Analyses of APA dynamics across rice tissues with the movAPA package

Xiaohui Wu, Wenbin Ye, Tao Liu, Hongjuan Fu

Last modified 2023-10-10

Contents

1 Overview

We investigated the application of movAPA on a poly(A) site dataset of 14 tissues in *Oryza sativa japonica* from 3'end sequencing (Fu, et al., 2016). We used a subset of the rice data containing 1233 PACs in 455 genes from three tissues (embryo, anther, and mature pollen) for demonstration. The original dataset containing full list of PACs can be downloaded from plantAPAdb (Zhu et al, 2019). Here the poly(A) sites are already poly(A) site clusters (PACs) which were grouped from nearby cleavage sites.

2 Preparations

2.1 Rice PAC data

movAPA implemented the *PACdataset* object for storing the expression levels and annotation of PACs from various conditions/samples. Almost all analyses of poly(A) site data in movAPA are based on the *PACdataset*. The "counts" matrix is the first element in the array list of *PACdataset*, which stores non-negative values representing expression levels of PACs. The "colData" matrix records the sample information and the "anno" matrix stores the genome annotation or additional information of the poly(A) site data.

The moveAPA package includes an example rice PAC data stored as a *PACdataset* object, which contains 1233 PACs from 455 genes. First load movAPA by *library(movAPA)* and then load the example data.

```r
library(movAPA, warn.conflicts = FALSE, quietly=TRUE)
data("PACds")
PACds
#> PAC# 1233
#> gene# 455
#>            nPAC
#> 3UTR        482
#> 5UTR         20
#> CDS          44
#> Ext_3UTR    391
#> intergenic  181
#> intron      115
#> sample# 9
#> anther1 anther2 anther3 embryo1 embryo2 ...
#> groups:
#> @colData...[9 x 1]
#>          group
#> anther1 anther
#> anther2 anther
#> @counts...[1233 x 9]
#>                      anther1 anther2 anther3 embryo1 embryo2 embryo3
#> Os01g0151600:2792379       0       1       0       2       1       1
#> Os01g0151600:2795487      11      16      17      60      55      51
#>                      maturePollen1 maturePollen2 maturePollen3
#> Os01g0151600:2792379             0             0             0
#> Os01g0151600:2795487            24             3            10
#> @colData...[9 x 1]
#>          group
#> anther1 anther
#> anther2 anther
#> @anno...[1233 x 10]
#>                      chr UPA_start UPA_end strand   coord    ftr         gene
#> Os01g0151600:2792379   1   2792363 2792427      + 2792379 intron Os01g0151600
#> Os01g0151600:2795487   1   2795427 2795509      + 2795487   3UTR Os01g0151600
#>                           gene_type ftr_start ftr_end
#> Os01g0151600:2792379 protein_coding   2792174 2792920
#> Os01g0151600:2795487 protein_coding   2795347 2795857
summary(PACds)
#> PAC# 1233
#> sample# 9
#> summary of expression level of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>       1       5      26     275     163   77248
#> summary of expressed sample# of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   1.000   3.000   6.000   5.637   8.000   9.000
#> gene# 455
#>            nPAC
#> 3UTR        482
#> 5UTR         20
#> CDS          44
#> Ext_3UTR    391
#> intergenic  181
```

```
#> intron      115
# Transform the older version of PACdataset to newer version; the counts slot was converted from data.f
# PACds@counts=asAnyMatrix(PACds@counts)
```

2.2 Reference genome

The reference genome is not necessary, while it is required for removing internal priming or poly(A) signal analyses. movAPA uses reference genome sequences that are represented as a *BSgenome* object or stored in a fasta file. The *BSgenome* of rice for this example can be downloaded from the github website of movAPA. Please refer to the *BSgenome* package for making a *BSgenome* object if there is no corresponding *BSgenome* package for your species. Alternatively, the genome assembly can be stored in a fasta file, which can also be used as input for movAPA.

```
devtools::load_all("/media/bmi/My Passport/scPACext_HC_288cells/movAPA/movAPA/BSgenome.Oryza.ENSEMBL.IRG
```

```
library("BSgenome.Oryza.ENSEMBL.IRGSP1", quietly = TRUE)
bsgenome <- BSgenome.Oryza.ENSEMBL.IRGSP1
```

2.3 Genome annotation

Genome annotation stored in a GFF/GTF file or a TXDB R object can be used for annotating PACs. The function *parseGff* or *parseGenomeAnnotation* is used to parse the given annotation and the processed annotation can be saved into an rdata object for further use. The GFF file or the processed rdata file of rice for this example can be downloaded from the github website of movAPA.

```
gffFile="Oryza_sativa.IRGSP-1.0.42.gff3"
gff=parseGff(gffFile)
save(gff, file='Oryza_sativa.IRGSP-1.0.42.gff.rda')
```

```
load('Oryza_sativa.IRGSP-1.0.42.gff.rda')
```

3 Preprocessing of PAC data

3.1 Remove internal priming artifacts

Internal priming (IP) artifacts can be removed by the *removePACdsIP* function. Here, PACs with six consecutive or more than six As within the -10 to +10 nt window are considered as internal priming artifacts. We scan the internal priming artifacts in PACds and get two *PACdatasets* recording internal priming PACs and real PACs. Since IP artifacts are already removed in the example PACds, we did not perform this step in this case study.

Note: removePACdsIP step should be performed in caution, because different parameter setting in removePACdsIP may result in very different number of internal priming artifacts.

```
PACdsIP=removePACdsIP(PACds, bsgenome, returnBoth=TRUE,
                      up=-10, dn=10, conA=6, sepA=7)
#> 345 IP PACs; 888 real PACs
length(PACdsIP$real)
#> [1] 888
```

```
length(PACdsIP$ip)
#> [1] 345
```

3.2 Group nearby cleavage sites

The function *mergePACds* can be used to group nearby cleavage sites into PACs. Here is an example to group nearby PACs within 100 bp into one PAC.

```
PACdsClust=mergePACds(PACds, d=100)
```

```
summary(PACds)
#> PAC# 1233
#> sample# 9
#> summary of expression level of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>       1       5      26     275     163   77248
#> summary of expressed sample# of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   1.000   3.000   6.000   5.637   8.000   9.000
#> gene# 455
#>            nPAC
#> 3UTR        482
#> 5UTR         20
#> CDS          44
#> Ext_3UTR    391
#> intergenic  181
#> intron      115
summary(PACdsClust)
#> PAC# 1132
#> sample# 9
#> summary of expression level of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>     1.0     5.0    29.0   299.5   175.0 77248.0
#> summary of expressed sample# of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   1.000   3.000   6.000   5.691   8.000   9.000
```

3.3 Merge multiple PAC datasets

The function *mergePACds* can also be used to merge multiple PACdatasets. Notably, the annotation columns (e.g., gene, ftr) are lost after merging, you need call *annotatePAC* to annotate the merged PACds.

In movAPA 0.2.0, a reference PACds can be used for merging PACdsList in a smarter way. Providing reference PACds for merging is useful when there are multiple large PAC lists to be merged, which can prevent generating PACs with a very wide range. If there is reference PACs from 3'seq, it is recommended to use it. Please see the help document of *mergePACds* for details.

```
## Constuct another demo PACdataset for merging.
PACds2=PACds
PACds2@anno$coord = PACds2@anno$coord + sample(-50:50, 1)
```

```r
## You may also change the sample names and group names.
# rownames(PACds2@colData)=paste0(rownames(PACds2@colData),'v2')
# PACds2@colData$group=paste0(PACds2@colData$group,'v2')
# colnames(PACds2@counts)=paste0(colnames(PACds2@counts),'v2')
## Construct a list of PACds to be merged.
PACdsList=list(pac1=PACds, pac2=PACds2)
```

```r
## Merge two PACdatasets, nearby PACs within 24bp of each other
## will be merged into one PAC.
pp=mergePACds(PACdsList, d=24)
#> mergePACds: there are 9 duplicated sample names in the PACdsList, will add .N to sample names of eac
#> mergePACds: total 2466 redundant PACs from 2 PACds to merge
#> mergePACds without refPACds: 2466 separate PACs reduce to 1233 PACs (d=24nt)
#> mergePACds: melted all counts tables, total 13900 triplet rows
#> mergePACds: link 2466 old PA IDs to 1233 new PA IDs by merge
#> mergePACds: convert 13900 triplets to dgCMatrix
#> mergePACds: construct Matrix[PA, sample], [1233, 18]
summary(pp)
#> PAC# 1233
#> sample# 18
#> summary of expression level of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>       2      10      52     550     326  154496
#> summary of expressed sample# of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    2.00    6.00   12.00   11.27   16.00   18.00
```

3.4 Normalization

The function *normalizePACds* can be called for normalization, which implements three strategies including TPM (Tags Per Million), the normalization method of DESeq (Anders and Huber, 2010), and the TMM method used in EdgeR (Robinson, et al., 2010).

Note: normalization should be performed in caution, because different methods would have significant and different impact on the data and downstream analysis!

```r
## Here normalization method TMM (or EdgeR) is used,
## while you may also choose TPM or DESeq.
PACds=normalizePACds(PACds, method='TMM')
#> converting counts to integer mode

## Library sizes after normalization.
colSums(PACds@counts)
#>       anther1       anther2       anther3       embryo1       embryo2
#>         20318         21529         21640         30468         31384
#>       embryo3  maturePollen1 maturePollen2 maturePollen3
#>         30768         76027         62261         54242
```

4 Annotate PACs

Users can use *annotatePAC* to annotate a *PACdataset* with a GFF/GTF file or a TXDB R object. Here we parse the genome annotation file in GFF3 format and save the processed annotation into a rdata object for further use.

```
load('Oryza_sativa.IRGSP-1.0.42.gff.rda')
```

Here is an example to annotate PACds with the genome annotation. Because the demo data already contains the annotation, we removed the annotation columns before calling *annotatePAC*.

```
PACds1=PACds
PACds1@anno[,c('gene','ftr','gene_type','ftr_start','ftr_end')]=NULL
PACds1=annotatePAC(PACds1, gff)
```

We can output the annotated PACds and the sample information to text files.

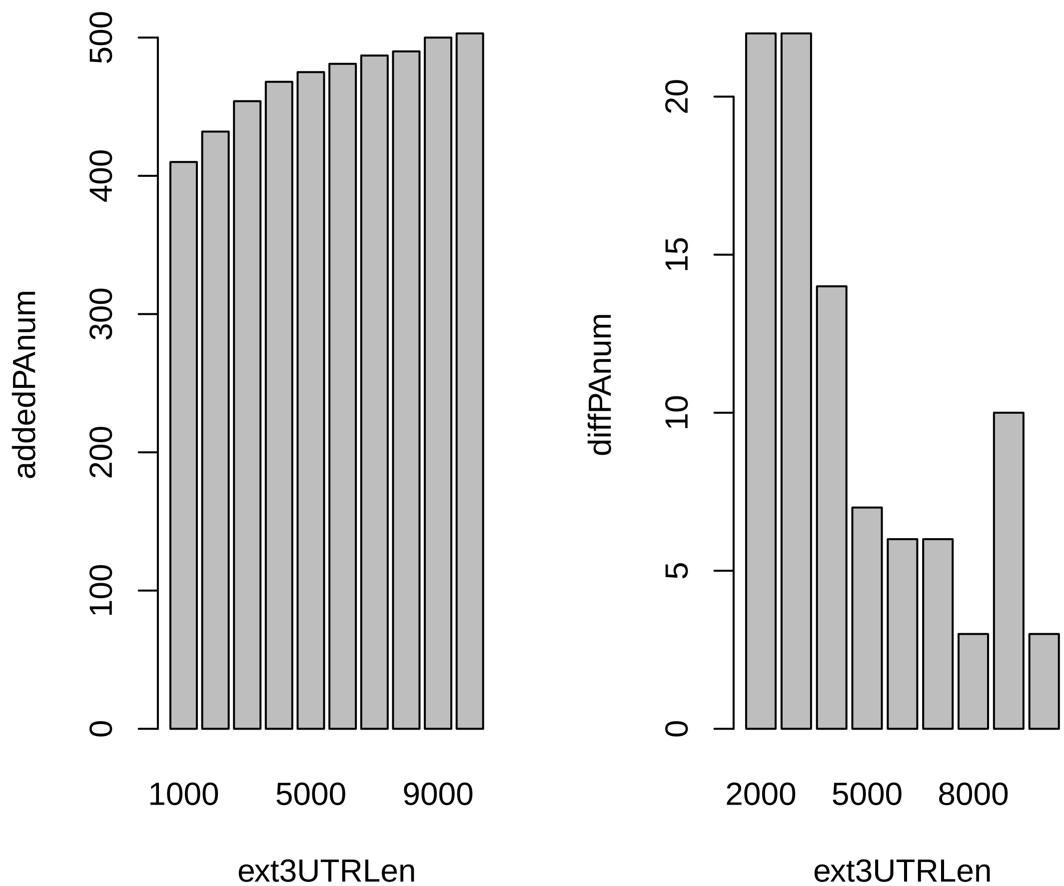
```
writePACds(PACds1, file='rice_pac_data.txt',
           colDataFile = 'rice_pac_data.coldata.txt')
```

4.1 Extending annotated 3'UTRs

Genes with or without annotated 3'UTR could be assigned an extended 3'UTR of a given length using the function `ext3UTRPACds`, which can improve the "recovery" of poly(A) sites falling within authentic 3'UTRs.

Before extending, we can calculate the number of PACs falling into extended 3'UTRs of different lengths.

```
testExt3UTR(PACds1, seq(1000, 10000, by=1000))
```




```
#>    ext3UTRLen addedPAnum
#> 1        1000        410
#> 2        2000        432
#> 3        3000        454
#> 4        4000        468
#> 5        5000        475
#> 6        6000        481
#> 7        7000        487
#> 8        8000        490
#> 9        9000        500
#> 10      10000        503
```

Here we extended 3'UTR length for 2000 bp. After extension, 70 PACs in intergenic region are now in extended 3'UTRs.

```
table(PACds1@anno$ftr)
#> 
#>       3UTR       5UTR        CDS intergenic     intron 
#>        482         18         44        572        117
```

```
PACds1=ext3UTRPACds(PACds1, ext3UTRlen=2000)
#> 432 PACs in extended 3UTR (ftr=intergenic >> ftr=3UTR)
#> Get 3UTR length (anno@toStop) for 3UTR/extended 3UTR PACs
table(PACds1@anno$ftr)
#>
#>       3UTR       5UTR        CDS intergenic     intron
#>        914         18         44        140        117
```

5 Statistical analyses of PACs

To make statistics of distributions of PACs for each sample, first we pooled replicates.

```
PACds1=subsetPACds(PACds, group='group', pool=TRUE)
head(PACds1@counts)
#>                         anther embryo maturePollen
#> Os01g0151600:2792379       1      3            0
#> Os01g0151600:2795487      33    116           65
#> Os01g0151600:2795636      51     60           11
#> Os01g0151600:2795858      17     45            3
#> Os01g0179300:4125553       6     13            0
#> Os01g0179300:4125845       3      1            0
```

Then we can make statistics of distribution of PACs using different PAT cutoffs. minPAT=5 means that only PACs with >=5 reads are used for statistics.

```
pstats=movStat(PACds1, minPAT=c(1, 5, 10, 20, 50, 60), ofilePrefix=NULL)
names(pstats)
#> [1] "pat1"  "pat5"  "pat10" "pat20" "pat50" "pat60"
pstats$pat10
#>              nPAC   nPAT nGene nAPAgene APAextent 3UTR_nPAT 5UTR_nPAT CDS_nPAT
#> anther        524  61855   340      135 0.3970588     33008       102       31
#> embryo        507  91051   307      150 0.4885993     66158        61      631
#> maturePollen  513 191317   332      122 0.3674699     47998        67        0
#> total         709 344223   388      200 0.5154639    147164       230      662
#>              Ext_3UTR_nPAT intergenic_nPAT intron_nPAT 3UTR_nPAC 5UTR_nPAC
#> anther               25951            2235         528       274         5
#> embryo               17494            6090         617       288         2
#> maturePollen        138323            3793        1136       277         3
#> total               181768           12118        2281       359         5
#>              CDS_nPAC Ext_3UTR_nPAC intergenic_nPAC intron_nPAC
#> anther              3           199              30          13
#> embryo              7           182              16          12
#> maturePollen        0           185              35          13
#> total               9           255              57          24
```

Statistical results can be visualized by barplots to show PAC#, PAT#, APA gene%, PAC%, PAT% across samples and genomic regions. Here we plot all statistical results with cutoffs 5 and 10, with each plot having two smaller plots corresponding to the two cutoffs.

```
plotPACdsStat(pstats, pdfFile='PACds_stat.pdf', minPAT=c(5,10))
```

Plot specific cutoffs and conditions.

```
plotPACdsStat(pstats, pdfFile='PACds_stat_anther_embryo.pdf',
              minPAT=c(5,10), conds=c('anther1','embryo1'))
```

Plot the overall distributions using pooled samples (total) and two cutoffs.

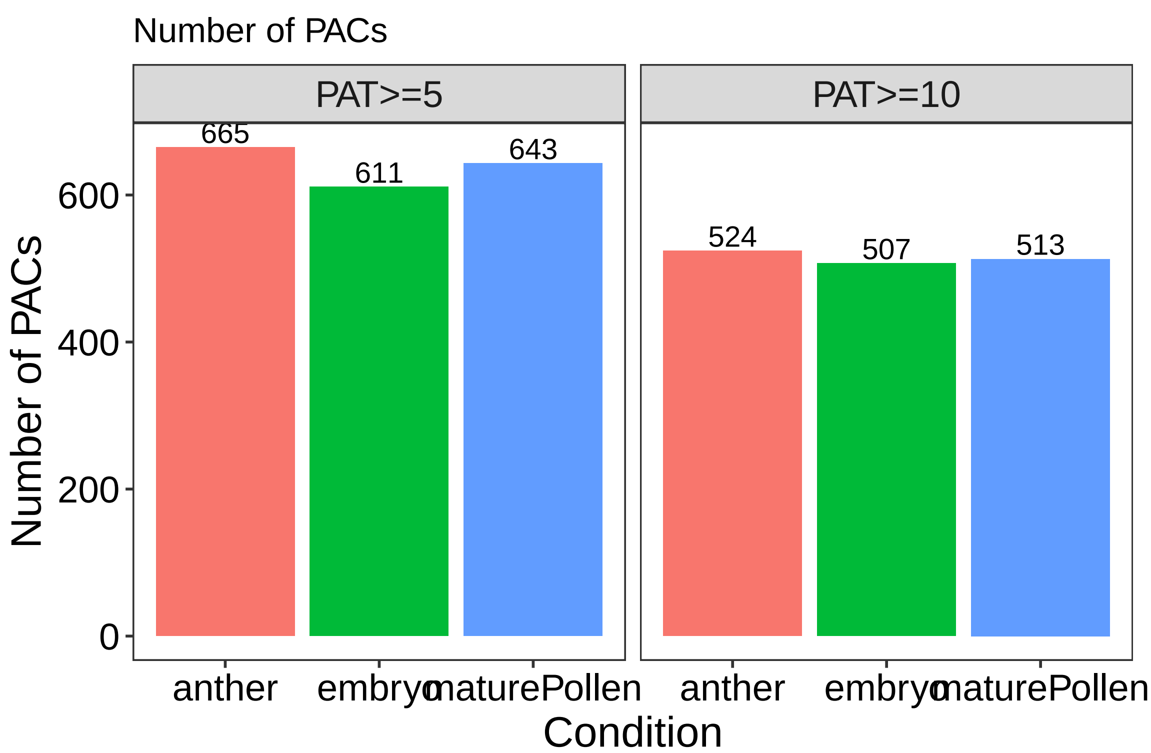
```
plotPACdsStat(pstats, pdfFile='PACds_stat_total.pdf',
              minPAT=c(5,10), conds=c('total'))
```

Plot the overall distributions using pooled samples (total) and one cutoff.

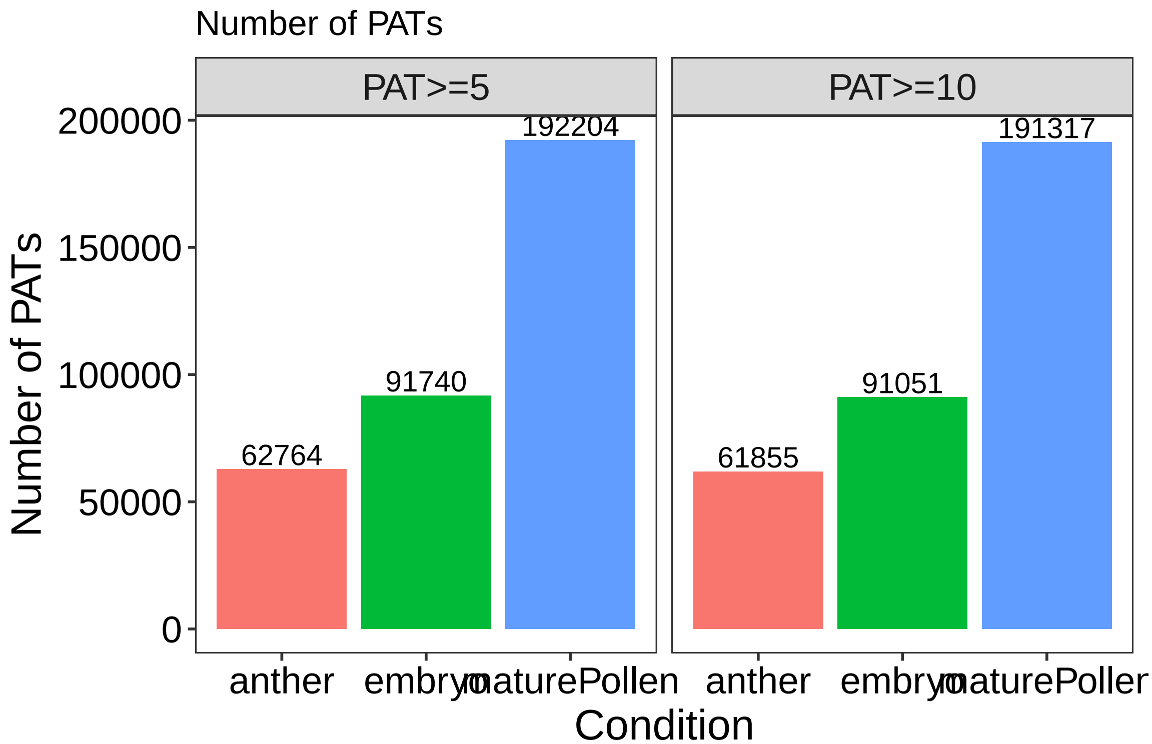
```
plotPACdsStat(pstats, pdfFile='PACds_stat_total_PAT10.pdf',
              minPAT=c(10), conds=c('total'))
```

Plot figures to the current device.

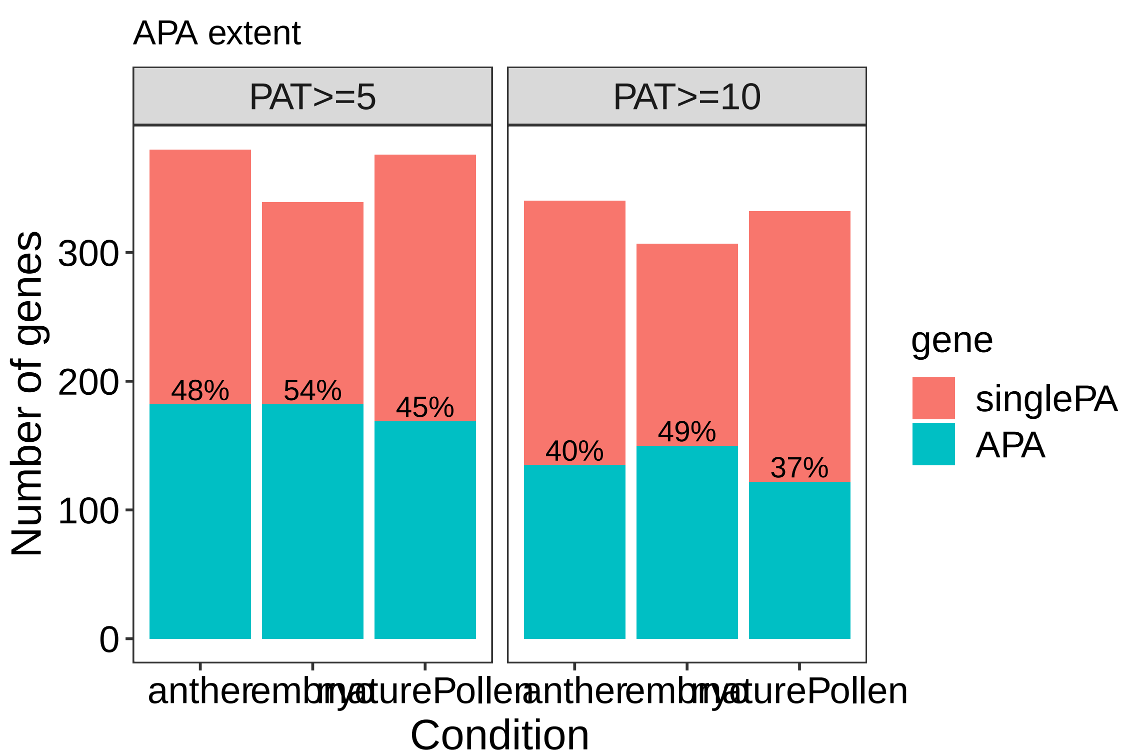
```
plotPACdsStat(pstats, pdfFile=NULL, minPAT=c(5,10))
```



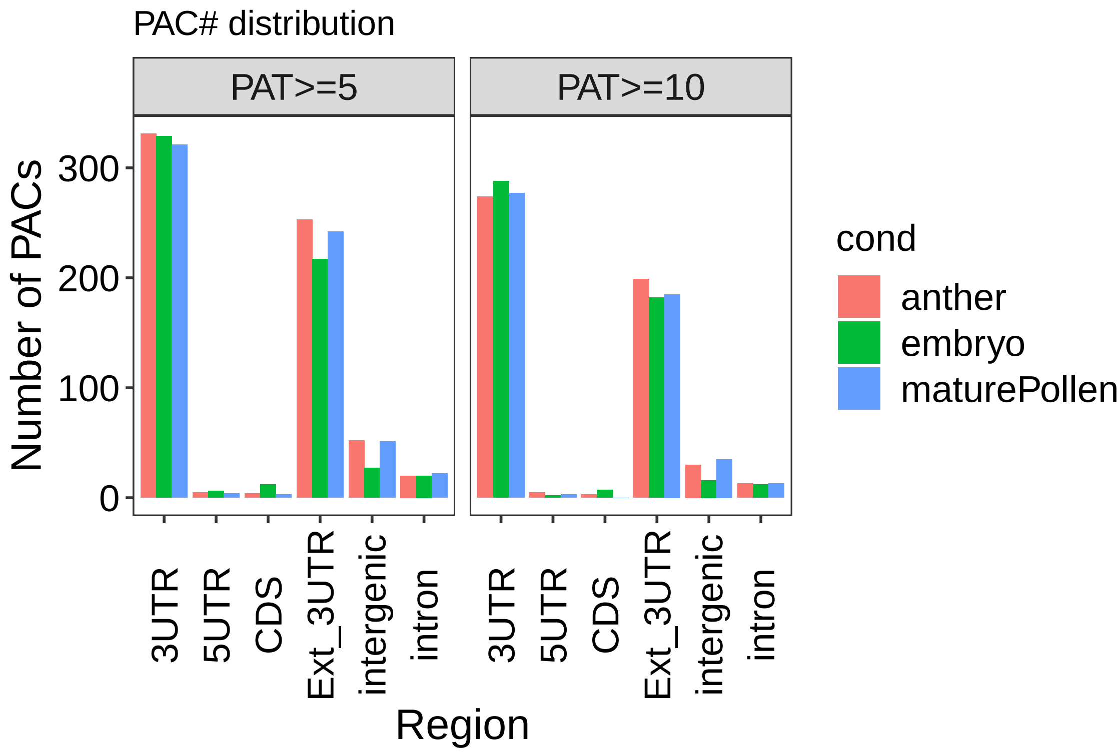

```
#> Plot Number of PACs
```



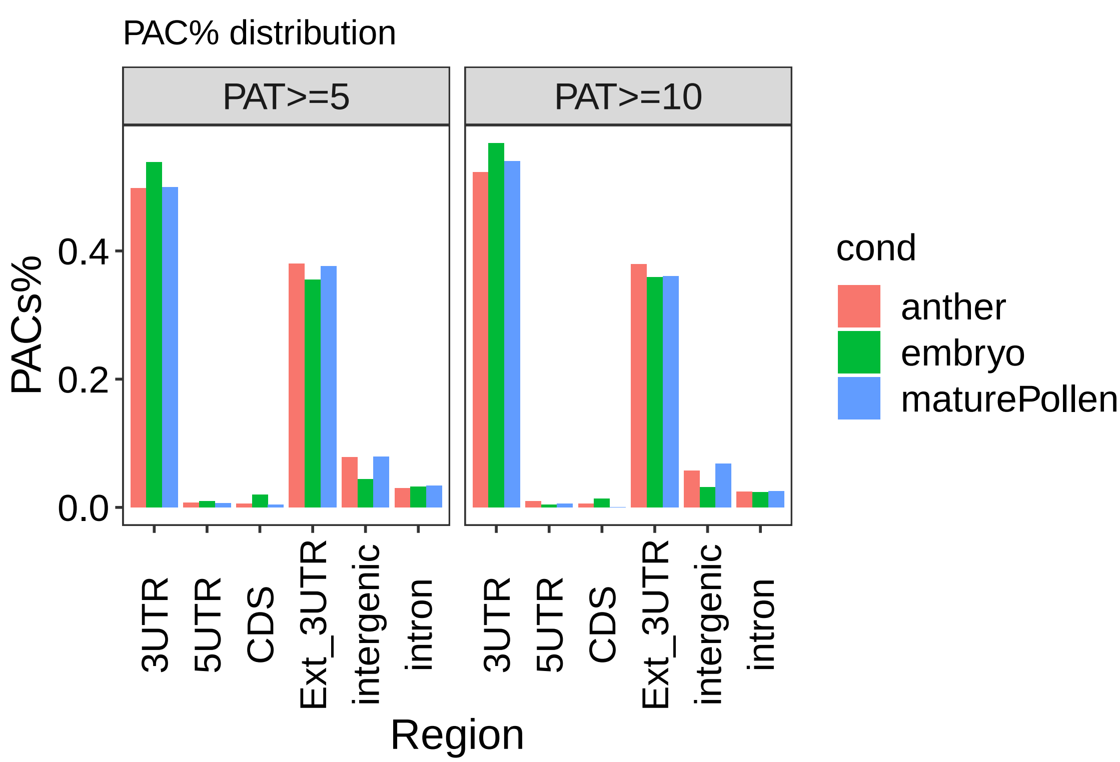

```
#> Plot Number of PATs
```



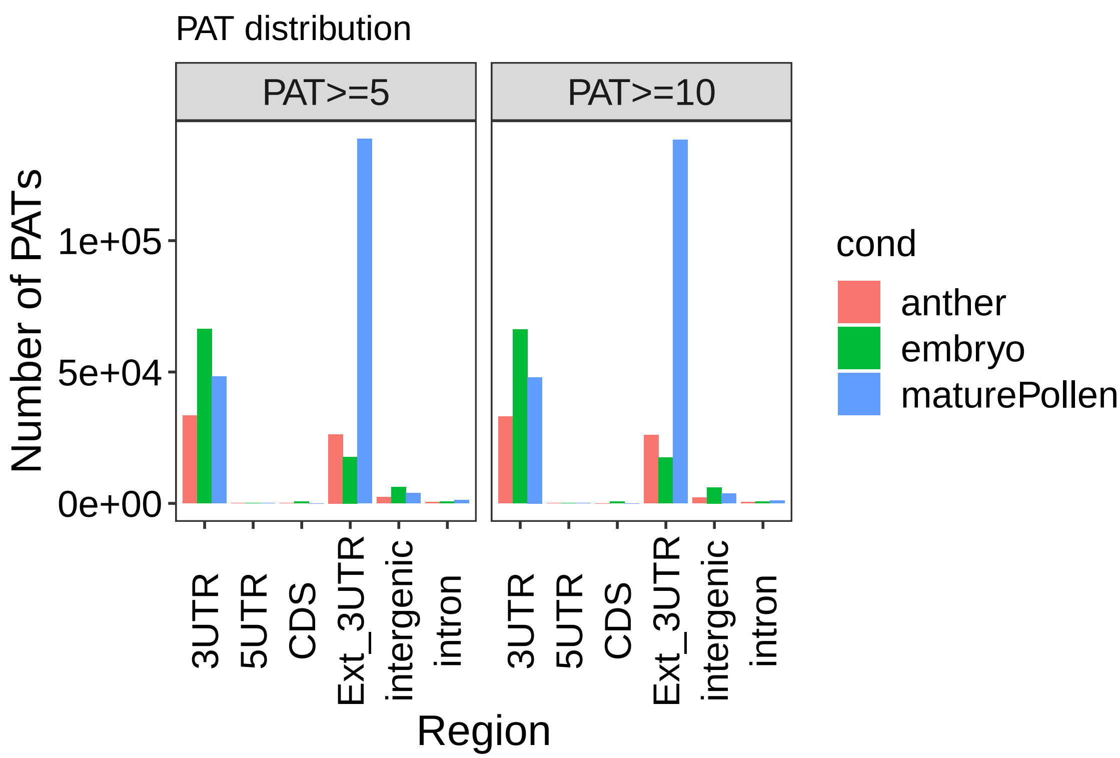

```
#> Plot APA extent
```



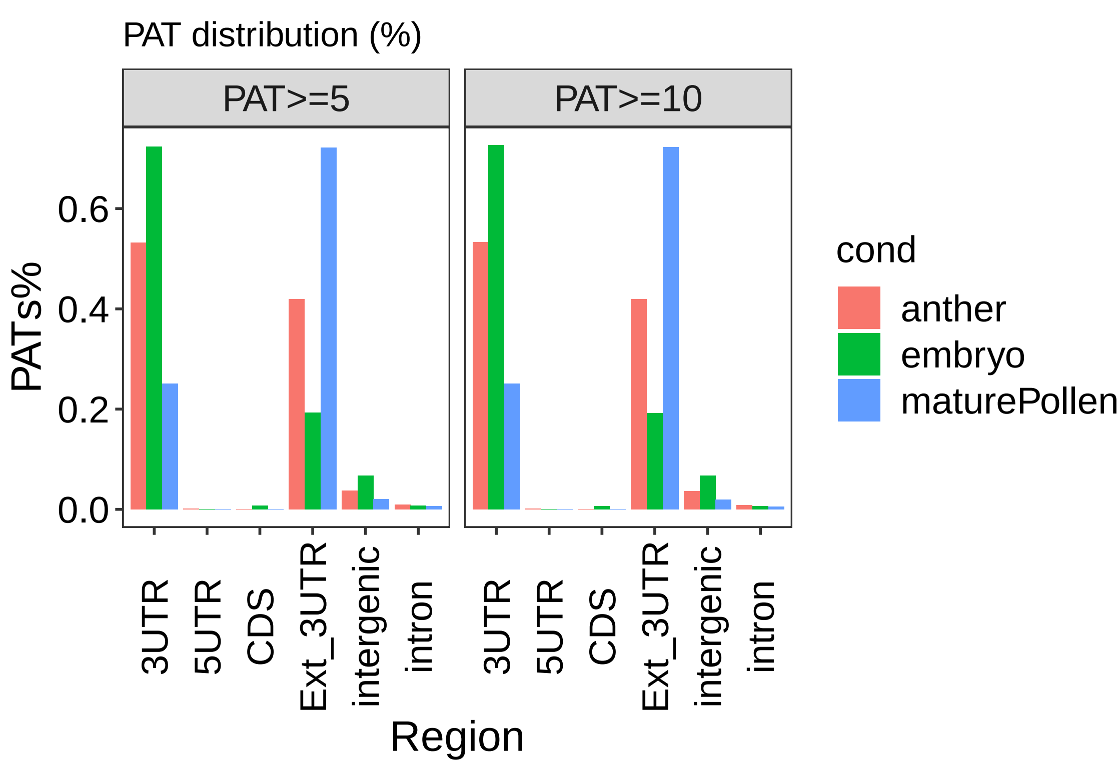
#> Plot PAC# distribution



#> Plot PAC% distribution




```
#> Plot PAT# distribution
```




```
#> Plot PAT% distribution
```

6 Poly(A) signals and sequences

movAPA provides several functions, including *annotateByPAS*, *faFromPACds*, *kcount*, and *plotATCGforFAfile*, for sequence extraction and poly(A) signal identification.

6.1 Poly(A) signals

Annotate PACs by corresponding signal of AATAAA located upstream 50 bp of the PAC.

```
PACdsPAS=annotateByPAS(PACds, bsgenome, grams='AATAAA',
                       from=-50, to=-1, label=NULL)
summary(PACdsPAS@anno$AATAAA_dist)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>   16.00   22.00   25.00   26.92   30.00   50.00    1132
```

Scan AATAAA's 1nt variants.

```
PACdsPAS=annotateByPAS(PACds, bsgenome, grams='V1',
                       from=-50, to=-1, label=NULL)
table(PACdsPAS@anno$V1_gram)
#>
#> AAAAAA AACAAA AAGAAA AATAAA AATAAC AATAAG AATAAT AATACA AATAGA AATATA AATCAA
#>     91     24     50     74     15     31     31     25     26     55     26
#> AATGAA AATTAA ACTAAA AGTAAA ATTAAA CATAAA GATAAA TATAAA
#>     56     21      4     21     27     13     11     36
```

Scan custom k-grams.

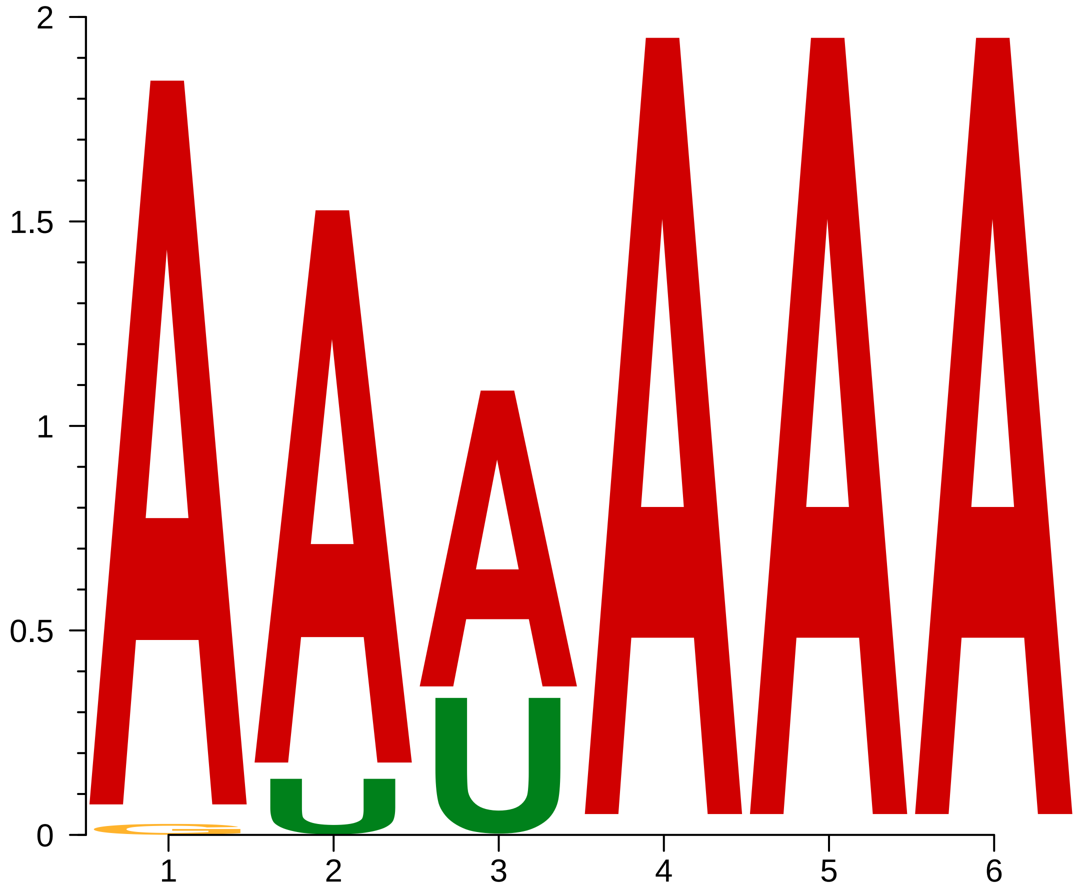
```
PACdsPAS=annotateByPAS(PACds, bsgenome,
                       grams=c('AATAAA','ATTAAA','GATAAA','AAAA'),
                       from=-50, to=-1, label='GRAM')
table(PACdsPAS@anno$GRAM_gram)
#>
#>   AAAA AATAAA ATTAAA GATAAA
#>    409     48     24      8
```

Scan patterns with priority: AATAAA » ATTAAA » remaining k-grams.

```
PACdsPAS=annotateByPAS(PACds, bsgenome,
                       grams=c('AATAAA','ATTAAA','GATAAA','AAAA'),
                       priority=c(1,2,3,3),
                       from=-50, to=-1, label='GRAM')
table(PACdsPAS@anno$GRAM_gram)
#>
#>   AAAA AATAAA ATTAAA GATAAA
#>    337    101     44      7
```

Plot signal logos.

```
pas=PACdsPAS@anno$GRAM_gram[!is.na(PACdsPAS@anno$GRAM_gram)]
plotSeqLogo(pas)
```



Here we show another example to scan mouse signals in rice PACs. First, we get mouse signals and set the priority.

```
v=getVarGrams('mm')
priority=c(1,2,rep(3, length(v)-2))
```

Then scan upstream regions of PACs for mouse signals.

```
PACdsMM=annotateByPAS(PACds, bsgenome, grams=v,
                      priority=priority,
                      from=-50, to=-1, label='mm')
```

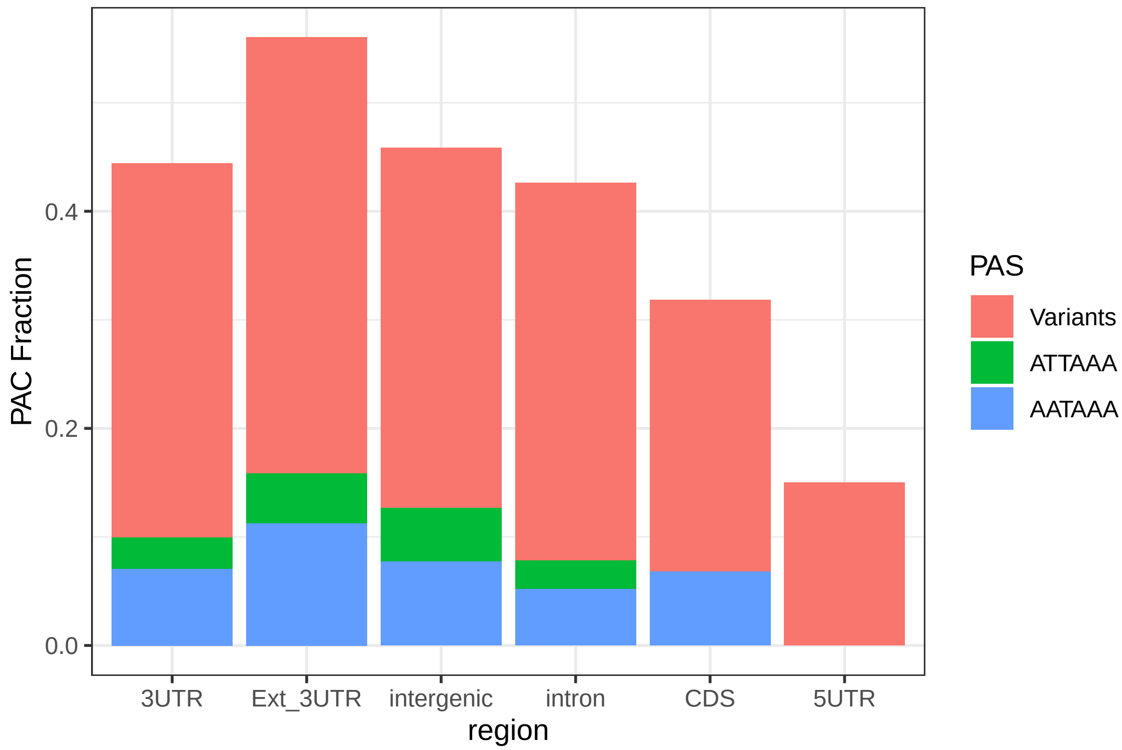
Prepare the data to plot PAS distributions.

```r
library(magrittr)
#>
#> Attaching package: 'magrittr'
#> The following object is masked from 'package:GenomicRanges':
#>
#>     subtract
library(dplyr)
#>
#> Attaching package: 'dplyr'
#> The following objects are masked from 'package:Biostrings':
#>
#>     collapse, intersect, setdiff, setequal, union
#> The following object is masked from 'package:XVector':
#>
#>     slice
#> The following object is masked from 'package:AnnotationDbi':
#>
#>     select
#> The following object is masked from 'package:Biobase':
#>
#>     combine
#> The following objects are masked from 'package:GenomicRanges':
#>
#>     intersect, setdiff, union
#> The following object is masked from 'package:GenomeInfoDb':
#>
#>     intersect
#> The following objects are masked from 'package:IRanges':
#>
#>     collapse, desc, intersect, setdiff, slice, union
#> The following objects are masked from 'package:S4Vectors':
#>
#>     first, intersect, rename, setdiff, setequal, union
#> The following objects are masked from 'package:BiocGenerics':
#>
#>     combine, intersect, setdiff, union
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union
pas=as.data.frame(cbind(region=PACdsMM@anno$ftr, PAS=PACdsMM@anno$mm_gram))
pas$PAS[is.na(pas$PAS)]='NOPAS'
pas$PAS[pas$PAS %in% v[-c(1:2)]]='Variants'
n=pas %>% dplyr::group_by(region, PAS)  %>% dplyr::summarise(nPAC=n())
#> `summarise()` has grouped output by 'region'. You can override using the
#> `.groups` argument.
n2=pas %>% dplyr::group_by(region)  %>% dplyr::summarise(nTot=n())
n=merge(n, n2)
n$PAC=n$nPAC/n$nTot
n=n[n$PAS!='NOPAS', ]
n$PAS=factor(n$PAS, levels=rev(c('AATAAA', 'ATTAAA','Variants', 'NOPAS')))
```

```
n$region=factor(n$region,
                levels=c('3UTR','Ext_3UTR', 'intergenic','intron','CDS','5UTR'))
```

Plot PAS distributions.

```
library(ggplot2)
ggplot(data=n, aes(x=region, y=PAC, fill=PAS)) +
  geom_bar(stat="identity") +
  ylab("PAC Fraction") + theme_bw()
```



6.2 Exract sequences

The *faFromPACds* function provides various options to extract sequences of interest.

```
## Extract the sequence of PACs, from UPA_start to UPA_end.
faFromPACds(PACds, bsgenome, what='pac', fapre='pac')

## Extract upstream 300 bp ~ downstream 100 bp around PACs,
## where the position of PAC is 301.
faFromPACds(PACds, bsgenome, what='updn', fapre='updn',
            up=-300, dn=100)

## Divide PACs into groups of genomic regions and then extract sequences for each group.
faFromPACds(PACds, bsgenome, what='updn', fapre='updn',
            up=-100, dn=100, byGrp='ftr')
```

```r
## Extract sequences for only 3UTR PACs.
faFromPACds(PACds, bsgenome, what='updn', fapre='updn',
            up=-300, dn=100, byGrp=list(ftr='3UTR'))

## Extract sequences for only 3UTR PACs and separate sequences by strand.
faFromPACds(PACds, bsgenome, what='updn', fapre='updn',
            up=-300, dn=100,
            byGrp=list(ftr='3UTR', strand=c('+','-')))

## Extract sequences of genomic regions where PACs are located.
faFromPACds(PACds, bsgenome, what='region', fapre='region', byGrp='ftr')
```

Here we show some examples to extract sequences from different poly(A) signal regions.

```r
## The suggested signal regions when species is 'chlamydomonas_reinhardtii'.
files=faFromPACds(PACds, bsgenome, what='updn', fapre='Chlamy.NUE',
                  up=-25, dn=-5, byGrp='ftr')
files=faFromPACds(PACds, bsgenome, what='updn', fapre='Chlamy.FUE',
                  up=-150, dn=-25, byGrp='ftr')
files=faFromPACds(PACds, bsgenome, what='updn', fapre='Chlamy.CE',
                  up=-5, dn=5, byGrp='ftr')
files=faFromPACds(PACds, bsgenome, what='updn', fapre='Chlamy.DE',
                  up=-5, dn=30, byGrp='ftr')

## The suggested signal regions when species is plant.
## In Arabidopsis or rice, signal regions are: FUE -200~-35, NUE -35~-10, CE -10~15.
files=faFromPACds(PACds, bsgenome, what='updn', fapre='plants.NUE',
                  up=-35, dn=-10, byGrp='ftr')
files=faFromPACds(PACds, bsgenome, what='updn', fapre='plants.FUE',
                  up=-200, dn=-35, byGrp='ftr')
files=faFromPACds(PACds, bsgenome, what='updn', fapre='plants.CE',
                  up=-10, dn=15, byGrp='ftr')
```

6.3 Base compostions and k-grams

The function *plotATCGforFAfile* is for plotting single nucleotide profiles for given fasta file(s), which is particularly useful for discerning base compositions surrounding PACs.

First trim sequences surrounding PACs. Sequences surrounding PACs in different genomic regions are extracted into files. The PAC position is 301.

```r
faFiles=faFromPACds(PACds, bsgenome, what='updn', fapre='updn',
                    up=-300, dn=100, byGrp='ftr')
#> 115 >>> updn.intron.fa
#> 482 >>> updn.3UTR.fa
#> 391 >>> updn.Ext_3UTR.fa
#> 44 >>> updn.CDS.fa
#> 181 >>> updn.intergenic.fa
#> 20 >>> updn.5UTR.fa
```

Then plot base compositions for specific sequence file(s).

```r
faFiles=c("updn.3UTR.fa", "updn.Ext_3UTR.fa", "updn.intergenic.fa", "updn.intron.fa")
## Plot single nucleotide profiles using the extracted sequences and merge all plots into one.
plotATCGforFAfile(faFiles, ofreq=FALSE, opdf=FALSE,
                  refPos=301, mergePlots = TRUE)
```

We can also plot a single fasta file and specify a region.

```r
plotATCGforFAfile (faFiles='updn.intron.fa',
                   ofreq=FALSE, opdf=FALSE, refPos=301)
plotATCGforFAfile (faFiles='updn.intron.fa',
                   ofreq=FALSE, opdf=FALSE, refPos=NULL,
                   filepre ='NUE', start=250, end=350)
```

Users can also generate these plots into a PDF file and save the calculated base compositions.

```r
plotATCGforFAfile (faFiles, ofreq=TRUE, opdf=TRUE, refPos=301,
                   filepre='singleBasePlot', mergePlots = TRUE)
```

After extracting sequences, we can call the *kcount* function to obtain the number of occurrences or frequencies of k-grams from the whole sequences or a specified region of sequences. Particularly, specific k-grams (e.g., AAUAAA, AUUAAA) or a value of k (e.g., k=6 means all hexamers) can be set.

```r
## Count top 10 hexamers (k=6) in the NUE region
## (normally from 265~295 if the PAC is at 301).
fafile='updn.3UTR.fa'
kcount(fafile=fafile, k=6, from=265, to=295, topn=10)
#>        grams count        perc
```

```
#> 1    AAAAAA    74 0.005904883
#> 274  ATATAT    38 0.003032237
#> 3073 GAAAAA    34 0.002713055
#> 2    AAAAAT    31 0.002473667
#> 257  ATAAAA    31 0.002473667
#> 5    AAAATA    30 0.002393872
#> 65   AATAAA    30 0.002393872
#> 1366 TTTTTT    28 0.002234280
#> 449  ATGAAA    27 0.002154485
#> 769  AGAAAA    27 0.002154485

## Count given hexamers.
kcount(fafile=fafile, grams=c('AATAAA','ATTAAA'),
       from=265, to=295, sort=FALSE)
#>    grams count      perc
#> 1 AATAAA    30 0.7142857
#> 2 ATTAAA    12 0.2857143

## Count AATAAA and its 1nt variants in a given region.
kcount(fafile=fafile, grams='v1', from=265, to=295, sort=FALSE)
#>     grams count        perc
#> 1  AATAAA    30 0.092024540
#> 2  TATAAA    14 0.042944785
#> 3  CATAAA     8 0.024539877
#> 4  GATAAA     9 0.027607362
#> 5  ATTAAA    12 0.036809816
#> 6  ACTAAA     3 0.009202454
#> 7  AGTAAA     8 0.024539877
#> 8  AAAAAA    74 0.226993865
#> 9  AACAAA    13 0.039877301
#> 10 AAGAAA    21 0.064417178
#> 11 AATTAA    14 0.042944785
#> 12 AATCAA    11 0.033742331
#> 13 AATGAA    19 0.058282209
#> 14 AATATA    26 0.079754601
#> 15 AATACA     9 0.027607362
#> 16 AATAGA     8 0.024539877
#> 17 AATAAT    23 0.070552147
#> 18 AATAAC     7 0.021472393
#> 19 AATAAG    17 0.052147239
```

7 Quantification of PACs by various metrics

movAPA provides various metrics to measure the usages of PACs across samples, including three metrics for the quantification of the usage of each single poly(A) site by the *movPAindex* function and four metrics for the quantification of APA site usage of a gene by the *movAPAindex* function.

7.1 Quantification of each PAC by *movPAindex*

movPAindex provides three metrics for the quantification of each PAC in a gene, including "ratio", "Shannon", and "geo". First you can merge replicates of the same sample and remove lowly expressed PACs before calculate the index.

```
p=subsetPACds(PACds, group='group', pool=TRUE, totPACtag=20)
```

Calculate the tissue-specificity. Q or H=0 means that the PAC is only expressed in one tissue. NA means the PAC is not expressed in the respective tissue.

```
paShan=movPAindex(p, method='shan')
#> Using count for Shannon.
#> Tissue-specific PAC's H_cutoff (mean-2*sd):  0.275623
#> Tissue-specific PAC's Q_cutoff (mean-2*sd):  0.2052354
#> Tissue-specific PAC# (H<H_cutoff):  35
#> Tissue-specific PAC# (Q<Q_cutoff):  25
#> Constitutive PAC's H_cutoff (mean+2*sd):  1.957418
#> Constitutive PAC's Q_cutoff (mean+2*sd):  3.42726
#> Constitutive PACs (H>H_cutoff):  0
#> Constitutive PACs (Q>Q_cutoff):  0
## Show some rows with low H value (which means high overall tissue-specificity).
head(paShan[paShan$H<0.2742785, ], n=2)
#>                                  H    Q_min Q_min_cond   anther   embryo
#> Os01g0266100:9088974  0.2006223 0.246426     embryo 5.200622 0.246426
#> Os01g0571300:21939527 0.0000000 0.000000     embryo       NA 0.000000
#>                       maturePollen
#> Os01g0266100:9088974            NA
#> Os01g0571300:21939527           NA
```

Use the relative expression levels (ratio) to calculate tissue-specificity.

```
paShan2=movPAindex(p, method='shan', shan.ratio = TRUE)
#> Using ratio for Shannon.
#> Tissue-specific PAC's H_cutoff (mean-2*sd):  0.6462506
#> Tissue-specific PAC's Q_cutoff (mean-2*sd):  0.9463281
#> Tissue-specific PAC# (H<H_cutoff):  20
#> Tissue-specific PAC# (Q<Q_cutoff):  24
#> Constitutive PAC's H_cutoff (mean+2*sd):  2.053762
#> Constitutive PAC's Q_cutoff (mean+2*sd):  3.81471
#> Constitutive PACs (H>H_cutoff):  0
#> Constitutive PACs (Q>Q_cutoff):  0
head(paShan2, n=2)
#>                                    H     Q_min Q_min_cond   anther    embryo
#> ENSRNA049472915:32398829 0.7060639 0.9176406     embryo 4.950709 0.9176406
#> ENSRNA049472915:32398407 1.5828394 3.1107952     anther 3.110795 3.2821041
#>                          maturePollen
#> ENSRNA049472915:32398829     4.285457
#> ENSRNA049472915:32398407     3.116965
```

Cacluate the geo metric, which is only suitable for APA genes. NA means no PAC of the gene is expressed in the respective tissue. geo>0 means the PAC is used more than average usage of all PACs in the gene. geo~0 means similar usage; <0 means less usage.

```
paGeo=movPAindex(p, method='geo')
head(paGeo, n=2)
```

```
#>                               anther   embryo maturePollen
#> ENSRNA049472915:32398829 -3.454947 -1.44546    -3.084963
#> ENSRNA049472915:32398407  3.454947  1.44546     3.084963
```

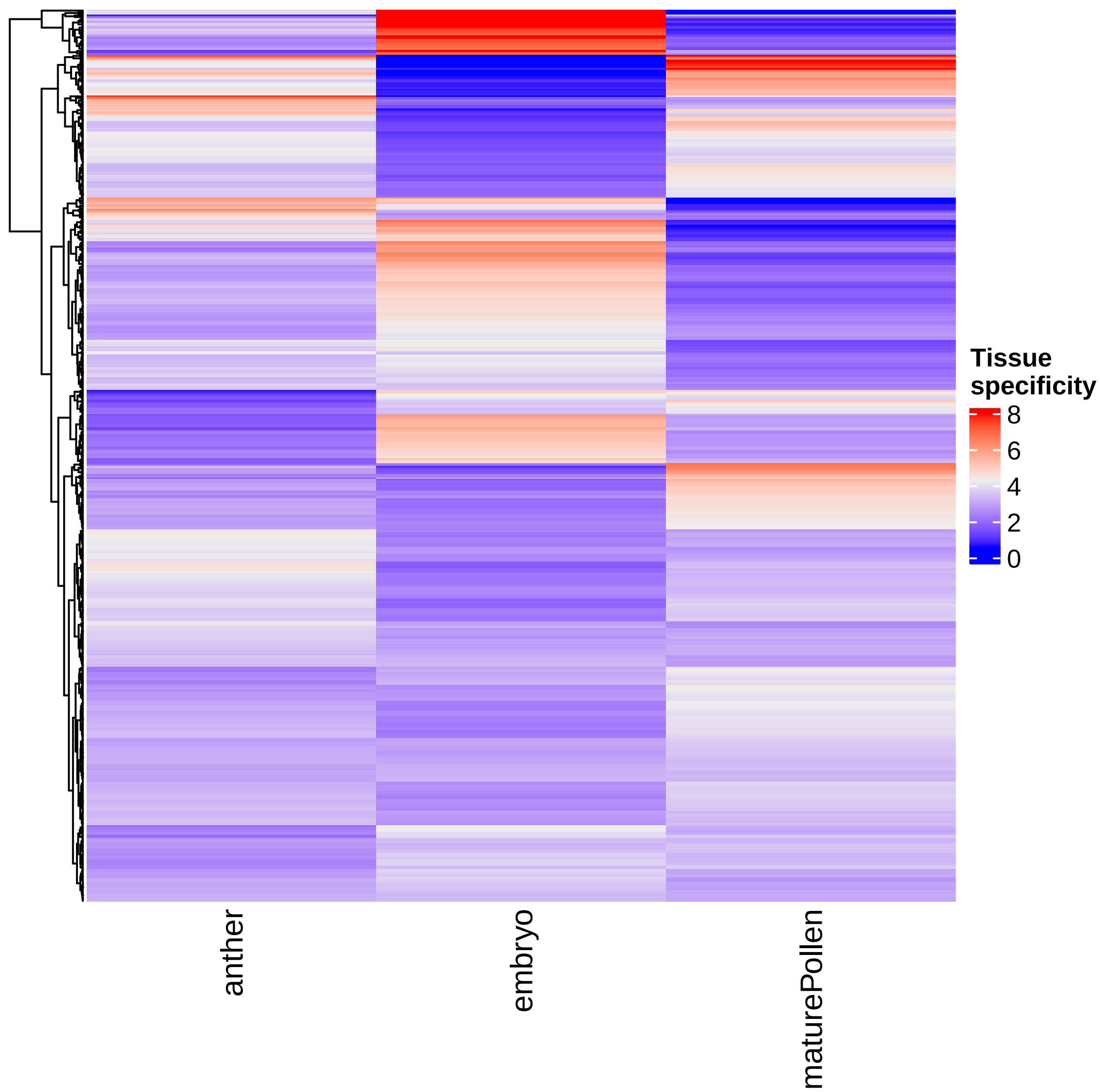
Cacluate the ratio metric, which is only suitable for APA genes. NA means no PAC of the gene is expressed in the respective tissue.

```
paRatio=movPAindex(p, method='ratio')
head(paRatio)
#>                                anther    embryo maturePollen
#> ENSRNA049472915:32398829 0.007231405 0.1183852   0.01146789
#> ENSRNA049472915:32398407 0.992768595 0.8816148   0.98853211
#> Os01g0151600:2795487     0.326732673 0.5248869   0.82278481
#> Os01g0151600:2795636     0.504950495 0.2714932   0.13924051
#> Os01g0151600:2795858     0.168316832 0.2036199   0.03797468
#> Os01g0179300:4126216     0.927272727 0.8699634   0.99152542
```

Plot a heatmap to show the distribution of tissue-specificity of PACs. It is only reasonable to plot the heatmap of the Shanno metric. Or you may filter the proximal or distal PAC of the gene first and plot the ratio or geo metrics.

First, remove rows with NA and then plot the heatmap.

```
paShanHm=paShan[, -(1:3)]
paShanHm=paShanHm[rowSums(is.na(paShanHm))==0, ]
library(ComplexHeatmap, quietly = TRUE)
Heatmap(paShanHm, show_row_names=FALSE,  cluster_columns = FALSE,
        heatmap_legend_param = list(title = 'Tissue\nspecificity'))
```

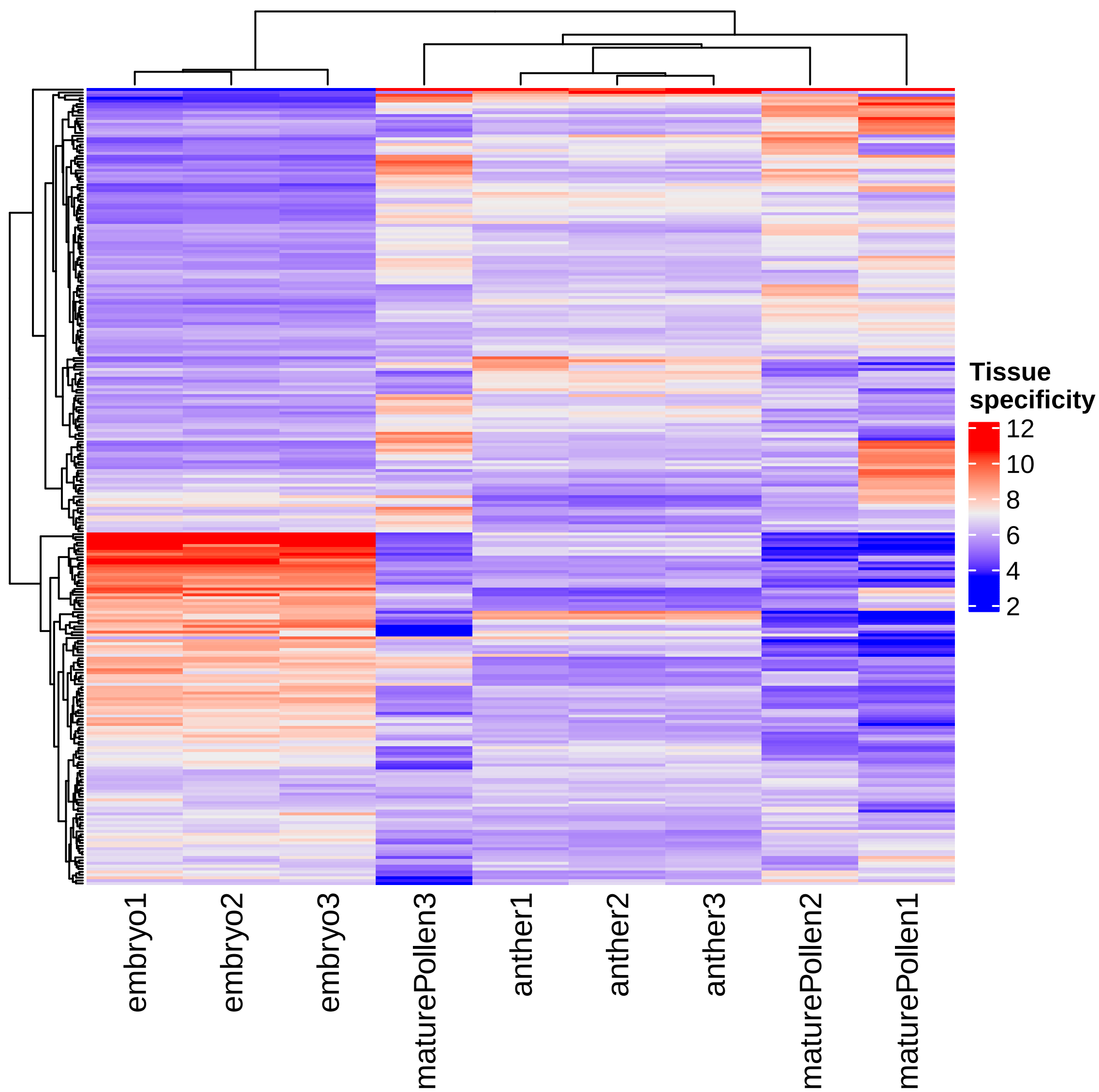


Calculate the tissue-specificity for each replicate.

```
paShan=movPAindex(PACds, method='shan')
#> Using count for Shannon.
#> Tissue-specific PAC's H_cutoff (mean-2*sd):  0.269235
#> Tissue-specific PAC's Q_cutoff (mean-2*sd):  0.3825375
#> Tissue-specific PAC# (H<H_cutoff):  91
#> Tissue-specific PAC# (Q<Q_cutoff):  91
#> Constitutive PAC's H_cutoff (mean+2*sd):  3.69439
#> Constitutive PAC's Q_cutoff (mean+2*sd):  6.527702
#> Constitutive PACs (H>H_cutoff):  0
#> Constitutive PACs (Q>Q_cutoff):  0
```

```
## Plot heamap to show the consistency among replicates.
paShanHm=paShan[, -(1:3)]
paShanHm=paShanHm[rowSums(is.na(paShanHm))==0, ]
Heatmap(paShanHm, show_row_names=FALSE,  cluster_columns = TRUE,
```

```
        heatmap_legend_param = list(title = 'Tissue\nspecificity'))
```



data## Quantification of APA by *movAPAindex* The *movAPAindex* function provides four gene-level metrics for the quantification of APA site usage, including RUD (Relative Usage of Distal PAC) (Ji, et al., 2009), WUL (Weighted 3' UTR Length) (Ulitsky, et al., 2012; Fu, et al., 2016), SLR (Short to Long Ratio) (Begik, et al., 2017), and GPI (Geometric Proximal Index) (Shulman and Elkon, 2019).

Get APA index using the smart RUD method (available in movAPA v2.0).

```
pd=get3UTRAPApd(pacds=p, minDist=50, maxDist=1000, minRatio=0.05, fixDistal=FALSE, addCols='pd')
rud=movAPAindex(pd, method="smartRUD", sRUD.oweight=TRUE)
head(rud$rud)
head(rud$weight)
geneRUD=rud$rud
geneRUD=geneRUD[rowSums(is.na(geneRUD))==0, ]
head(geneRUD, n=2)
Heatmap(geneRUD, show_row_names=FALSE,  cluster_columns = F,
```

```
        heatmap_legend_param = list(title = 'RUD'))
```

Get APA index using the WUL method.

```
geneWUL=movAPAindex(p, method="WUL", choose2PA=NULL)
head(geneWUL, n=2)
#>                 anther   embryo maturePollen
#> Os01g0151600 231.4643 191.7955     162.5658
#> Os01g0254900 265.4275 286.5083     233.5099
```

Plot gene's metric values across samples by heatmap with the ComplexHeatmap package.

```
## Remove NA rows before plotting heatmap.
geneWUL=geneWUL[rowSums(is.na(geneWUL))==0, ]
Heatmap(geneWUL, show_row_names=FALSE)
```

Get APA index using the RUD method.

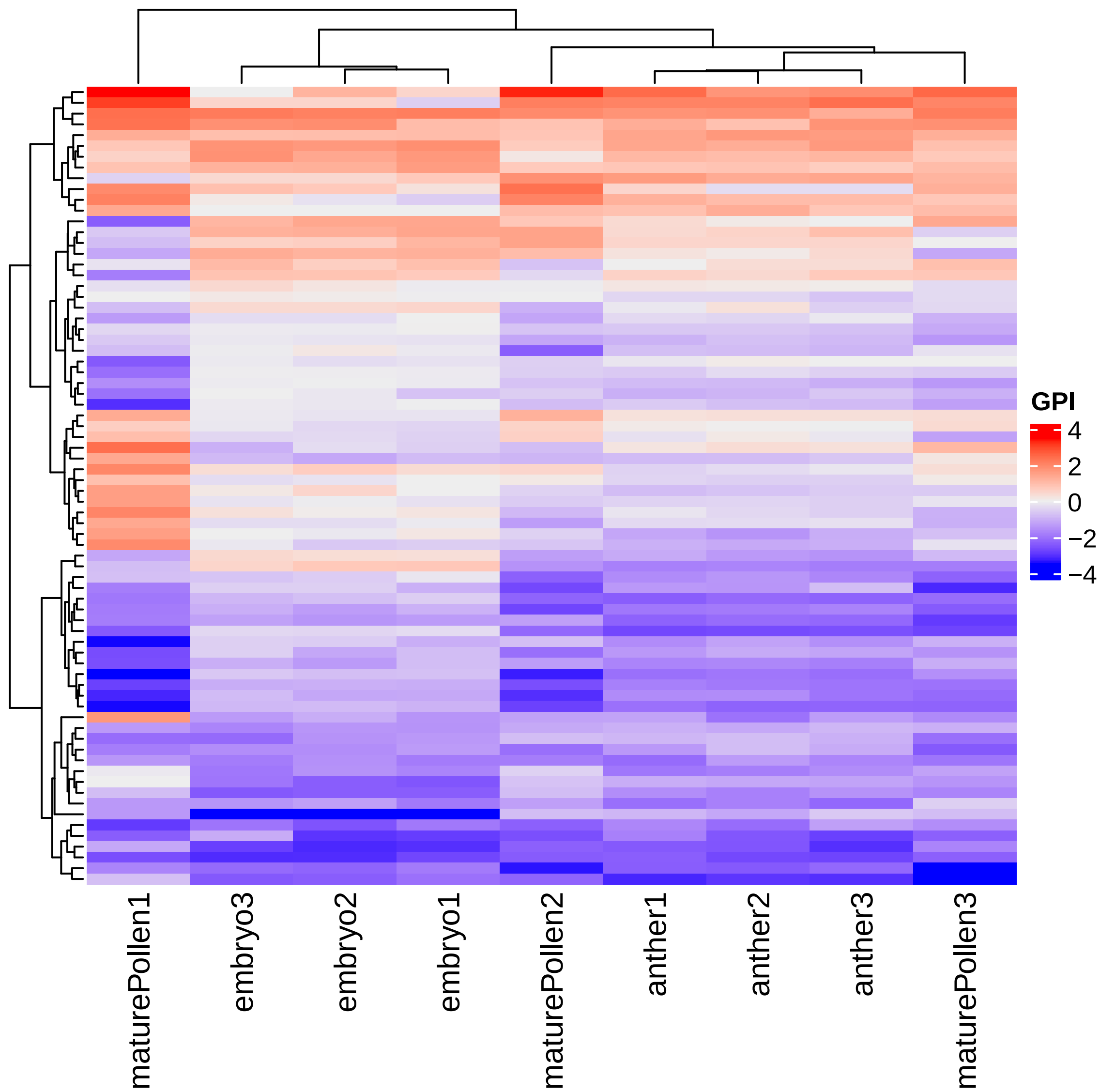
```
geneRUD=movAPAindex(p, method="RUD",
                    choose2PA=NULL, RUD.includeNon3UTR=TRUE)
geneRUD=geneRUD[rowSums(is.na(geneRUD))==0, ]
head(geneRUD, n=2)
Heatmap(geneRUD, show_row_names=FALSE,  cluster_columns = F,
        heatmap_legend_param = list(title = 'RUD'))
```

Get APA index by method=SLR, using the proximal and distal PACs.

```
geneSLR=movAPAindex(p, method="SLR", choose2PA='PD')
head(geneSLR, n=2)
geneSLR=geneSLR[rowSums(is.na(geneSLR))==0, ]
Heatmap(geneSLR, show_row_names=FALSE)
```

Get APA index by method=GPI, using the proximal and distal PACs.

```
geneGPI=movAPAindex(PACds, method="GPI", choose2PA='PD')
head(geneGPI)
#>                 anther1    anther2    anther3    embryo1    embryo2
#> Os01g0151600 -0.4587689 -0.3107442 -0.1400540  0.3912043  0.6609640
#> Os01g0254900  0.2721603  0.3175016  0.3064884 -0.1752486 -0.1705185
#> Os01g0261200 -1.0253130 -0.9437626 -0.6497801 -0.7075187 -0.1315172
#> Os01g0263600 -0.5000000        NaN -0.5000000 -1.2924813 -0.2924813
#> Os01g0314000  1.1609640  0.7924813  1.0000000  0.5000000  1.0000000
#> Os01g0524500 -0.3140156 -0.5000000 -0.2237295        NaN        NaN
#>                    embryo3 maturePollen1 maturePollen2 maturePollen3
#> Os01g0151600  3.552467e-01      2.014874     0.5000000    0.35021986
#> Os01g0254900 -9.965440e-02      1.347822     1.2221596    0.35975231
#> Os01g0261200 -3.203427e-16     -2.043731    -0.5351947   -1.00000000
#> Os01g0263600 -1.292481e+00           NaN           NaN           NaN
#> Os01g0314000  2.924813e-01           NaN           NaN           NaN
#> Os01g0524500  0.000000e+00      1.100817     0.2311716    0.03700029
geneGPI=geneGPI[rowSums(is.na(geneGPI))==0, ]
Heatmap(geneGPI, show_row_names=FALSE,  cluster_columns = TRUE,
        heatmap_legend_param = list(title = 'GPI'))
```



8 DE genes

3' seq data have been demonstrated informative in quantifying expression levels of genes by summing up 3' seq reads of all PACs in a gene (Lianoglou, et al., 2013). To detect DE genes between samples with 3' seq, we implemented the function *movDEgene* with the widely used R package DESeq2.

Note: DE detection should be performed in caution, because different methods would have significant and different impact on the DE results!

8.1 Detect DE genes

First we show an example of detecting DE genes for two conditions.

```r
library(DESeq2)
## Subset two conditions first.
pacds=subsetPACds(PACds, group='group', cond1='anther', cond2='embryo')
## Detect DE genes using DESeq2 method,
## only genes with total read counts in all samples >=50 are used.
DEgene=movDEGene(PACds=pacds, method='DESeq2', group='group', minSumPAT=50)
```

Make statistics of the DE gene results; genes with padj<0.05 & log2FC>=0.5 are considered as DE genes.

```r
stat=movStat(object=DEgene, padjThd=0.05, valueThd=0.5)
stat$nsig
#>               sig.num
#> anther.embryo     219
head(stat$siglist$anther.embryo)
#> [1] "ENSRNA049472915" "Os01g0151600"    "Os01g0179300"    "Os01g0210600"
#> [5] "Os01g0247600"    "Os01g0254900"
```

We can also detect DE genes among more than two conditions.

```r
DEgene=movDEGene(PACds=PACds, method='DESeq2', group='group', minSumPAT=50)
stat=movStat(object=DEgene, padjThd=0.05, valueThd=1)
```

```r
## Number of DE genes in each pair of conditions.
stat$nsig
#>                     sig.num
#> anther.embryo           150
#> anther.maturePollen      77
#> embryo.maturePollen     192
## Overlap between condition pairs.
stat$ovp
#>                                      pair n1.sig.num n2.sig.num novp.sig.num
#> 1       anther.embryo-anther.maturePollen        150         77           47
#> 2       anther.embryo-embryo.maturePollen        150        192          122
#> 3 anther.maturePollen-embryo.maturePollen         77        192           62
```

8.2 Output DE genes

Output *movStat* results into files: "DEgene.plots.pdf" and 'DEgene.stat'. Several heatmaps are generated.

```r
outputHeatStat(heatStats=stat, ostatfile='DEgene.stat', plotPre='DEgene')
```

You can further call movSelect() to select DE gene results with more information. Select DE gene results with full information including the read counts in each sample.

```r
selFull=movSelect(DEgene, condpair='embryo.anther', padjThd=0.05, valueThd=1,
                  out='full', PACds=PACds)
#> Warning: condpair is flip of movRes@conds, so movRes@pairwise$value*(-1)
head(selFull)
#>           gene anther1 anther2 anther3 embryo1 embryo2 embryo3 maturePollen1
#> 1 Os01g0179300      38      42      39     195     195     170            58
```

```
#> 2 Os01g0210600      59      65      52     376     359     295            22
#> 3 Os01g0224200       1       1       0      15      13      16             0
#> 4 Os01g0238500       8      11      12       0       0       2            62
#> 5 Os01g0247600     106     132     123      20      15      12             6
#> 6 Os01g0524500      26      25      24       0       0       2           138
#>   maturePollen2 maturePollen3         padj     value
#> 1            31            29 3.594718e-10  2.234465
#> 2            60            82 5.726760e-08  2.548997
#> 3             5             1 4.048010e-03  4.459420
#> 4            17           111 6.343326e-03 -3.954189
#> 5            63             7 1.080655e-03 -2.941261
#> 6           105            76 1.104847e-05 -5.228812
```

Select DE gene results with only padj and value. Here value is log2(anther/embryo).

```
sel=movSelect(DEgene, condpair='anther.embryo',
              padjThd=0.05, valueThd=1, out='pv')
head(sel)
#>                     padj     value
#> Os01g0179300 3.594718e-10 -2.234465
#> Os01g0210600 5.726760e-08 -2.548997
#> Os01g0224200 4.048010e-03 -4.459420
#> Os01g0238500 6.343326e-03  3.954189
#> Os01g0247600 1.080655e-03  2.941261
#> Os01g0524500 1.104847e-05  5.228812
```

Output gene names of DE genes.

```
sel=movSelect(DEgene, condpair='embryo.anther',
              padjThd=0.05, upThd=0.5, out='gene')
#> Warning: condpair is flip of movRes@conds, so movRes@pairwise$value*(-1)
head(sel)
#> [1] "ENSRNA049472915" "Os01g0179300"    "Os01g0210600"    "Os01g0224200"
#> [5] "Os01g0571300"    "Os01g0586600"
```

9 DE PACs

movAPA provides the function *movDEPAC* to identify DE PACs between samples. Three strategies were utilized: (i) using DESeq2 with replicates; (ii) using DEXseq with replicates; (iii) using chi-squared test without replicates ("chisq").

9.1 Detect DE PACs

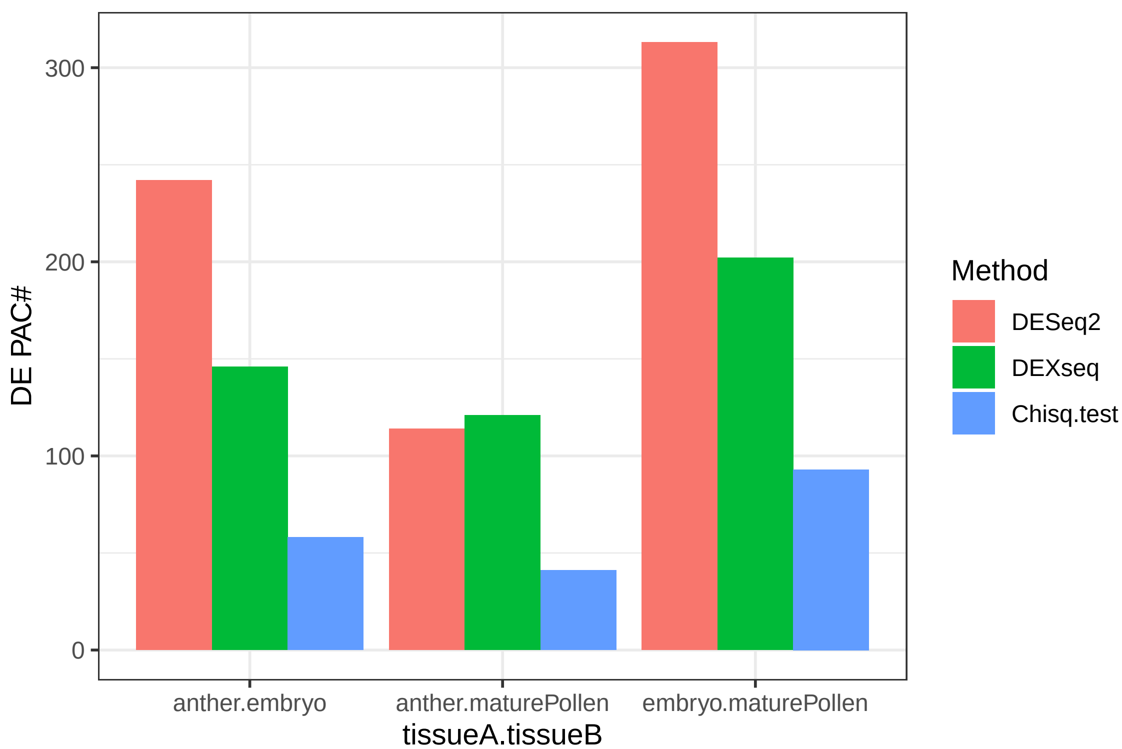
First we show an example of detecting DE PACs among all pairwise conditions using three different methods. Only PACs with total read counts in all samples >=20 are used.

```
DEPAC=movDEPAC(PACds, method='DESeq2', group='group', minSumPAT=20)
DEXPAC=movDEPAC(PACds, method='DEXseq', group='group', minSumPAT=20)
DEqPAC=movDEPAC(PACds, method='chisq', group='group', minSumPAT=20)
```

Number of DE PACs among methods.

```
library(ggplot2)
## Get significant DE results.
stat1=movStat(object=DEPAC, padjThd=0.05, valueThd=1)
stat2=movStat(object=DEXPAC, padjThd=0.05, valueThd=1)
stat3=movStat(object=DEqPAC, padjThd=0.05, valueThd=0.95)
```

```
## Count the number of DE PACs by different methods.
nsig=as.data.frame(cbind(stat1$nsig, stat2$nsig, stat3$nsig))
colnames(nsig)=c('DESeq2','DEXseq','Chisq.test')
nsig$tissueA.tissueB=rownames(nsig)
nsig
#>                      DESeq2 DEXseq Chisq.test      tissueA.tissueB
#> anther.embryo           242    146         58        anther.embryo
#> anther.maturePollen     114    121         41 anther.maturePollen
#> embryo.maturePollen     313    202         93 embryo.maturePollen
## Plot a barplot.
nsig=reshape2::melt(nsig, variable.name='Method')
#> Using tissueA.tissueB as id variables
ggplot(data=nsig, aes(x=tissueA.tissueB, y=value, fill=Method)) +
  geom_bar(stat="identity", position=position_dodge()) +
  ylab("DE PAC#") + theme_bw()
```



We can also detect DE PACs between two given conditions.

```
## First subset PACs in two conditions.
PACds1=subsetPACds(PACds, group='group',
                   cond1='anther', cond2='embryo', choosePA='apa')
```

```r
## Detect DE PACs.
DEPAC1=movDEPAC(PACds1, method='DESeq2', group='group', minSumPAT=10)
DEXPAC1=movDEPAC(PACds1, method='DEXseq', group='group', minSumPAT=10)
DEqPAC1=movDEPAC(PACds1, method='chisq', group='group', minSumPAT=10)
```

9.2 Statistics of DE PACs

Make statistics of the DE PACs result by DESeq2 method (*DEPAC*).

```r
stat=movStat(object=DEPAC, padjThd=0.05, valueThd=1)
```

```r
## Number of DE PACs between conditions.
stat$nsig
#>                      sig.num
#> anther.embryo            242
#> anther.maturePollen      114
#> embryo.maturePollen      313
## Overlap of DE PACs between different pairs of conditions.
head(stat$ovp)
#>                                       pair n1.sig.num n2.sig.num novp.sig.num
#> 1        anther.embryo-anther.maturePollen        242        114           68
#> 2        anther.embryo-embryo.maturePollen        242        313          199
#> 3 anther.maturePollen-embryo.maturePollen        114        313           90
## DE PAC list
head(stat$siglist[[1]])
#> [1] "Os01g0151600:2795487" "Os01g0179300:4126216" "Os01g0179300:4126779"
#> [4] "Os01g0238500:7668102" "Os01g0247600:8130944" "Os01g0247600:8131074"
```

We can also plot a venn diagram to show the overlap among results from different pairwise comparisons.

```r
library(VennDiagram, quietly = TRUE)
x=venn.diagram(stat$siglist, fill=brewer.pal(3, "Set1"), cex=2,
               cat.fontface=4,  filename='DEPAC.venn')
```

Stat the DE PAC result from the chisq-test method, here the value column of DEqPAC is 1-pvalue_of_the_gene. So using padjThd=0.05 and valueThd=0.95 means filtering DE PACs with adjusted pvalue of PAC <0.05 and adjusted pvalue of gene <0.05.

```r
stat=movStat(object=DEqPAC, padjThd=0.05, valueThd=0.95)
```

9.3 Output DE PACs

We can use *movSelect* to output full or simple list of DE PACs.

```r
## Here method is DEXseq, so the valueThd (log2FC) threshold is automatelly determined.
sel=movSelect(aMovRes=DEXPAC, condpair='embryo.anther',
              padjThd=0.1, out='full', PACds=PACds)
#> Warning: condpair is flip of movRes@conds, so movRes@pairwise$value*(-1)
#> Warning: movRes is DEXPAC, but valueThd/upThd/dnThd are all NULL, mannually set valueThd=min(maxfc)=
#> Warning: movRes is DEXPAC, also filter by rowMax(movRes@pairwise$value)
```

```
head(sel, n=2)
#>                          PA chr UPA_start  UPA_end strand    coord        ftr
#> 1 ENSRNA049444301:25040070  12  25040068 25040071      - 25040070 intergenic
#> 2 ENSRNA049472915:32398407   3  32398154 32398573      - 32398407   Ext_3UTR
#>              gene gene_type ftr_start  ftr_end anther1 anther2 anther3 embryo1
#> 1 ENSRNA049444301      tRNA  25043356 25032325       0       0       0       0
#> 2 ENSRNA049472915    snoRNA  32398830 32398830     285     324     352     510
#>   embryo2 embryo3 maturePollen1 maturePollen2 maturePollen3         padj
#> 1       0       0            24             3             0 6.971614e-02
#> 2     558     548           130           191           110 1.081613e-13
#>        value
#> 1 -0.7671602
#> 2  0.8623713

## You can also mannually set a log2FC threshold.
sel=movSelect(aMovRes=DEXPAC, condpair='embryo.anther',
              padjThd=0.1, valueThd=2, out='pa');
#> Warning: condpair is flip of movRes@conds, so movRes@pairwise$value*(-1)
#> Warning: movRes is DEXPAC, also filter by rowMax(movRes@pairwise$value)
head(sel)
#> [1] "Os01g0263600:8948833"  "Os01g0327400:12630082" "Os01g0327400:12630218"
#> [4] "Os01g0812200:34534443" "Os01g0841000:36096864" "Os01g0881300:38257576"

## Filter only up-regulated PACs in embryo
## (value=log2(embryo_this_others/anther_this_others)).
sel=movSelect(aMovRes=DEXPAC, condpair='embryo.anther',
              padjThd=0.1, upThd=2, out='full', PACds=PACds)
#> Warning: condpair is flip of movRes@conds, so movRes@pairwise$value*(-1)
#> Warning: movRes is DEXPAC, also filter by rowMax(movRes@pairwise$value)
head(sel, 2)
#>                          PA chr UPA_start  UPA_end strand    coord        ftr
#> 1 ENSRNA049472915:32398407   3  32398154 32398573      - 32398407   Ext_3UTR
#> 2    Os01g0327400:12630082   1  12629955 12630117      + 12630082 intergenic
#>              gene      gene_type ftr_start  ftr_end anther1 anther2 anther3
#> 1 ENSRNA049472915         snoRNA  32398830 32398830     285     324     352
#> 2    Os01g0327400 protein_coding  12582256 12646586       1       1       1
#>   embryo1 embryo2 embryo3 maturePollen1 maturePollen2 maturePollen3
#> 1     510     558     548           130           191           110
#> 2       0       0       0             0             0             0
#>           padj      value
#> 1 1.081613e-13  0.8623713
#> 2 3.171817e-02 11.5717831
```

9.4 Visualize DE PACs in a gene

Here we take the DEPAC result for example to show the visualization of DE PACs in a gene.

```
## Filter less results and plot the heatmap clearly.
stat=movStat(object=DEPAC, padjThd=0.001, valueThd=8)
outputHeatStat(heatStats=stat, ostatfile='DEPAC.stat', plotPre='DEPAC',
               show_rownames = TRUE)
```

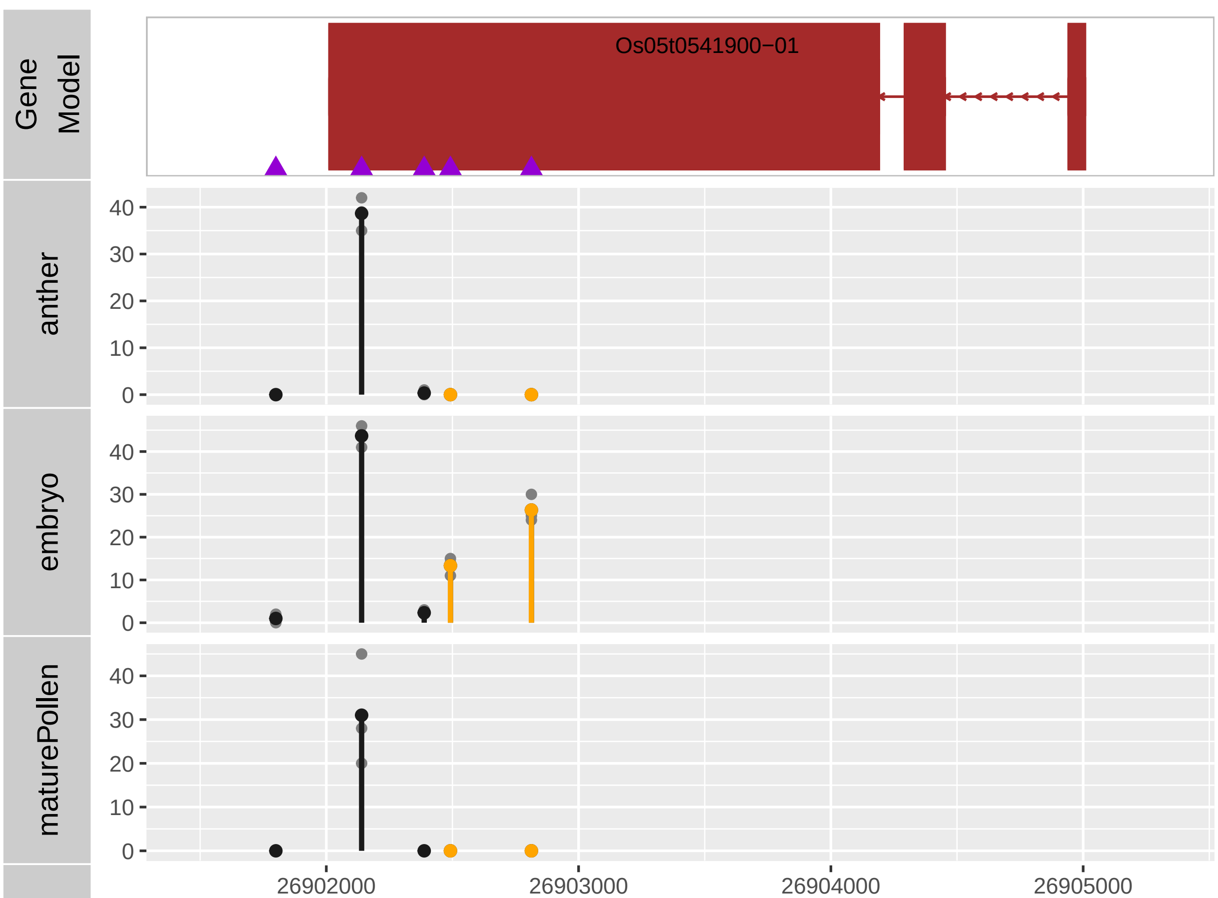
Visualize DE PACs in an example gene by *movViz*. First, we examine all PACs in this gene. There are three

PACs, two in 3'UTR and one in extended 3'UTR. But the expression level of the PAC in extended 3UTR is only 3.

```
gene='Os05g0541900'
gp=PACds[PACds@anno$gene==gene, ]
cbind(gp@anno$ftr, rowSums(gp@counts))
#>                         [,1]         [,2]
#> Os05g0541900:26902813 "3UTR"        "79"
#> Os05g0541900:26902492 "3UTR"        "40"
#> Os05g0541900:26902388 "3UTR"        "8"
#> Os05g0541900:26902140 "3UTR"        "340"
#> Os05g0541900:26901800 "Ext_3UTR"    "3"
#> Os05g0541900:26900274 "intergenic" "2"
```

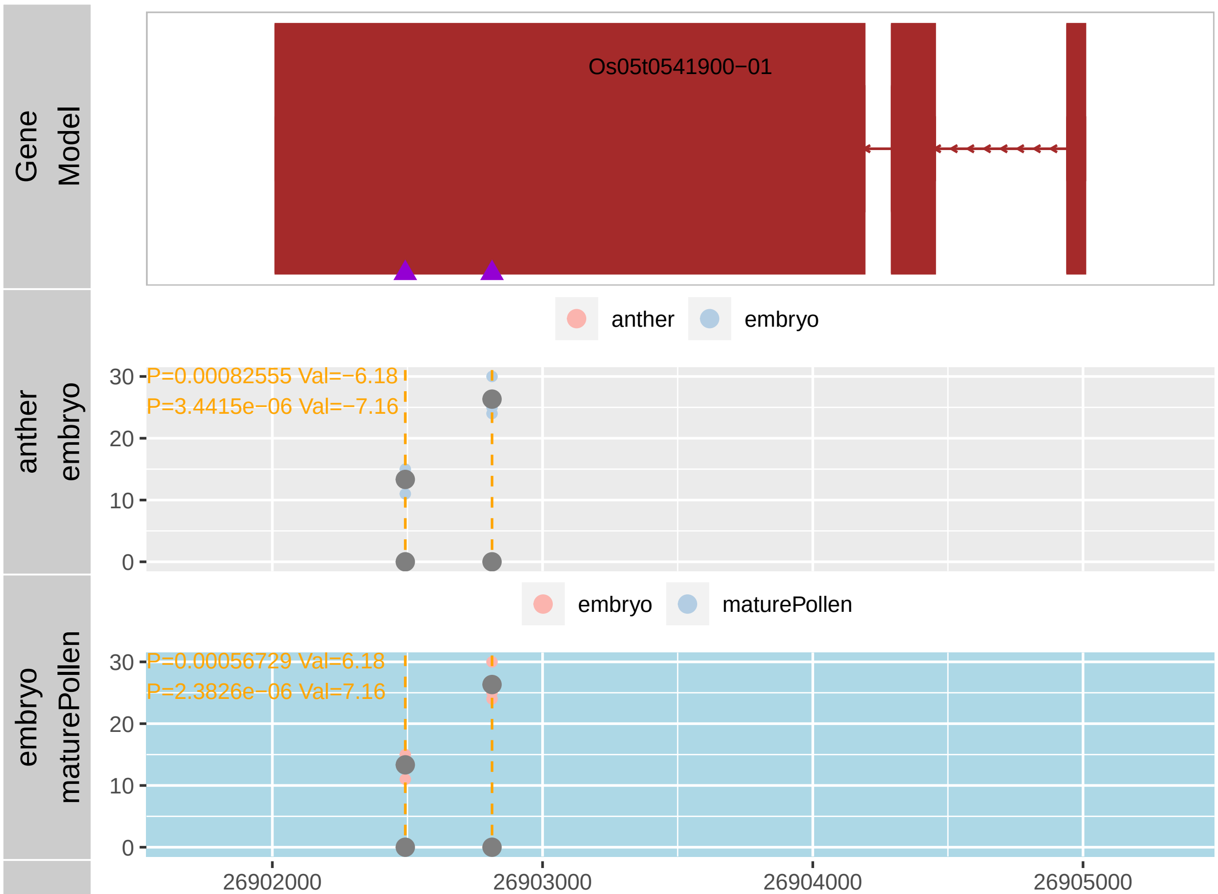
Visualize PACs of this gene in individual conditions. Here the Y-axis is read count, the scale of which is different among conditions.DE PACs identified by DESeq2 method with padj < padjThd are highlighted in dashed yellow lines.

```
movViz(object=DEPAC, gene=gene, txdb=gff, PACds=PACds, collapseConds=FALSE,
       padjThd=0.01, showRatio=FALSE, showAllPA=TRUE)
```



We can also show condition pairs in individual tracks and only display and/or highlight given condition pairs. If padjThd is given, then the DE PACs (padj < padjThd) will be highlighted (dashed yellow line).

```
movViz(object=DEPAC, gene=gene, txdb=gff, PACds=PACds, collapseConds=TRUE,
       padjThd=0.01, showPV=TRUE, showAllPA=FALSE, showRatio=F,
       conds=DEPAC@conds[c(1,3), ], highlightConds=DEPAC@conds[c(3), ])
```



10 3’UTR switching

APA dynamics (i.e., APA site switching or 3’UTR lengthening/shortening) of a gene can be deduced by comparing the ratios of expression levels of one poly(A) site (e.g., the short isoform) over the other poly(A) site (e.g., the long isoform) between two biological samples. For unity, here we refer 3’UTR lengthening/shortening to 3’UTR switching, and refer APA dynamics involving a pair of PACs to APA site switching. Function *movUTRtrend* is used to identify 3’UTR switching events between samples. We developed three methods in movUTRtrend for detecting 3’UTR switching events from samples with or without replicates: (i) the strategy based on the chi-squared test for trend in proportions (“linearTrend”); (ii) the strategy based on DE PACs from DESeq2 (“DE”); (iii) the strategy based on DE PACs from DEXSeq (“DEX”).

10.1 Detect 3’UTR switching events

First, we used the ‘linearTrend’ method to detect 3’UTR switching events. Only PACs and genes with average read count between the two conditions >=10 and >=20 are used.

```r
utr=movUTRtrend(PACds, group='group', method='linearTrend',
                avgPACtag=10, avgGeneTag=20)
#> anther.embryo
#> anther.maturePollen
#> embryo.maturePollen
## Number of genes for analyzing, including those not significant.
lapply(utr@fullList, nrow)
#> $anther.embryo
#> [1] 44
#>
#> $anther.maturePollen
#> [1] 31
#>
#> $embryo.maturePollen
#> [1] 47
head(utr@fullList[["anther.embryo"]], n=2)
#>                      gene nPAC geneTag1 geneTag2 avgUTRlen1 avgUTRlen2
#> Os01g0151600 Os01g0151600    2       28       59   231.4643   191.5085
#> Os01g0254900 Os01g0254900    2      221      101   265.5520   286.9604
#>                   pvalue      padj change        cor  logRatio
#> Os01g0151600 0.018098744 0.4886661     -1 -0.2534036 -1.072588
#> Os01g0254900 0.008878056 0.2840978      1  0.1458238  1.128916
#>                                                             PAs1
#> Os01g0151600  Os01g0151600:2795487=11;Os01g0151600:2795636=17
#> Os01g0254900 Os01g0254900:8475658=133;Os01g0254900:8475521=88
#>                                                            PAs2
#> Os01g0151600 Os01g0151600:2795487=39;Os01g0151600:2795636=20
#> Os01g0254900 Os01g0254900:8475658=45;Os01g0254900:8475521=56
```

Make statistics of the results; genes with padj<0.1 and abs(cor)>0 are considered as 3'UTR switching.

```r
stat=movStat(object=utr, padjThd=0.1, valueThd=0)
#> All cond pairs in heat@colData, get de01 and deNum
stat$nsig
#>                     sig.num
#> anther.embryo             9
#> anther.maturePollen       4
#> embryo.maturePollen      24
```

Output 3'UTR switching results for a pair of conditions.

```r
## Only output gene ids.
out=movSelect(aMovRes=utr, condpair='anther.embryo',
              padjThd=0.1, valueThd=0, out='gene')
## Output PAC ids.
out=movSelect(aMovRes=utr, condpair='anther.maturePollen',
              padjThd=0.1, valueThd=0, out='pa')
## Output gene ids with padj and value.
out=movSelect(aMovRes=utr, condpair='anther.embryo',
              padjThd=0.1, valueThd=0, out='pv')
## Output full information with expression levels, 3UTR length,
## read counts of each PA in each sample, etc.
out=movSelect(aMovRes=utr, condpair='anther.embryo',
```

```
              padjThd=0.1, valueThd=0, out='full')
## Output full information for 3UTR lengthening genes from anther to embryo (change=1).
out=movSelect(aMovRes=utr, condpair='anther.embryo',
              padjThd=0.1, upThd=0, out='full')
```

```
## Output full information for 3UTR shortening genes from anther to embryo (change=-1).
out=movSelect(aMovRes=utr, condpair='anther.embryo',
              padjThd=0.1, dnThd=0, out='full')
head(out, n=2)
#>                      gene nPAC geneTag1 geneTag2 avgUTRlen1 avgUTRlen2
#> Os02g0759700 Os02g0759700    2       77       54   539.7662   323.3704
#> Os05g0438800 Os05g0438800    2       93      197   348.7204   294.7005
#>                    pvalue         padj change        cor   logRatio
#> Os02g0759700 2.499868e-09 1.049944e-07     -1 -0.5208556  0.5111023
#> Os05g0438800 6.160952e-06 2.464381e-04     -1 -0.2654699 -1.0820747
#>                                                              PAs1
#> Os02g0759700 Os02g0759700:31988970=10;Os02g0759700:31989403=67
#> Os05g0438800  Os05g0438800:21501003=4;Os05g0438800:21500764=89
#>                                                               PAs2
#> Os02g0759700  Os02g0759700:31988970=34;Os02g0759700:31989403=20
#> Os05g0438800 Os05g0438800:21501003=53;Os05g0438800:21500764=144
```

Here is another example of using DEX method to detect 3'UTR switching events. First get DE PAC results by DEXseq and then get 3'UTR switching events.

```
DEXPAC=movDEPAC(PACds, method='DEXseq', group='group', minSumPAT=10)
swDEX=movUTRtrend(PACds, group='group', method='DEX',
                  avgPACtag=10, avgGeneTag=20,
                  aMovDEPACRes=DEXPAC, DEPAC.padjThd=0.01,
                  mindist=50, fisherThd=0.01, logFCThd=1, selectOne='farest')
```

Get 3'UTR switching genes with padj<0.1 and |log2FC|>=1.

```
stat=movStat(object=swDEX, padjThd=0.01, valueThd=1)
#> All cond pairs in heat@colData, get de01 and deNum
stat$nsig
#>                      sig.num
#> anther.embryo              6
#> anther.maturePollen        1
#> embryo.maturePollen       15

out=movSelect(aMovRes=swDEX, condpair='anther.embryo',
              padjThd=0.01, valueThd=1, out='full')
head(out, n=2)
#>           gene nPAC geneTag1 geneTag2 avgUTRlen1 avgUTRlen2     fisherPV
#> 1 Os02g0759700    2       77       54   539.7662   323.3704 4.912714e-09
#> 2 Os05g0438800    2       93      197   348.7204   294.7005 3.852680e-06
#>       logFC change                   PA1                    PA2 dist nDEPA
#> 1 -3.364997     -1 Os02g0759700:31988970 Os02g0759700:31989403  434     2
#> 2 -2.744903     -1 Os05g0438800:21501003 Os05g0438800:21500764  240     1
#>   nSwitchPair                                              PAs1
#> 1           1 Os02g0759700:31988970=10;Os02g0759700:31989403=67
```

```
#> 2           1  Os05g0438800:21501003=4;Os05g0438800:21500764=89
#>                                                             PAs2
#> 1  Os02g0759700:31988970=34;Os02g0759700:31989403=20
#> 2 Os05g0438800:21501003=53;Os05g0438800:21500764=144
```

10.2 Statistics of 3'UTR switching results

Here we used three methods to call 3'UTR switching and then compared the results from these methods.

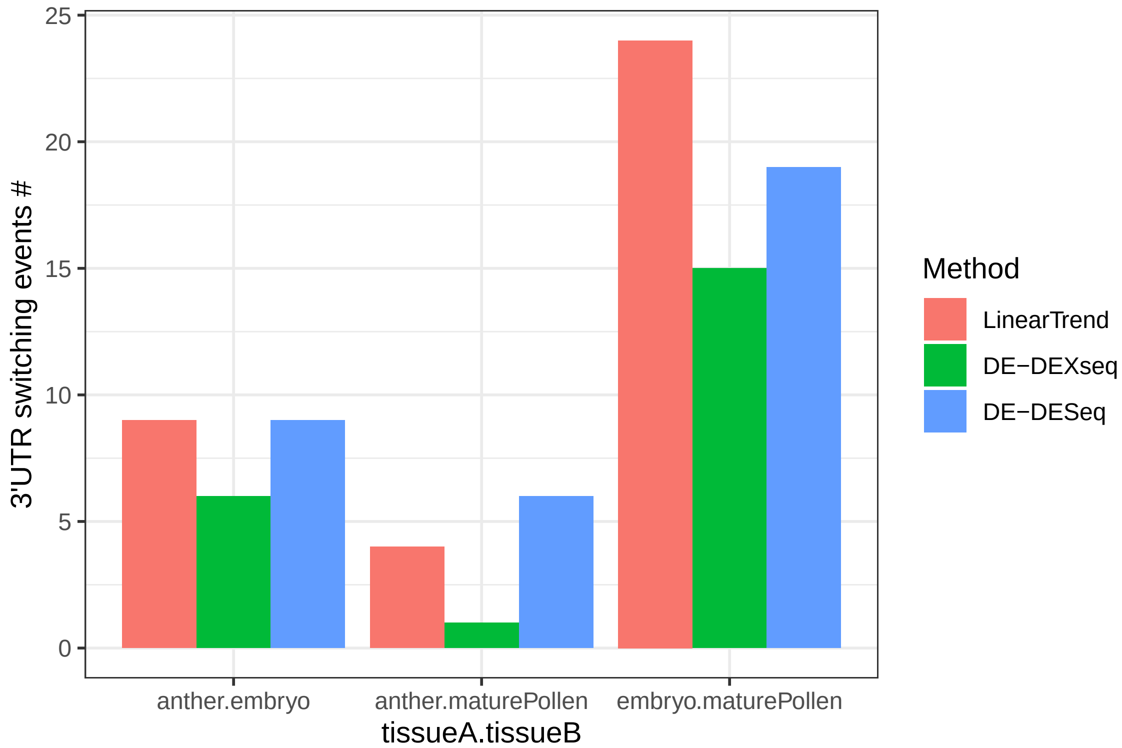
```
swLinear=movUTRtrend(PACds, group='group',method='linearTrend',
                     avgPACtag=10, avgGeneTag=20)
swDEX=movUTRtrend(PACds, group='group', method='DEX',
                 avgPACtag=10, avgGeneTag=20,
                 aMovDEPACRes=DEXPAC, DEPAC.padjThd=0.01,
                 mindist=50, fisherThd=0.01, logFCThd=1, selectOne='fisherPV')

swDE=movUTRtrend(PACds, group='group', method='DE',
                  avgPACtag=10, avgGeneTag=20,
                  aMovDEPACRes=DEPAC, DEPAC.padjThd=0.01,
                  mindist=50, fisherThd=0.01, logFCThd=1, selectOne='fisherPV')
```

Get significant 3'UTR switching events.

```
stat1=movStat(object=swLinear, padjThd=0.1, valueThd=0)
stat2=movStat(object=swDEX, padjThd=0.01, valueThd=1)
stat3=movStat(object=swDE, padjThd=0.01, valueThd=1)
```

Count number of 3'UTR switching events by different methods

```
nsig=as.data.frame(cbind(stat1$nsig, stat2$nsig, stat3$nsig))
colnames(nsig)=c('LinearTrend','DE-DEXseq','DE-DESeq')
nsig$tissueA.tissueB=rownames(nsig)
nsig
#>                     LinearTrend DE-DEXseq DE-DESeq      tissueA.tissueB
#> anther.embryo                 9         6        9        anther.embryo
#> anther.maturePollen           4         1        6 anther.maturePollen
#> embryo.maturePollen          24        15       19 embryo.maturePollen
nsig=reshape2::melt(nsig, variable.name='Method')
#> Using tissueA.tissueB as id variables
ggplot(data=nsig, aes(x=tissueA.tissueB, y=value, fill=Method)) +
  geom_bar(stat="identity", position=position_dodge()) +
  ylab("3\'UTR switching events #") + theme_bw()
```



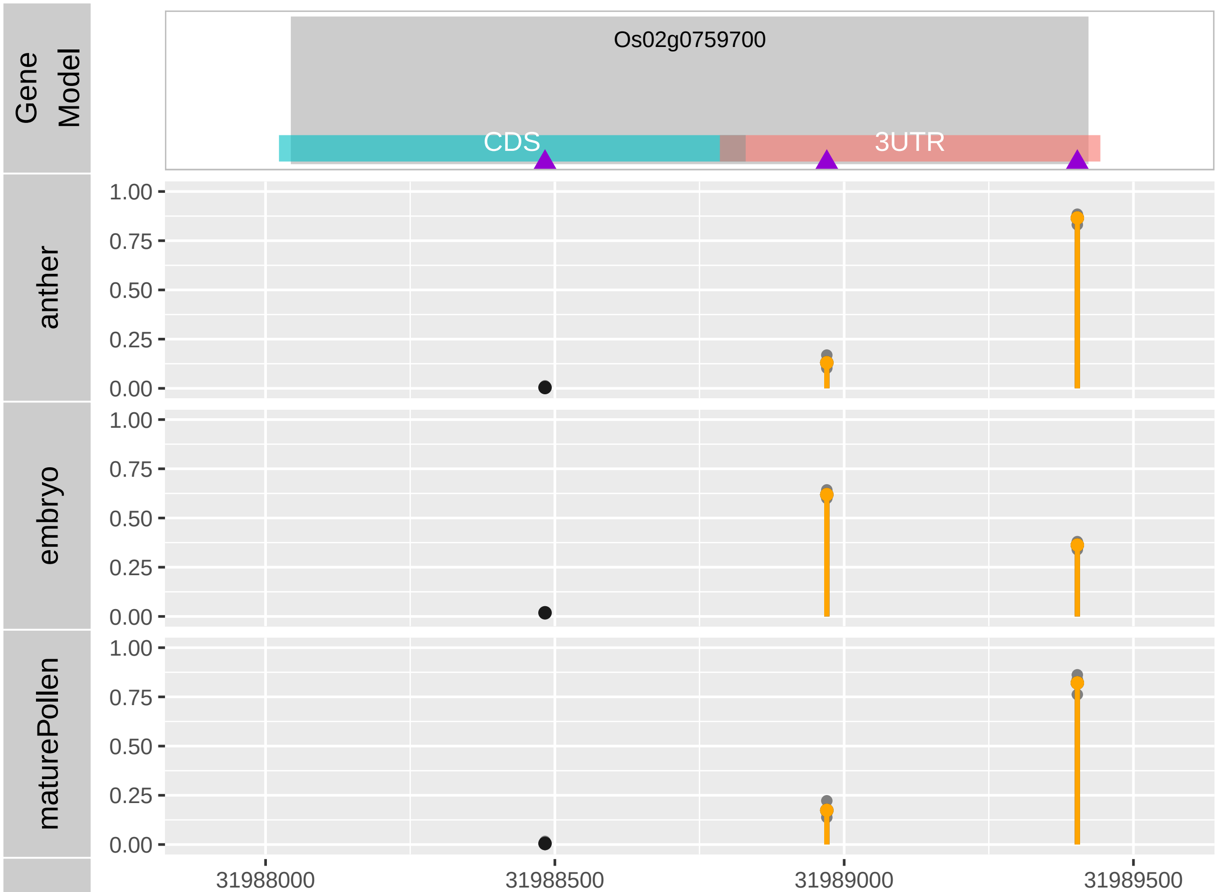
10.3 Visualize 3'UTR switching events

Gene Os02g0759700 is identified as 3'UTR switching. This gene has one PAC in CDS and two PACs in 3UTR; the 3UTR switching happens between anther~embryo and between embryo~maturePollen.

```
gene='Os02g0759700'
gp=PACds[PACds@anno$gene==gene, ]
cbind(gp@anno$ftr, rowSums(gp@counts))
#>                         [,1]         [,2]
#> Os02g0759700:31988483 "CDS"        "5"
#> Os02g0759700:31988970 "3UTR"       "204"
#> Os02g0759700:31989403 "3UTR"       "649"
#> Os02g0759700:31990233 "intergenic" "4"
```

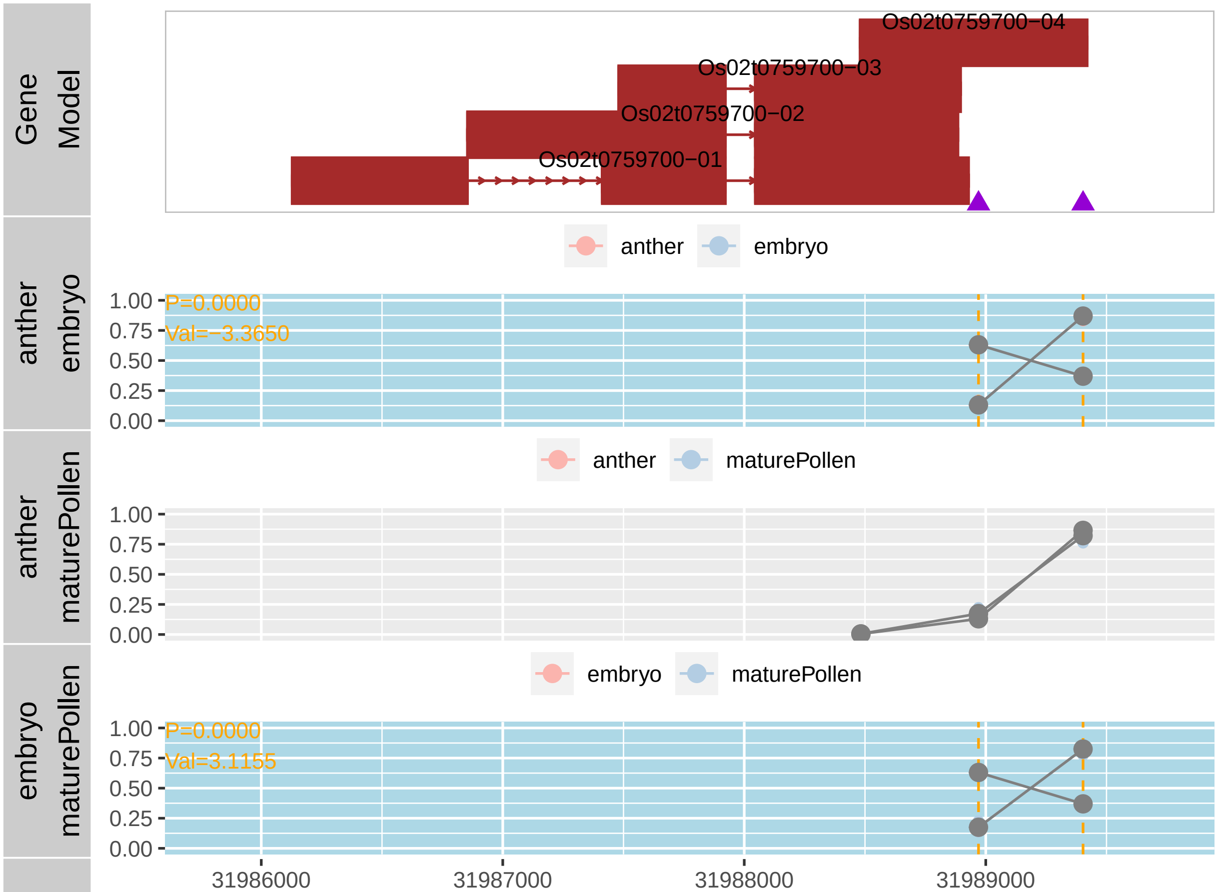
Plot all PACs of this gene in all conditions and replicates. Highlight PACs involving in the switching analysis in orange.

```
movViz(object=swDE, gene=gene, txdb=NULL, PACds=PACds, showRatio=TRUE,
       padjThd=0.01, showAllPA=TRUE)
```



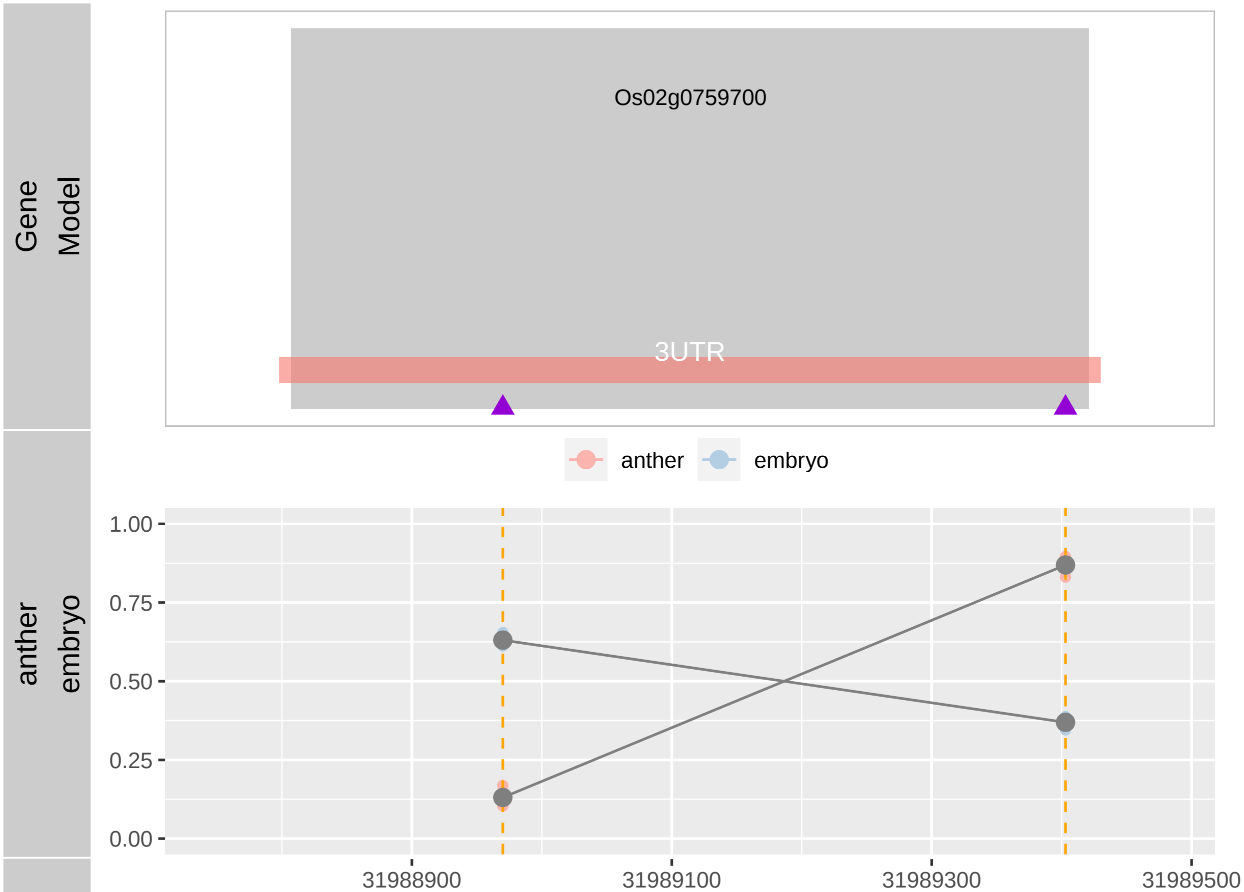
Show in each track a condition pair and use line to link PACs to show the trend. There is 3'UTR switching between anther and maturePollen, and embryo and maturePollen. Highlight specific condition pair with blue background and only show PACs involving the switching analysis with a dashed line in orange.

```
movViz(object=swDE, gene=gene, txdb=gff, PACds=PACds, collapseConds=TRUE,
        conds=swDE@conds, highlightConds=swDE@conds[c(1,3), ], showRatio=TRUE,
        linkPAs=TRUE, padjThd=0.01, showAllPA=FALSE, showPV=TRUE)
```



Show only the condition pair anther~embryo and only PACs involving the 3UTR switching. Do not show gene model but only the genomic region of PACs, and show all PACs but hightlight the switching PACs in dashed yellow line. Show only switching PACs.

```
movViz(object=swDE, gene=gene, txdb=NULL, PACds=PACds, collapseConds=TRUE,
       conds=swDE@conds[1, ], highlightConds=NULL, showRatio=TRUE, linkPAs=TRUE,
       padjThd=0.01, showAllPA=FALSE, showPV=FALSE)
```



This example shows using heatmaps for DEPAC results. First call the differential analysis and then call *movStat* to stat the results.

```
stat=movStat(object=swDE, padjThd=0.01, valueThd=1)
#> All cond pairs in heat@colData, get de01 and deNum
stat$nsig
#>                       sig.num
#> anther.embryo               9
#> anther.maturePollen         6
#> embryo.maturePollen        19
```

Output stat results into files. The pdf file stores the plots about the number of significant events and the overlap among different condition pairs.

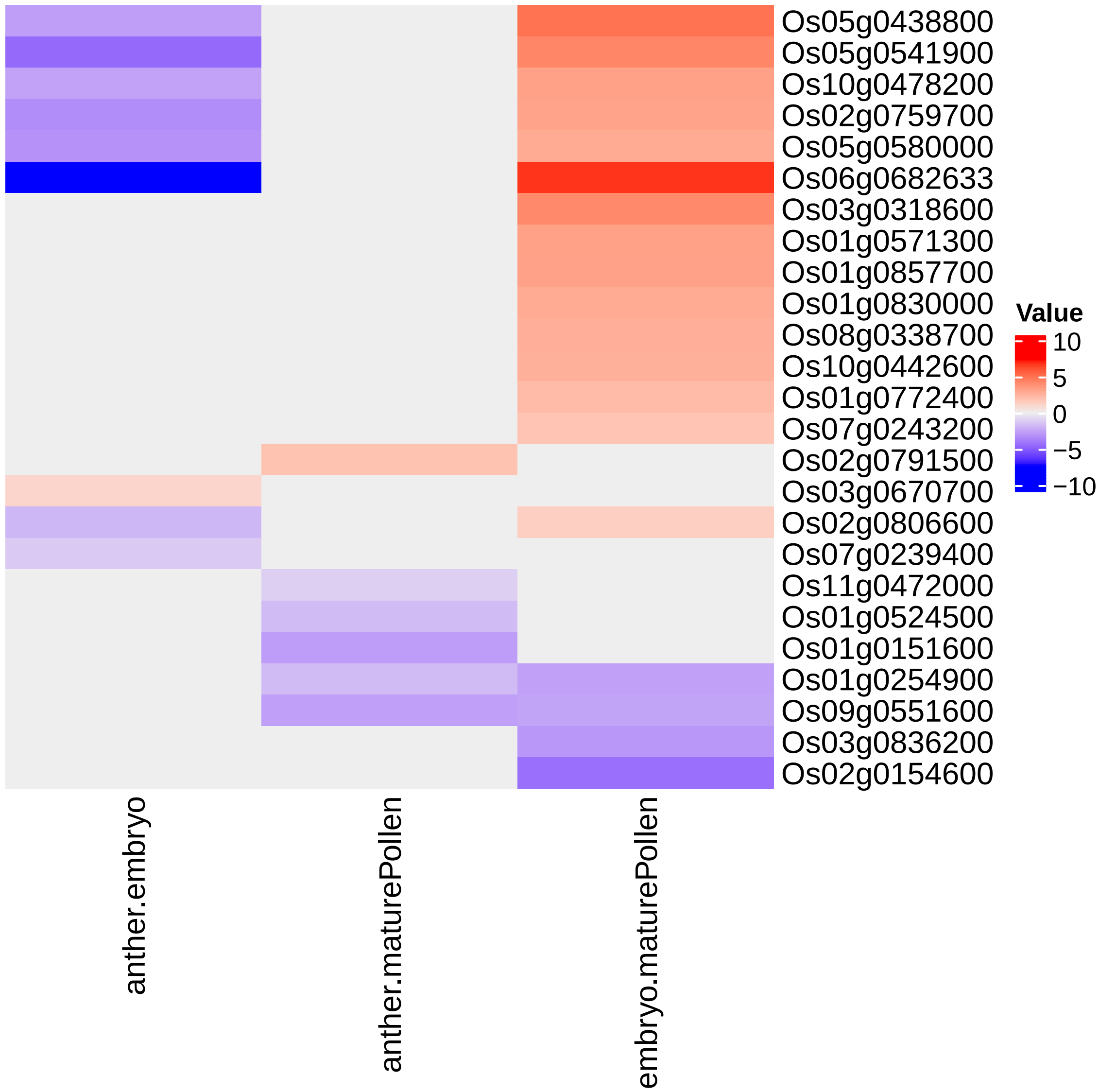
```
outputHeatStat(heatStats=stat, ostatfile='3UTR_switching_DE.stat',
               plotPre='3UTR_switching_DE', show_rownames = TRUE)
```

To plot heatmap mannually, first convert the *movRes* object to a heatmap object and then filter switching genes.

```
heat=movRes2heatmapResults(swDE)
heatUp=subsetHeatmap(heat, padjThd=0.05, valueThd=1)
```

From the heatmap, we can see gene Os06g0682633 is shorter from anther to embryo (value=-8) and longer from embryo to maturePollen (value=7).

```r
plotHeatmap(heatUp@value, show_rownames=TRUE, plotPre=NULL, cluster_rows=TRUE)
```



Get the switching information for this gene.

```r
fl=swDE@fullList$anther.embryo
fl[fl$gene=='Os06g0682633',]
#>            gene nPAC geneTag1 geneTag2 avgUTRlen1 avgUTRlen2     fisherPV
#> 7 Os06g0682633    2       11      353       1181   558.3938 1.420396e-12
#>       logFC change                   PA1                   PA2 dist nDEPA
#> 7 -7.370687     -1 Os06g0682633:28447307 Os06g0682633:28447973  667     1
#>   nSwitchPair                                          PAs1
#> 7           3 Os06g0682633:28447307=0;Os06g0682633:28447973=11
#>                                                     PAs2
#> 7 Os06g0682633:28447307=330;Os06g0682633:28447973=23
```

11 APA site switching

The function *movAPAswitch* is used to detect both canonical and non-canonical APA site switching events. The strategy of *movAPAswitch* is similar to the strategy based on DE PACs in *movUTRtrend* but with higher flexibility. If a gene has more than two PACs, then each pair of PACs (denoted as PA1 and PA2) are analyzed. The following criteria are used to determine a APA switching event: whether PA1 or PA2 are DE; average read count for both sites; distance between PA1 and PA2; average read count for a gene; relative change of PA1 and PA2 (RC); read count ratio (PA1:PA2) >1 in one sample and <1 in another sample; p-value of the Fisher' s exact test for PA1 and PA2 read counts between samples. Pairs of PACs that meet user specified conditions are considered as APA site switching events. Users can use the *movSelect* function to filter 3' UTR switching events or APA site switching events with higher flexibility.

11.1 Detect 3'UTR-PAC switching

First get DE PAC results by DEXseq.

```
DEXPAC=movDEPAC(PACds, method='DEXseq', group='group', minSumPAT=10)
```

Then get 3'UTR switching genes, usig selectOne=NULL to detect all pairs of switching PACs.

```
swDEX=movAPAswitch(PACds, group='group',aMovDEPACRes=DEXPAC,
                   avgPACtag=5, avgGeneTag=10,
                   only3UTR=TRUE,
                   DEPAC.padjThd=0.1, nDEPAC=1,
                   mindist=50, fisherThd=0.1, logFCThd=0.5,
                   cross=FALSE, selectOne=NULL)
```

Stat the switching results.

```
stat=movStat(object=swDEX, padjThd=0.1, valueThd=1)
#> All cond pairs in heat@colData, get de01 and deNum
stat$nsig
#>                       sig.num
#> anther.embryo              32
#> anther.maturePollen        11
#> embryo.maturePollen        38
```

Output switching genes with full information for anther~embryo.

```
sel=movSelect(aMovRes=swDEX, condpair='anther.embryo',
             padjThd=0.1, valueThd=1, out='full')
head(sel, n=2)
#>           gene nPAC geneTag1 geneTag2 avgUTRlen1 avgUTRlen2     fisherPV
#> 1 Os01g0151600    2       84      176   231.4643   191.7955 6.670538e-05
#> 3 Os01g0655400    2       70       76   144.7857   200.5000 2.241096e-02
#>       logFC change                  PA1                     PA2 dist nDEPA
#> 1 -1.552604     -1  Os01g0151600:2795487  Os01g0151600:2795636  150     2
#> 3  1.120104      1 Os01g0655400:26602269 Os01g0655400:26601984  286     1
#>   nSwitchPair                                                PAs1
#> 1           1   Os01g0151600:2795487=33;Os01g0151600:2795636=51
#> 3           1 Os01g0655400:26602269=45;Os01g0655400:26601984=25
```

```
#>                                                PAs2
#> 1  Os01g0151600:2795487=116;Os01g0151600:2795636=60
#> 3 Os01g0655400:26602269=34;Os01g0655400:26601984=42
```

11.2 Detect APA-site switching

Detect APA switching events involving non-3'UTR PACs, using selectOne=NULL to get all pairs of switching PACs.

```
swDE=movAPAswitch(PACds, group='group', aMovDEPACRes=DEXPAC,
                   avgPACtag=10, avgGeneTag=20,
                   only3UTR=FALSE,
                   DEPAC.padjThd=0.1, nDEPAC=1,
                   mindist=50, fisherThd=0.1, logFCThd=0.5,
                  cross=FALSE, selectOne=NULL)
```

Stat the switching results.

```
stat=movStat(object=swDE, padjThd=0.1, valueThd=1)
#> All cond pairs in heat@colData, get de01 and deNum
stat$nsig
#>                      sig.num
#> anther.embryo             43
#> anther.maturePollen       21
#> embryo.maturePollen       57
```

Output switching genes with full information for anther~embryo.

```
sw=movSelect(aMovRes=swDE, condpair='anther.embryo',
             padjThd=0.01, valueThd=1, out='full')
head(sw[order(sw$fisherPV), ], n=2)
#>            gene nPAC geneTag1 geneTag2     fisherPV     logFC change
#> 25 Os04g0635800    2       94     3437 7.680786e-75 -6.255737     -1
#> 16 Os02g0790500    2      549      124 6.634803e-35 -3.837820     -1
#>                     PA1                   PA2 dist nDEPA nSwitchPair
#> 25 Os04g0635800:32341126 Os04g0635800:32339713 1414     2           1
#> 16 Os02g0790500:33573091 Os02g0790500:33573166   76     2           1
#>                                                   PAs1
#> 25  Os04g0635800:32341126=21;Os04g0635800:32339713=73
#> 16 Os02g0790500:33573091=81;Os02g0790500:33573166=468
#>                                                     PAs2           ftr
#> 25 Os04g0635800:32341126=3293;Os04g0635800:32339713=144 3UTR,Ext_3UTR
#> 16     Os02g0790500:33573091=89;Os02g0790500:33573166=35 3UTR,Ext_3UTR
```

11.3 Subset PACds by switching genes or PACs

First get list of genes or PACs of switching events, then subset PACds by genes or PACs.

```r
genes=movSelect(aMovRes=swDE, condpair='anther.embryo',
                padjThd=0.01, valueThd=1, out='gene')
swPAC=subsetPACds(PACds, genes=genes, verbose=TRUE)
#>                     count
#> before subsetPACds  1233
#> minExprConds>=1     1233
#> genes                138
table(swPAC@anno$ftr)
#> 
#>       3UTR       5UTR        CDS   Ext_3UTR intergenic     intron 
#>         66          8          7         31         11         15

PAs=movSelect(aMovRes=swDE, condpair='anther.embryo', padjThd=0.01,
              valueThd=1, out='pa')
swPAC=subsetPACds(PACds, PAs=PAs, verbose=TRUE)
#>                     count
#> before subsetPACds  1233
#> minExprConds>=1     1233
#> PAs                   77
table(swPAC@anno$ftr)
#> 
#>     3UTR Ext_3UTR   intron 
#>       53       22        2
```

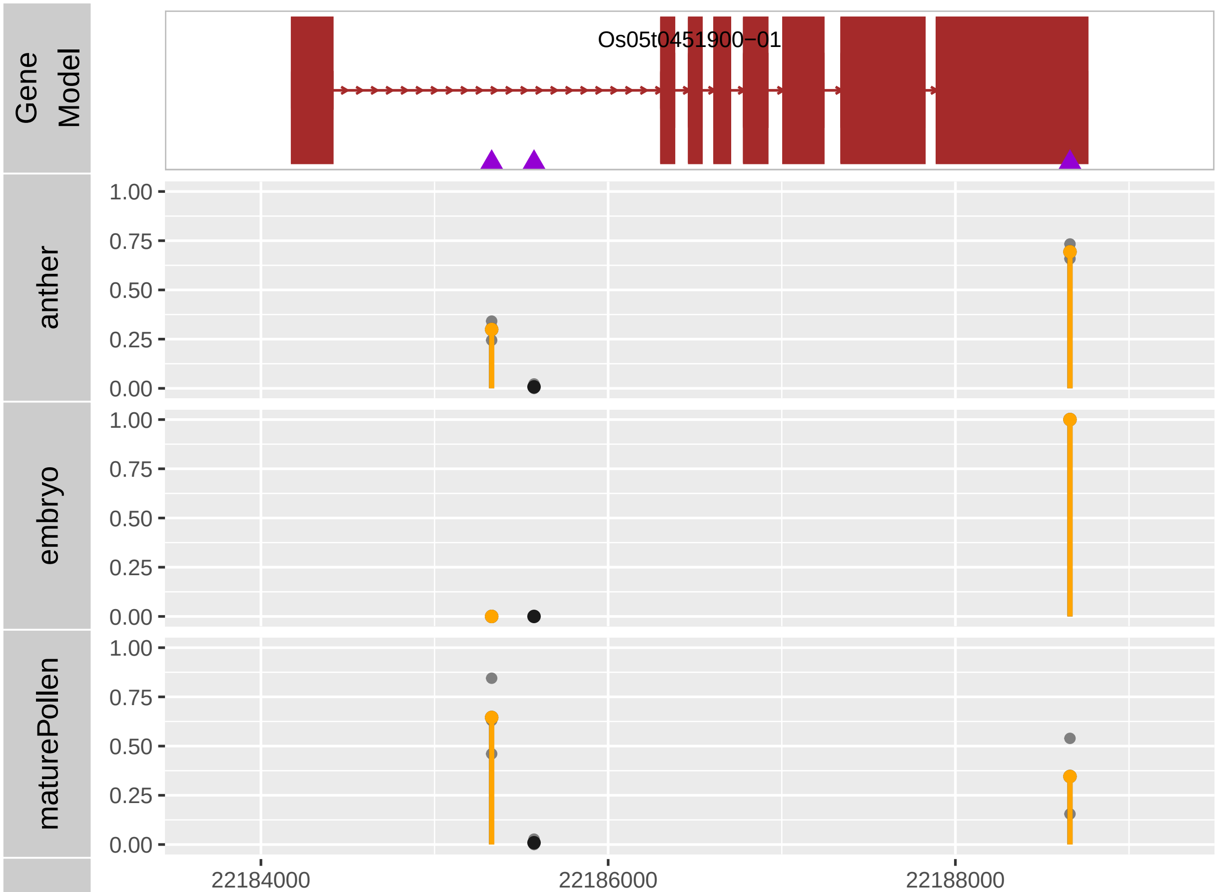
11.4 Visualization of APA-site switching

Show one switching gene (Os05g0451900), where switching happens between a 3'UTR PAC and an intronic PAC. This gene has 2 PACs in intron and 1 PAC in 3UTR; the APA-site switching happens between anther~maturePollen.

```r
gene='Os05g0451900'
gp=PACds[PACds@anno$gene==gene, ]
cbind(gp@anno$ftr, rowSums(gp@counts))
#>                       [,1]     [,2] 
#> Os05g0451900:22185329 "intron" "294"
#> Os05g0451900:22185573 "intron" "6"  
#> Os05g0451900:22188660 "3UTR"   "223"
```

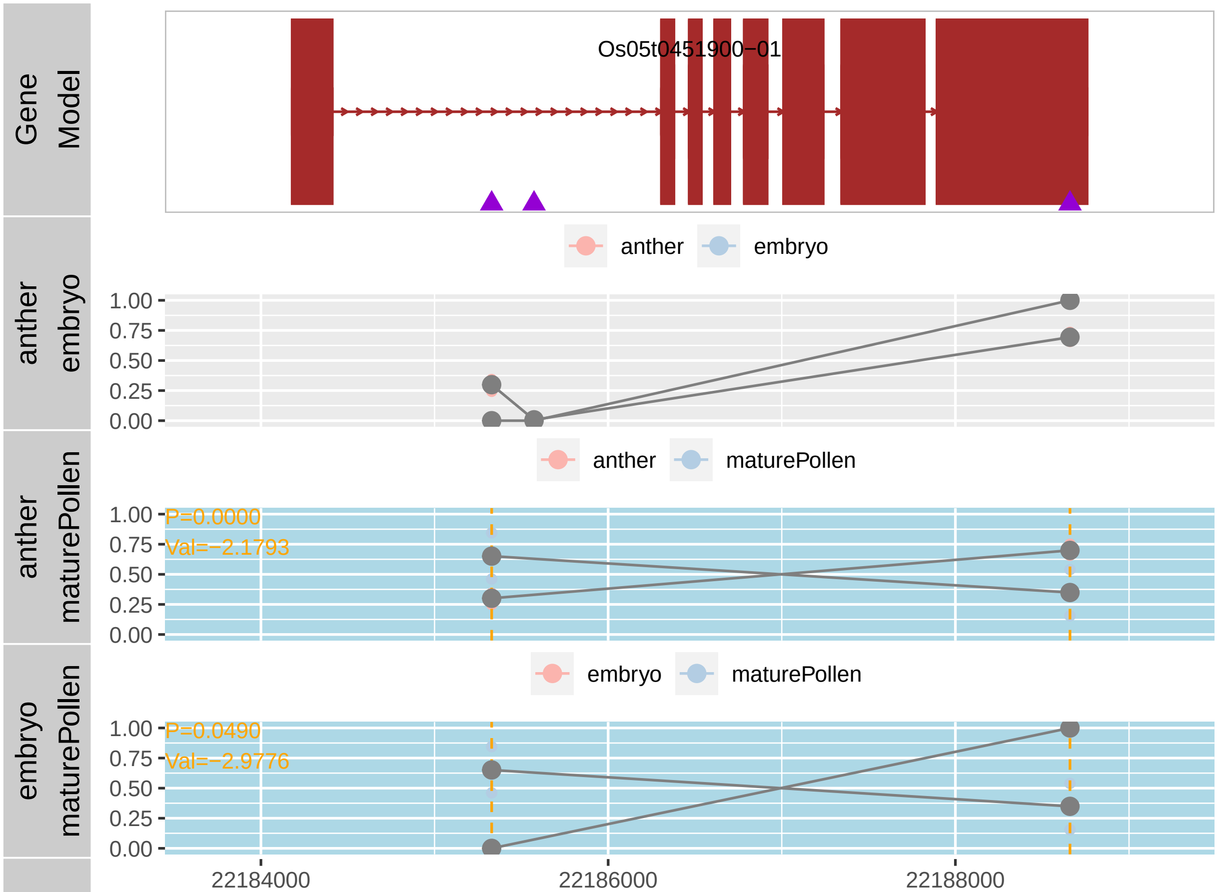
Plot all PACs of this gene in all conditions and replicates. Highlight PACs involving in the switching analysis in orange.

```r
movViz(object=swDE, gene=gene, txdb=gff, PACds=PACds,
       showRatio=TRUE, padjThd=0.01, showAllPA=TRUE)
```



Show in each track a condition pair and use line to link PACs to show the trend. Highlight specific condition pair with blue background and only show PACs involving the switching analysis with a dashed line in orange. There is APA-site switching between anther and maturePollen.

```
movViz(object=swDE, gene=gene, txdb=gff, PACds=PACds, collapseConds=TRUE,
       conds=swDE@conds, highlightConds=swDE@conds[c(2,3), ], showRatio=TRUE,
       linkPAs=TRUE, padjThd=0.01, showAllPA=FALSE)
```



12 Session Information

The session information records the versions of all the packages used in the generation of the present document.

```
sessionInfo()
#> R version 4.2.2 (2022-10-31 ucrt)
#> Platform: x86_64-w64-mingw32/x64 (64-bit)
#> Running under: Windows 10 x64 (build 22621)
#>
#> Matrix products: default
#>
#> locale:
#> [1] LC_COLLATE=Chinese (Simplified)_China.utf8
#> [2] LC_CTYPE=Chinese (Simplified)_China.utf8
#> [3] LC_MONETARY=Chinese (Simplified)_China.utf8
#> [4] LC_NUMERIC=C
#> [5] LC_TIME=Chinese (Simplified)_China.utf8
#>
#> attached base packages:
#> [1] grid      stats4    stats     graphics  grDevices utils     datasets
#> [8] methods   base
#>
```

```
#> other attached packages:
#>  [1] DESeq2_1.38.1
#>  [2] SummarizedExperiment_1.28.0
#>  [3] MatrixGenerics_1.10.0
#>  [4] matrixStats_0.63.0
#>  [5] ComplexHeatmap_2.14.0
#>  [6] dplyr_1.0.10
#>  [7] magrittr_2.0.3
#>  [8] BSgenome.Oryza.ENSEMBL.IRGSP1_1.4.2
#>  [9] BSgenome_1.66.2
#> [10] rtracklayer_1.58.0
#> [11] Biostrings_2.66.0
#> [12] XVector_0.38.0
#> [13] TxDb.Mmusculus.UCSC.mm10.ensGene_3.4.0
#> [14] GenomicFeatures_1.50.2
#> [15] AnnotationDbi_1.60.0
#> [16] Biobase_2.58.0
#> [17] GenomicRanges_1.50.1
#> [18] GenomeInfoDb_1.34.9
#> [19] IRanges_2.32.0
#> [20] S4Vectors_0.36.0
#> [21] BiocGenerics_0.44.0
#> [22] ggplot2_3.4.0
#> [23] movAPA_0.2.0
#>
#> loaded via a namespace (and not attached):
#>   [1] rappdirs_0.3.3              ggthemes_4.2.4
#>   [3] GGally_2.1.2                R.methodsS3_1.8.2
#>   [5] tidyr_1.2.1                 bit64_4.0.5
#>   [7] knitr_1.41                  irlba_2.3.5.1
#>   [9] DelayedArray_0.24.0         R.utils_2.12.2
#>  [11] hwriter_1.3.2.1             data.table_1.14.6
#>  [13] rpart_4.1.19                KEGGREST_1.38.0
#>  [15] TFBSTools_1.36.0            RCurl_1.98-1.9
#>  [17] AnnotationFilter_1.22.0     doParallel_1.0.17
#>  [19] generics_0.1.3              RSQLite_2.2.18
#>  [21] proxy_0.4-27                bit_4.0.5
#>  [23] tzdb_0.3.0                  xml2_1.3.3
#>  [25] assertthat_0.2.1            DirichletMultinomial_1.40.0
#>  [27] xfun_0.35                   hms_1.1.2
#>  [29] evaluate_0.18               DEoptimR_1.0-11
#>  [31] fansi_1.0.3                 restfulr_0.0.15
#>  [33] progress_1.2.2              caTools_1.18.2
#>  [35] dbplyr_2.2.1                DBI_1.1.3
#>  [37] geneplotter_1.76.0          htmlwidgets_1.5.4
#>  [39] reshape_0.8.9               purrr_0.3.5
#>  [41] ellipsis_0.3.2              RSpectra_0.16-1
#>  [43] backports_1.4.1             grImport2_0.2-0
#>  [45] annotate_1.76.0             biomaRt_2.54.0
#>  [47] deldir_1.0-6                vctrs_0.5.1
#>  [49] SingleCellExperiment_1.20.0 ensembldb_2.22.0
#>  [51] Cairo_1.6-0                 TTR_0.24.3
#>  [53] abind_1.4-5                 cachem_1.0.6
```

```
#>  [55] RcppEigen_0.3.3.9.3          withr_2.5.0
#>  [57] robustbase_0.95-0            checkmate_2.1.0
#>  [59] vcd_1.4-11                   GenomicAlignments_1.34.0
#>  [61] xts_0.13.0                   prettyunits_1.1.1
#>  [63] cluster_2.1.4                lazyeval_0.2.2
#>  [65] seqLogo_1.64.0               laeken_0.5.2
#>  [67] crayon_1.5.2                 genefilter_1.80.0
#>  [69] edgeR_3.40.0                 pkgconfig_2.0.3
#>  [71] labeling_0.4.2               ProtGenerics_1.30.0
#>  [73] nnet_7.3-18                  rlang_1.0.6
#>  [75] lifecycle_1.0.3              filelock_1.0.2
#>  [77] BiocFileCache_2.6.0          dichromat_2.0-0.1
#>  [79] RcppHNSW_0.4.1               lmtest_0.9-40
#>  [81] graph_1.76.0                 Matrix_1.5-3
#>  [83] carData_3.0-5                boot_1.3-28
#>  [85] zoo_1.8-11                   base64enc_0.1-3
#>  [87] GlobalOptions_0.1.2          png_0.1-7
#>  [89] rjson_0.2.21                 bitops_1.0-7
#>  [91] R.oo_1.25.0                  blob_1.2.3
#>  [93] shape_1.4.6                  stringr_1.4.1
#>  [95] readr_2.1.3                  jpeg_0.1-10
#>  [97] CNEr_1.34.0                  scales_1.2.1
#>  [99] memoise_2.0.1                plyr_1.8.8
#> [101] hexbin_1.28.3                zlibbioc_1.44.0
#> [103] compiler_4.2.2               tinytex_0.43
#> [105] BiocIO_1.8.0                 RColorBrewer_1.1-3
#> [107] pcaMethods_1.90.0            clue_0.3-63
#> [109] Rsamtools_2.14.0             cli_3.4.1
#> [111] ade4_1.7-22                  htmlTable_2.4.1
#> [113] Formula_1.2-4                ggplot.multistats_1.0.0
#> [115] MASS_7.3-58.1                tidyselect_1.2.0
#> [117] stringi_1.7.8                highr_0.9
#> [119] yaml_2.3.6                   locfit_1.5-9.6
#> [121] latticeExtra_0.6-30          VariantAnnotation_1.44.0
#> [123] tools_4.2.2                  parallel_4.2.2
#> [125] circlize_0.4.15              rstudioapi_0.14
#> [127] TFMPvalue_0.0.9              foreach_1.5.2
#> [129] foreign_0.8-83               DEXSeq_1.44.0
#> [131] gridExtra_2.3                smoother_1.1
#> [133] scatterplot3d_0.3-43         farver_2.1.1
#> [135] digest_0.6.30                BiocManager_1.30.19
#> [137] pracma_2.4.2                 Rcpp_1.0.9
#> [139] car_3.1-1                    OrganismDbi_1.40.0
#> [141] httr_1.4.4                   motifStack_1.42.0
#> [143] ggbio_1.46.0                 biovizBase_1.46.0
#> [145] colorspace_2.0-3             XML_3.99-0.12
#> [147] ranger_0.14.1                splines_4.2.2
#> [149] statmod_1.4.37               RBGL_1.74.0
#> [151] sp_1.5-1                     xtable_1.8-4
#> [153] poweRlaw_0.70.6              destiny_3.12.0
#> [155] R6_2.5.1                     Hmisc_5.0-0
#> [157] pillar_1.8.1                 htmltools_0.5.3
#> [159] glue_1.6.2                   fastmap_1.1.0
```

```
#> [161] VIM_6.2.2                  BiocParallel_1.32.1
#> [163] class_7.3-20               codetools_0.2-18
#> [165] utf8_1.2.2                 lattice_0.20-45
#> [167] tibble_3.1.8               curl_4.3.3
#> [169] gtools_3.9.4               magick_2.7.3
#> [171] GO.db_3.16.0               interp_1.1-3
#> [173] survival_3.4-0             limma_3.54.0
#> [175] rmarkdown_2.18             munsell_0.5.0
#> [177] e1071_1.7-13               GetoptLong_1.0.5
#> [179] GenomeInfoDbData_1.2.9     iterators_1.0.14
#> [181] reshape2_1.4.4             gtable_0.3.1
```

13 References

[1] Fu, H., Yang, D., Su, W., et al. Genome-wide dynamics of alternative polyadenylation in rice. Genome Res. 2016;26(12):1753-1760.

[2] Zhu, S., Ye, W., Ye, L., et al. PlantAPAdb: A Comprehensive Database for Alternative Polyadenylation Sites in Plants. Plant Physiol. 2020;182(1):228-242.

[3] Anders, S. and Huber, W. Differential expression analysis for sequence count data. Genome Biol. 2010;11(10):2010-2011.

[4] Robinson, M.D., McCarthy, D.J. and Smyth, G.K. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 2010;26(1):139-140.

[5] Ji, Z., Lee, J.Y., Pan, Z., et al. Progressive lengthening of 3' untranslated regions of mRNAs by alternative polyadenylation during mouse embryonic development. Proc. Natl. Acad. Sci. USA 2009;106(17):7028-7033.

[6] Ulitsky, I., Shkumatava, A., Jan, C.H., et al. Extensive alternative polyadenylation during zebrafish development. Genome Res. 2012;22(10):2054-2066.

[7] Begik, O., Oyken, M., Cinkilli Alican, T., et al. Alternative Polyadenylation Patterns for Novel Gene Discovery and Classification in Cancer. Neoplasia 2017;19(7):574-582.

[8] Shulman, E.D. and Elkon, R. Cell-type-specific analysis of alternative polyadenylation using single-cell transcriptomics data. Nucleic Acids Res 2019;47(19):10027-10039.

[9] Lianoglou, S., Garg, V., Yang, J.L., et al. Ubiquitously transcribed genes use alternative polyadenylation to achieve tissue-specific expression. Genes Dev. 2013;27(21):2380-2396.