

RAPProject

Mingzhi Ye

2022/10/10

```
sce0 <- readRDS('toyTCRdata.rds')
sce <- clonoStats(sce0, method = 'EM')

#A<-sce0$contigs[sce0$contigs[, 'chain'] %in% c('TRA','TRB')]
# for (i in 1:length(sce0$contigs)){
#   #sce0$contigs[[i]] "DFrame" "S4Vectors"
#   #sce0$contigs[[i]]<-sce0$contigs[[i]] %>% filter(chain %in% c("TRA","TRB"))
#   #sce0$contigs[[i]]<-sce0$contigs[[i]][filter(sce0$contigs[[i]]$chain %in% c("TRA","TRB")),]
#   sce0$contigs[[i]]<-sce0$contigs[[i]][order(sce0$contigs[[i]]$chain),]
# }
# }
```

Generate sample pool

```
AminoAcids=c('A','R','N','D','C','Q','E','G','H','I','L','K','M','F','P','S','T','W','Y','V')
getCDR3 <- function(x){

  cdr3=sample(AminoAcids,15, replace=T)
  return(paste(cdr3,collapse = ''))
}
poolSize=1000
TemplateClonotype=sce0$contigs[[4]][c(2,1),]
getClonotypeForPool<-function(clonotype_index, CDRAAlpha, CDRBeta){
  result=TemplateClonotype
  result$cdr3[1]=CDRAAlpha
  result$cdr3[2]=CDRBeta
  return(result)
}
getSamplePool <- function(poolSize){
  Alpha_pool=sapply(1:poolSize,getCDR3,simplify = TRUE)
  Beta_pool=sapply(1:poolSize,getCDR3,simplify = TRUE)

  CDR_Alpha=sample(Alpha_pool,poolSize, replace=T)
  CDR_Beta=sample(Beta_pool,poolSize, replace=T)

  barcode <- paste('cell', 1:poolSize)
```

```

samplePool <- lapply(1:poolSize, function(clonotype_index){
  getClonotypeForPool(clonotype_index, CDR_Alpha[clonotype_index], CDR_Beta[clonotype_index])
})
samplePool <- SplitDataFrameList(samplePool)
samplePool[, 'barcode'] <- barcode
samplePool[, 'sample'] <- 'sim'
names(samplePool) <- barcode

# sample1<-as(sample, 'SplitDataFrameList')
return(samplePool)
}
samplePool=getSamplePool(1000)

```

Functions for generate a sample

```

getClonotype<-function(clonotype_index, numberA, numberB, errorProb = .01){
  result=DataFrame()
  indexInPool=sample.int(poolSize, 3, replace=FALSE)

  if (numberA>=1){
    if(runif(1) < errorProb){
      result=rbind(result,samplePool[[indexInPool[2]]][1,])
    }else{
      result=rbind(result,samplePool[[indexInPool[1]]][1,])
    }
    numberA=numberA-1
  }
  if (numberB>=1){
    if(runif(1) < errorProb){
      result=rbind(result,samplePool[[indexInPool[3]]][2,])
    }else{
      result=rbind(result,samplePool[[indexInPool[1]]][2,])
    }

    numberB=numberB-1
  }
  while(numberA>0 | numberB>0){
    indexInPool=sample.int(poolSize, 1, replace=TRUE)
    if (numberA>=1){
      result=rbind(result,samplePool[[indexInPool]][1,])
      numberA=numberA-1
    }
    else{
      result=rbind(result,samplePool[[indexInPool]][2,])
      numberB=numberB-1
    }
  }
  return(result)
}

```

Use the functions to generate a sample with size 50

However, function 'clonoStats' is not compatible with SimpleDFrameList, how to transform SimpleDFrameList to CompressedSplitDFrameList

```
getSample <- function(samplesize){
  sample=DataFrame()
  DistributeTRA=sum(sce0$contigs[, 'chain']=='TRA')
  DistributeTRB=sum(sce0$contigs[, 'chain']=='TRB')
  # table(DistributeTRA,DistributeTRB)
  IndexnAnBMap=cbind(DistributeTRA,DistributeTRB)
  RandomIntegers <- sample(1:poolSize, samplesize, replace=T)

  barcode <- paste('cell', 1:samplesize)

  sample <- lapply(1:samplesize, function(clonotype_index){
    getClonotype(clonotype_index, IndexnAnBMap[RandomIntegers[clonotype_index],1], IndexnAnBMap[RandomIntegers[clonotype_index],2])
  })
  sample <- SplitDataFrameList(sample)
  sample[, 'barcode'] <- barcode
  sample[, 'sample'] <- 'sim'
  names(sample) <- barcode

  # sample1<-as(sample, 'SplitDFrameList')
  return(sample)
}

# samplelist=getSample(SampleSize)
# EMpredicted <- clonoStats(samplelist, method = 'EM')
# UNpredicted <- clonoStats(samplelist, method = 'unique')
```

Generate hashmap. Key is clonotype in Truth(sample pool), Value is percentage of the clonotype in Truth

```
TruthPercentageMap=hashmap()

for(i in 1:poolSize){

  key=paste(samplePool[[i]]$cdr3[1],samplePool[[i]]$cdr3[2],sep=' ')

  if(is.null(TruthPercentageMap[[key]])){
    TruthPercentageMap[[key]]<-1/poolSize
  }
  else{
    TruthPercentageMap[[key]]<-TruthPercentageMap[[key]]+1/poolSize
  }
}
```

Get Variation Distance

```
getDistance<-function(predicted){

  clonotypes=clonoNames(predicted)
  abundance=clonoAbundance(predicted)
  SumAbundance=sum(abundance)
  distance=0
  seen <- hashmap()
  for(i in 1:length(abundance)){
    if(is.null(TruthPercentageMap[[clonotypes[i]]])){
      distance=distance+abundance[i]/SumAbundance
    }
    else{
      seen[[clonotypes[i]]]<-TruthPercentageMap[[clonotypes[i]]]
      distance=distance+abs(abundance[i]/SumAbundance-TruthPercentageMap[[clonotypes[i]]])
    }
  }

  distance=distance+Reduce("+", values(TruthPercentageMap)) -Reduce("+", values(seen))
  return(distance)
}
```

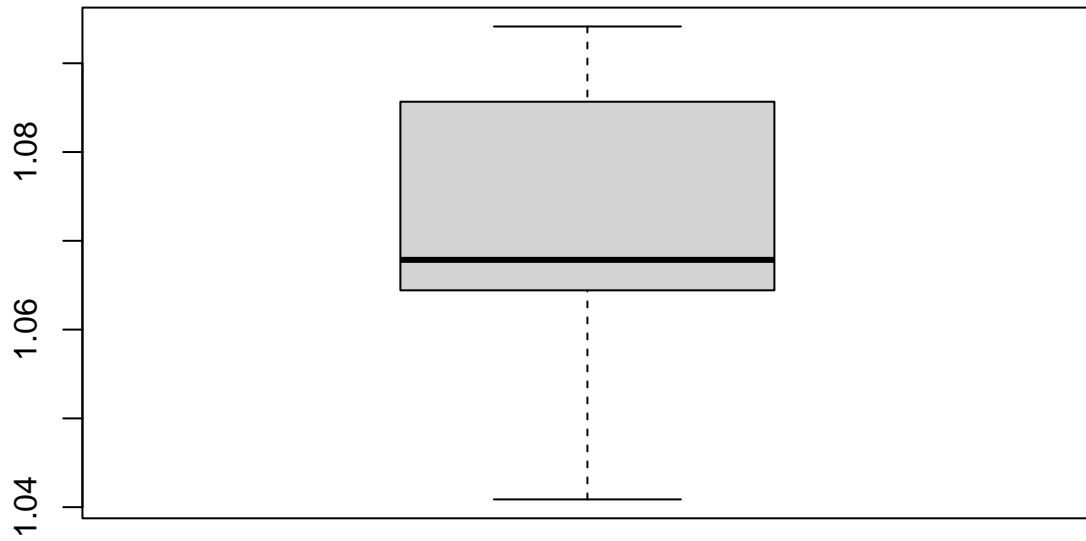
Calculate distances for samples

```
DistanceListFromEM=c()
DistanceListFromUN=c()
for(i in 1:10){
  samplelist=getSample(1000)
  EMpredicted <- clonoStats(samplelist, method = 'EM')
  distance1=getDistance(EMpredicted)
  DistanceListFromEM[i]=distance1
  UNpredicted <- clonoStats(samplelist, method = 'unique')
  distance2=getDistance(UNpredicted)
  DistanceListFromUN[i]=distance2
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

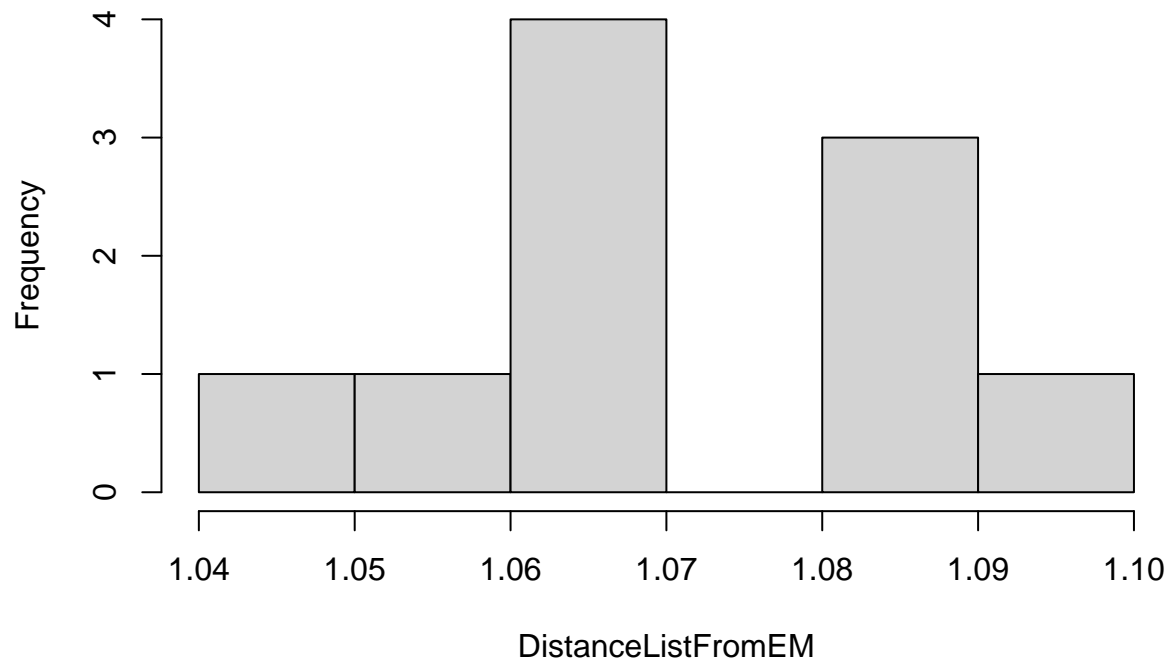
Visualize

```
boxplot(DistanceListFromEM)
```

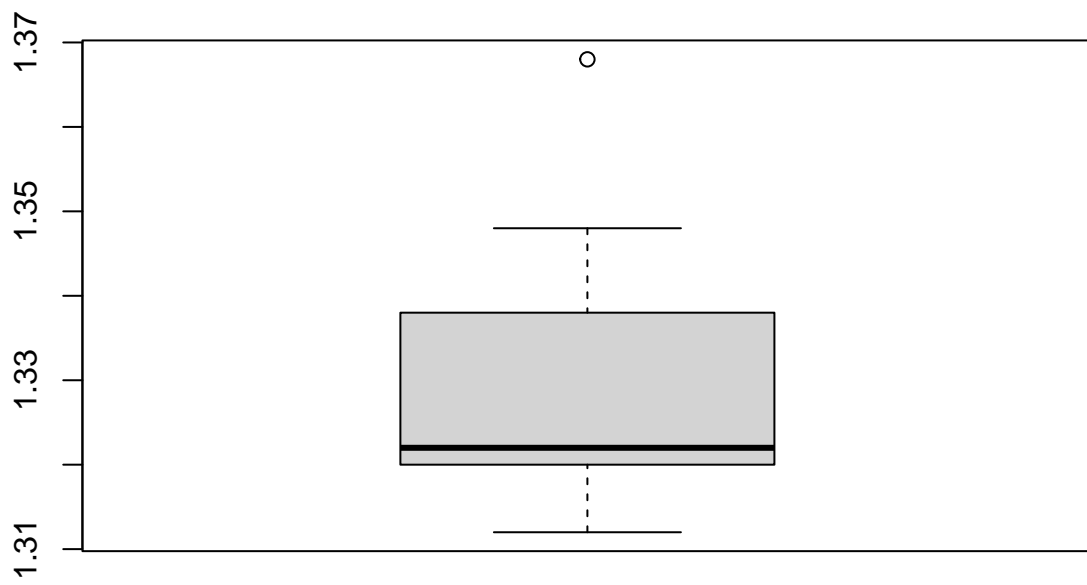


```
hist(DistanceListFromEM)
```

Histogram of DistanceListFromEM

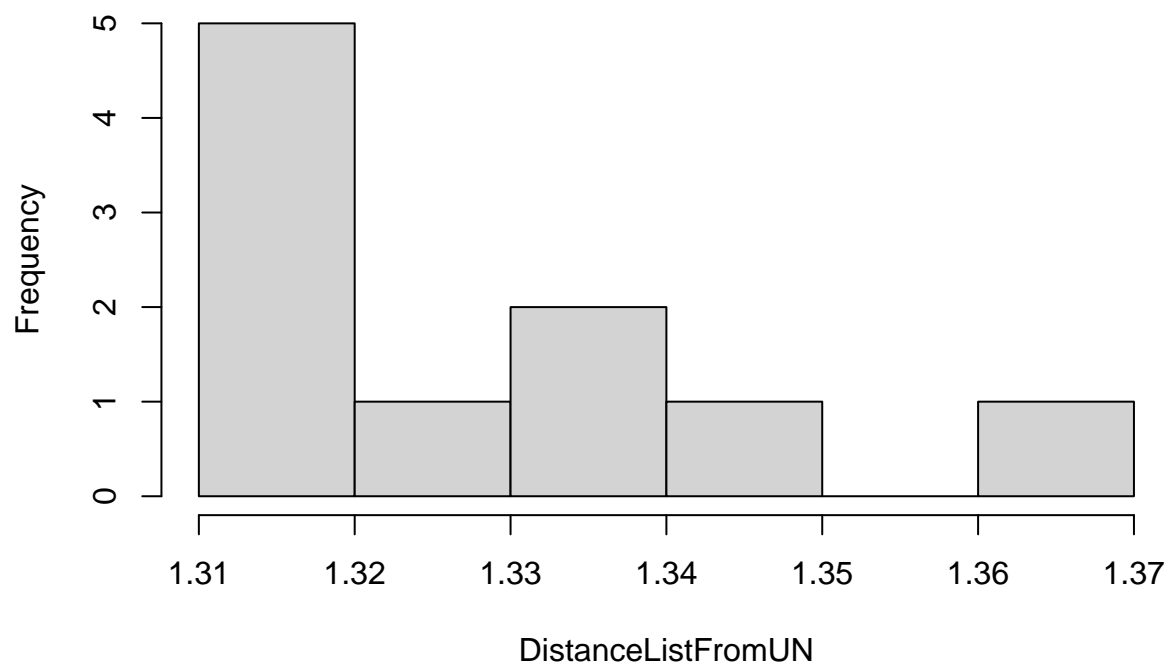


```
boxplot(DistanceListFromUN)
```



```
hist(DistanceListFromUN)
```

Histogram of DistanceListFromUN



```
print(median(DistanceListFromEM))
```

```
## [1] 1.067856
```

```
print(median(DistanceListFromUN))
```

```
## [1] 1.322
```

```
print(mean(DistanceListFromEM))
```

```
## [1] 1.071464
```

```
print(mean(DistanceListFromUN))
```

```
## [1] 1.33
```