Finding Significantly Edited Bases

Jaydee Sereewit

11/7/2021

Libraries

```
library(xlsx)
```

Reading in and trimming the data

```
edits.raw=read.csv("aes_profile.csv")
```

restricting the data to only bases for which there is mRNA coverage for all 6 octopuses.

```
edits=edits.raw[edits.raw$octo1_tot>0&edits.raw$octo2_tot>0&edits.raw$octo3_tot>0&edits.raw$octo4_tot>0
```

Finding the actual edited base

None of the fields that are currently in the dataset actually give what the base edited to for the weak editing sites. For the strong edits you can gather this data from the mrna_cons field, but for the weak edits this field will match the gdna_cons field and not represent what the base was changed to. This bit of code is intended to give what is the most common mRNA base that does not match the gDNA.

```
edits$A=edits$octo1_A+edits$octo2_A+edits$octo3_A+edits$octo4_A+edits$octo5_A+edits$octo6_A
edits$C=edits$octo1_C+edits$octo2_C+edits$octo3_C+edits$octo4_C+edits$octo5_C+edits$octo6_C
edits$G=edits$octo1_G+edits$octo2_G+edits$octo3_G+edits$octo4_G+edits$octo5_G+edits$octo6_G
edits$T=edits$octo1_T+edits$octo2_T+edits$octo3_T+edits$octo4_T+edits$octo5_T+edits$octo6_T

edits$edited=NA

base.edit=apply(edits[edits$gdna_con=="A",44:46],1,FUN="which.max")
base.edit[base.edit==1]="C"
base.edit[base.edit==2]="G"
base.edit[base.edit==3]="T"
edits$edited[edits$gdna_con=="A"]=base.edit


base.edit=apply(edits[edits$gdna_con=="C",c(43,45,46)],1,FUN="which.max")
base.edit[base.edit==1]="A"
base.edit[base.edit==2]="G"
base.edit[base.edit==3]="T"
edits$edited[edits$gdna_con=="C"]=base.edit

base.edit=apply(edits[edits$gdna_con=="G",c(43,44,46)],1,FUN="which.max")
```

```
base.edit[base.edit==1]="A"
base.edit[base.edit==2]="C"
base.edit[base.edit==3]="T"
edits$edited[edits$gdna_con=="G"]=base.edit

base.edit=apply(edits[edits$gdna_con=="T",c(43:45)],1,FUN="which.max")
base.edit[base.edit==1]="A"
base.edit[base.edit==2]="C"
base.edit[base.edit==3]="G"
edits$edited[edits$gdna_con=="T"]=base.edit
```

Differential editing significant testing

Calculation of editing percentages for each octopus

```
edits$octo1_per=0
edits$octo2_per=0
edits$octo3_per=0
edits$octo4_per=0
edits$octo5_per=0
edits$octo6_per=0

bases=c("A","C","G","T")

start.per.columns=which(colnames(edits)=="octo1_per")
spc=start.per.columns

for (j in 0:5){
  for (i in 0:3){
    per.column=start.per.columns+j
    base.column=(5*j)+6+i
    tot.column=10+5*j
    #This next line I used to troubleshoot.  Can keep commented out unless troubleshooting.
    #print(paste("Result Column: ",per.column,", Base Column: ",base.column,", Total Column: ",tot.colu
    edits[edits$gdna_con==bases[i+1],start.per.columns+j]=1-edits[edits$gdna_con==bases[i+1],((5*j)+6+i
  }
}
```

Now we run the randomization t-test on each base. First we calculate the t-stat for all 55k edits.

```
diff=abs(apply(edits[spc:(spc+2)],1,mean)-apply(edits[(spc+3):(spc+5)],1,mean))
denom1=apply(edits[spc:(spc+2)],1,sd)^2/apply(edits[spc:(spc+2)],1,length)
denom2=apply(edits[(spc+3):(spc+5)],1,sd)^2/apply(edits[(spc+3):(spc+5)],1,length)
edits$tstat=diff/sqrt(denom1+denom2)
```

Next, we make the bank of shuffled editing values. To make this repeatable, I am setting the seed to 56 to help reproducibility.

```
set.seed(56)

B=1000000

shuf=cbind(
  sample(as.matrix(edits[,spc:(spc+5)]),B,replace=T),
  sample(as.matrix(edits[,spc:(spc+5)]),B,replace=T),
```

```
  sample(as.matrix(edits[,spc:(spc+5)]),B,replace=T),
  sample(as.matrix(edits[,spc:(spc+5)]),B,replace=T),
  sample(as.matrix(edits[,spc:(spc+5)]),B,replace=T),
  sample(as.matrix(edits[,spc:(spc+5)]),B,replace=T)
)
```

Now we calculate a randomized bank of t-stats from shuffled editing values.

```
diff.shuf=abs(apply(shuf[,1:3],1,mean)-apply(shuf[,4:6],1,mean))
denom1.shuf=apply(shuf[,1:3],1,sd)^2/apply(shuf[,1:3],1,length)
denom2.shuf=apply(shuf[,4:6],1,sd)^2/apply(shuf[,4:6],1,length)
tstat.shuf=diff.shuf/sqrt(denom1.shuf+denom2.shuf)
```

Finally, we compare the actual t-stats form the data to the randomization t-stat bank to find how many are more extreme the the actual t-stats, and that is the p-value.

```
edits$pval=0

for (i in 1:nrow(edits)){
  edits$pval[i]=sum(tstat.shuf>edits$tstat[i],na.rm = T)/length(tstat.shuf)
}
```

We now have a huge pile of p-values, and suffer from ~55k multiple comparisons.
To solve this we can apply a Benjamini & Hochberg false discovery rate correction to the p-values.

```
edits$padj=p.adjust(edits$pval,method="BH")
edits$pval[is.nan(edits$tstat)]=NA
edits$padj[is.nan(edits$tstat)]=NA
```

Ok, lets see how many significant edits there are at a false discovery rate of 10% or less:

```
sum(edits$padj<=0.1,na.rm = T)
```

```
## [1] 744
```

And how many are A to G edits?

```
sum(edits$padj<=0.1&edits$gdna_con=="A"&edits$edited=="G",na.rm = T)
```

```
## [1] 136
```

Writing significant edits out to file

We will write out all of the bases that have a false discovery rate of 0.1, or 10%, or less.

```
sig.edits=edits[edits$padj<=0.1,]
sig.edits=sig.edits[!is.na(sig.edits$padj),]
write.xlsx(sig.edits,"Significant_Edits.xlsx")
```