,which((x[,29]>0 | x[,31]>0)&(x[,12]>0 | x[,14]>0)))

x=x[setdiff(1:nrow(x),ambiguous),]

#additionally remove stim vs unstim ambiguous within recipient post-Tx (R’4L, R’4U, R’8L, R’8U); which can be considered as additional step of optimization for CD4 and CD8 sorting error.

#R'4L vs R'8U

normalize(x[,c(28,30)])

ratio=2

c1indices=which(x[,28]>0 & x[,30]>0 & x[,28]>ratio\*x[,30])

c2indices=which(x[,28]>0 & x[,30]>0 & x[,30]>ratio\*x[,28])

ambiindices=which(x[,28]>0 & x[,30]>0)

ambiindices=setdiff(setdiff(ambiindices,c1indices),c2indices)

x[c1indices,30]=0

x[c2indices,28]=0

x[ambiindices,c(28,30)]=0

#R'8L vs R'4U

normalize(x[,c(30,29)])

ratio=2

c1indices=which(x[,30]>0 & x[,29]>0 & x[,30]>ratio\*x[,29])

c2indices=which(x[,30]>0 & x[,29]>0 & x[,29]>ratio\*x[,30])

ambiindices=which(x[,30]>0 & x[,29]>0)

ambiindices=setdiff(setdiff(ambiindices,c1indices),c2indices)

x[c1indices,29]=0

x[c2indices,30]=0

x[ambiindices,c(30,29)]=0

write.table(x,file ="P:/CCTI\_USERS/Jianing Fu/Adaptive samples and analysis/Pt14 iITx/Pt14 01082020 resolve ambiguous updated with postTx MLR.tsv",quote=F,row.names=F,col.names=T, sep="\t")

# add "fold=2" below, use 2 fold expansion (default 5 fold)

# add "freq=0.00002" below, use freq=0.00002 for template counts; use default freq=0.00001 for read counts

#post-Tx POD456 HVG direction

# For CD4 HvG, cd4.HVG=x[,c(unstim(R4U),stim(R4L))]

# For CD8 HvG, cd8.HVG=x[,c(unstim(R8U),stim(R8L))]

cd4.HVG= x[,c(29,28)]

cd8.HVG= x[,c(31,30)]

allo.HVG=listAlloreactive(cd4.HVG,cd8.HVG, fold=2, freq=0.00002)

#When you define allo=listAlloreactive(cd4,cd8),then you should get cd4 alloreactives stored in allo[[1]] and cd8 alloreactives stored in allo[[2]]. If you do allo=union(allo[[1]],allo[[2]]), then allo contains all the alloreactives, and you can define nonallo=setdiff(rownames(x),allo)

length(allo.HVG[[1]])

[1] 446

length(allo.HVG[[2]])

[1] 173

write.table(allo.HVG[[1]],file ="P:/CCTI\_USERS/Jianing Fu/Adaptive samples and analysis/Pt14 iITx/Pt14 CD4 post-Tx POD456 HVGlist.txt",quote=F,row.names=F,col.names=F, sep="\t")

write.table(allo.HVG[[2]],file ="P:/CCTI\_USERS/Jianing Fu/Adaptive samples and analysis/Pt14 iITx/Pt14 CD8 post-Tx POD456 HVGlist.txt",quote=F,row.names=F,col.names=F, sep="\t")

rCD4mappable=rownames(x[(x[,29]+x[,28])>0,])

rCD8mappable=rownames(x[(x[,31]+x[,30])>0,])

CD4nonHVG=setdiff(rCD4mappable,allo.HVG[[1]])

CD8nonHVG=setdiff(rCD8mappable,allo.HVG[[2]])

write.table(CD4nonHVG,file ="P:/CCTI\_USERS/Jianing Fu/Adaptive samples and analysis/Pt14 iITx/Pt14 CD4nonHVG rmappable list post-Tx POD456.txt",quote=F,row.names=F,col.names=F, sep="\t")

write.table(CD8nonHVG,file ="P:/CCTI\_USERS/Jianing Fu/Adaptive samples and analysis/Pt14 iITx/Pt14 CD8nonHVG rmappable list post-Tx POD456.txt",quote=F,row.names=F,col.names=F, sep="\t")

length(rCD4mappable)

[1] 15438

length(rCD8mappable)

[1] 9372

length(CD4nonHVG)

[1] 14992

length(CD8nonHVG)

[1] 9199

rmappable=rownames(x[(x[,29]+x[,28]+x[,31]+x[,30])>0,])

length(rmappable)

[1] 24810

write.table(rmappable,file ="P:/CCTI\_USERS/Jianing Fu/Adaptive samples and analysis/Pt14 iITx/Pt14 recipient mappable list post-Tx POD456.txt",quote=F,row.names=F,col.names=F, sep="\t")

length(rCD4mappable)+length(rCD8mappable)

[1] 24810

length(intersect(rCD4mappable,rCD8mappable))

[1] 0

#Pt14 integrate pre- and post-Tx MLRs

#load resolve ambiguous tsv

Pt14\_new <- read.delim("~/Desktop/10x TRM 2021/pre post MLR analysis/Pt14 01082020 resolve ambiguous updated with postTx MLR.tsv")

colnames(Pt14\_new)

[1] "Rearrangement" "Sum..Templates."

[3] "Present.In" "Pt14\_Post\_PBMC\_day0030"

[5] "Pt14\_Post\_PBMC\_day0456" "Pt14\_Post\_ileum\_day0009"

[7] "Pt14\_Post\_ileum\_day0016" "Pt14\_Post\_ileum\_day0156"

[9] "Pt14\_Post\_ileum\_day0226" "Pt14\_Post\_ileum\_day0527"

[11] "Pt14\_Post\_ileum\_day0717" "Pt14\_Pre\_Donor\_CD4\_CFSElo"

[13] "Pt14\_Pre\_Donor\_CD4\_unstim" "Pt14\_Pre\_Donor\_CD8\_CFSElo"

[15] "Pt14\_Pre\_Donor\_CD8\_unstim" "Pt14\_Pre\_Recipient\_CD4\_CFSElo"

[17] "Pt14\_Pre\_Recipient\_CD4\_unstim" "Pt14\_Pre\_Recipient\_CD8\_CFSElo"

[19] "Pt14\_Pre\_Recipient\_CD8\_unstim" "Pt14\_iITx\_Bx\_POD527\_Native\_colon"

[21] "Pt14\_iITx\_Bx\_POD527\_Tx\_colon" "Pt14\_iITx\_Bx\_POD717\_Native\_colon"

[23] "Pt14\_iITx\_Bx\_POD717\_Tx\_colon" "Pt14\_iITx\_POD1764\_PBMC\_T"

[25] "Pt14\_iITx\_POD1764\_Tx\_colon\_Bx\_T" "Pt14\_iITx\_POD1764\_ileum\_Bx\_Chronic\_Rej"

[27] "Pt14\_iITx\_POD1764\_native\_colon\_Bx\_T" "Pt14\_iITx\_POD456\_PBMC\_R.4L\_D\_SP"

[29] "Pt14\_iITx\_POD456\_PBMC\_R.4U" "Pt14\_iITx\_POD456\_PBMC\_R.8L\_D\_SP"

[31] "Pt14\_iITx\_POD456\_PBMC\_R.8U"

#unstim, stim

rcd4 <- Pt14\_new[,c(17,16)]

rcd4 <- normalize(rcd4)

rCD4HVG <- rownames(rcd4[rcd4[,2]>0.00002 & rcd4[,2] > rcd4[,1]\*2,])

rCD4NonHVG <- setdiff(rownames(Pt14\_new[Pt14\_new[,17]>0|Pt14\_new[,16]>0,]), rCD4HVG)

rcd8 <- Pt14\_new[,c(19,18)]

rcd8 <- normalize(rcd8)

rCD8HVG <- rownames(rcd8[rcd8[,2]>0.00002 & rcd8[,2] > rcd8[,1]\*2,])

rCD8NonHVG <- setdiff(rownames(Pt14\_new[Pt14\_new[,19]>0|Pt14\_new[,18]>0,]), rCD8HVG)

dcd4 <- Pt14\_new[,c(13,12)]

dcd4 <- normalize(dcd4)

rCD4GVH <- rownames(dcd4[dcd4[,2]>0.00002 & dcd4[,2] > dcd4[,1]\*2,])

rCD4NonGVH <- setdiff(rownames(Pt14\_new[Pt14\_new[,13]>0|Pt14\_new[,12]>0,]), rCD4GVH)

dcd8 <- Pt14\_new[,c(15,14)]

dcd8 <- normalize(dcd8)

rCD8GVH <- rownames(dcd8[dcd8[,2]>0.00002 & dcd8[,2] > dcd8[,1]\*2,])

rCD8NonGVH <- setdiff(rownames(Pt14\_new[Pt14\_new[,15]>0|Pt14\_new[,14]>0,]), rCD8GVH)

r2cd4 <- Pt14\_new[,c(29,28)]

r2cd4 <- normalize(r2cd4)

r2CD4HVG <- rownames(r2cd4[r2cd4[,2]>0.00002 & r2cd4[,2] > r2cd4[,1]\*2,])

r2CD4NonHVG <- setdiff(rownames(Pt14\_new[Pt14\_new[,29]>0|Pt14\_new[,28]>0,]), r2CD4HVG)

r2cd8 <- Pt14\_new[,c(31,30)]

r2cd8 <- normalize(r2cd8)

r2CD8HVG <- rownames(r2cd8[r2cd8[,2]>0.00002 & r2cd8[,2] > r2cd8[,1]\*2,])

r2CD8NonHVG <- setdiff(rownames(Pt14\_new[Pt14\_new[,31]>0|Pt14\_new[,30]>0,]), r2CD8HVG)

CD4HVG <- setdiff(rCD4HVG,c(rCD4NonHVG,rCD4GVH,rCD4NonGVH,rCD8HVG,rCD8NonHVG,rCD8GVH,rCD8NonGVH,r2CD8HVG,r2CD8NonHVG))

CD4NonHVG <- setdiff(rCD4NonHVG,c(rCD4HVG,rCD4GVH,rCD4NonGVH,rCD8HVG,rCD8NonHVG,rCD8GVH,rCD8NonGVH,r2CD8HVG,r2CD8NonHVG))

CD8HVG <- setdiff(rCD8HVG,c(rCD4NonHVG,rCD4GVH,rCD4NonGVH,rCD4HVG,rCD8NonHVG,rCD8GVH,rCD8NonGVH,r2CD4HVG,r2CD4NonHVG))

CD8NonHVG <- setdiff(rCD8NonHVG,c(rCD4HVG,rCD4GVH,rCD4NonGVH,rCD8HVG,rCD4NonHVG,rCD8GVH,rCD8NonGVH,r2CD4HVG,r2CD4NonHVG))

unmappable <- setdiff(rownames(Pt14\_new), c(rCD4HVG,rCD4NonHVG,rCD4GVH,rCD4NonGVH,rCD8HVG,rCD8NonHVG,rCD8GVH,rCD8NonGVH))

CD4H2VG <- setdiff(r2CD4HVG,c(r2CD4NonHVG,r2CD8HVG,r2CD8NonHVG, rCD8HVG,rCD8NonHVG))

CD4NonH2VG <- setdiff(r2CD4NonHVG,c(r2CD4HVG,r2CD8HVG,r2CD8NonHVG, rCD8HVG,rCD8NonHVG))

CD8H2VG <- setdiff(r2CD8HVG,c(r2CD4NonHVG,r2CD4HVG,r2CD8NonHVG, rCD4HVG,rCD4NonHVG))

CD8NonH2VG <- setdiff(r2CD8NonHVG,c(r2CD4HVG,r2CD8HVG,r2CD4NonHVG, rCD4HVG,rCD4NonHVG))

unmappable2 <- setdiff(rownames(Pt14\_new), c(r2CD4HVG,r2CD4NonHVG,r2CD8HVG,r2CD8NonHVG))

CD4presistentHVG <- intersect(CD4HVG, CD4H2VG)

CD4acquiredHVG <- intersect(CD4NonHVG, CD4H2VG)

CD4denovoHVG <- intersect(unmappable, CD4H2VG)

CD4tolerantHVG <- intersect(CD4HVG, CD4NonH2VG)

CD4persistentNonHVG <- intersect(CD4NonHVG, CD4NonH2VG)

CD4denovoNonHVG <- intersect(unmappable, CD4NonH2VG)

CD4missingHVG <- intersect(CD4HVG, unmappable2)

CD4missingNonHVG <- intersect(CD4NonHVG, unmappable2)

Unmappable <- intersect(unmappable, unmappable2)

CD8presistentHVG <- intersect(CD8HVG, CD8H2VG)

CD8acquiredHVG <- intersect(CD8NonHVG, CD8H2VG)

CD8denovoHVG <- intersect(unmappable, CD8H2VG)

CD8tolerantHVG <- intersect(CD8HVG, CD8NonH2VG)

CD8persistentNonHVG <- intersect(CD8NonHVG, CD8NonH2VG)

CD8denovoNonHVG <- intersect(unmappable, CD8NonH2VG)

CD8missingHVG <- intersect(CD8HVG, unmappable2)

CD8missingNonHVG <- intersect(CD8NonHVG, unmappable2)

matrix1 <- as.data.frame(matrix(nrow=17,ncol=17))

rownames(matrix1) <- c("CD4presistentHVG","CD4acquiredHVG","CD4denovoHVG",

"CD4tolerantHVG","CD4persistentNonHVG","CD4denovoNonHVG",

"CD4missingHVG","CD4missingNonHVG","Unmappable",

"CD8presistentHVG","CD8acquiredHVG","CD8denovoHVG",

"CD8tolerantHVG","CD8persistentNonHVG","CD8denovoNonHVG",

"CD8missingHVG","CD8missingNonHVG") -> colnames(matrix1)

subsets <- list(CD4presistentHVG,CD4acquiredHVG,CD4denovoHVG,

CD4tolerantHVG,CD4persistentNonHVG,CD4denovoNonHVG,

CD4missingHVG,CD4missingNonHVG,Unmappable,

CD8presistentHVG,CD8acquiredHVG,CD8denovoHVG,

CD8tolerantHVG,CD8persistentNonHVG,CD8denovoNonHVG,

CD8missingHVG,CD8missingNonHVG)

for (m in 1:17){

for (n in 1:17) {

matrix1[m,n] <- length(intersect(subsets[[m]], subsets[[n]]))

}

}

write.table(matrix1, file ="~/Desktop/10x TRM 2021/pre post MLR analysis/Pt14 17 subsets uniqueclonenubmer\_JF.tsv",row.names=T,col.names=T, sep="\t")

matrix2 <- as.data.frame(matrix(nrow=17\*4,ncol=(ncol(Pt14\_new)-2)))

rownames(matrix2) <- paste0(rep(c("CD4presistentHVG","CD4acquiredHVG","CD4denovoHVG",

"CD4tolerantHVG","CD4persistentNonHVG","CD4denovoNonHVG",

"CD4missingHVG","CD4missingNonHVG","Unmappable",

"CD8presistentHVG","CD8acquiredHVG","CD8denovoHVG",

"CD8tolerantHVG","CD8persistentNonHVG","CD8denovoNonHVG",

"CD8missingHVG","CD8missingNonHVG"),4), c(rep("template#", 17), rep("template%", 17), rep("uniqueclone#", 17), rep("uniqueclone%", 17)))

colnames(matrix2) <- colnames(Pt14\_new[,4:ncol(Pt14\_new)])

length1 <- function(x){

return(length(which(x>0)))

}

#Pt14\_new\_v2 only choose defined 17 subsets in the demoninator when calculating %, which is not what we want, use total template counts instead.

Pt14\_new\_v2 <- Pt14\_new

Pt14\_new\_v2 <- Pt14\_new\_v2[rownames(Pt14\_new\_v2) %in% c(CD4presistentHVG,CD4acquiredHVG,CD4denovoHVG,

CD4tolerantHVG,CD4persistentNonHVG,CD4denovoNonHVG,

CD4missingHVG,CD4missingNonHVG,Unmappable,

CD8presistentHVG,CD8acquiredHVG,CD8denovoHVG,

CD8tolerantHVG,CD8persistentNonHVG,CD8denovoNonHVG,

CD8missingHVG,CD8missingNonHVG), ]

for (m in 1:17) {

for (n in 4:ncol(Pt14\_new\_v2)) {

matrix2[m,n-2] <- sum(Pt14\_new\_v2[subsets[[m]], n])

matrix2[m+17,n-2] <- sum(Pt14\_new\_v2[subsets[[m]], n])/sum(Pt14\_new\_v2[,n])\*100

matrix2[m+17\*2,n-2] <- length1(Pt14\_new\_v2[subsets[[m]], n])

matrix2[m+17\*3,n-2] <- length1(Pt14\_new\_v2[subsets[[m]], n])/length1(Pt14\_new\_v2[, n])\*100

}

}

write.table(matrix2, file ="~/Desktop/10x TRM 2021/pre post MLR analysis/Pt14 17 subsets number% in uniqueclone and template\_JF.tsv",row.names=T,col.names=T, sep="\t")

### matrix first row: column names need slightly adjust. calculation takes about ~10-15mins

#total template counts

apply(Pt14\_new[,c(4:31)],2,sum)

#generate data matrix

matrix<-apply(Pt14\_new[,c(4:31)],2, sum)

write.table(matrix,file ="~/Desktop/10x TRM 2021/pre post MLR analysis/Pt14 total template counts.tsv",quote=F,row.names=T,col.names=T, sep="\t")