Analyses of APA dynamics in mouse sperm cells with the movAPA package

Xiaohui Wu, Wenbin Ye, Tao Liu, Hongjuan Fu

Last modified 2020-07-25

Contents

1 Overview **1**

2 Preparations **2**
2.1 PAC data of mouse sperm cells 2
2.2 Reference genome 3
2.3 Genome annotation 3

3 Preprocessing of PAC data **3**
3.1 Extending annotated 3'UTRs 3
3.2 Normalization 4
3.3 Filter PACs or cells 4

4 Statistics of PACds **4**
4.1 PAC distributions among cell types 4
4.2 PAC distributions in single cells 5

5 Analyses of APA dynamics **7**
5.1 Detecting DE PACs 7
5.2 Fisher's exact test for APA genes 11
5.3 Detecting 3'UTR switching genes 12
5.4 Visualization of 3'UTR switching genes 15
5.5 Proximal PAC's GPI index 17

6 Visualize PACs in single cells **19**

7 Session Information **22**

8 References **24**

1 Overview

Here we investigated the application of movAPA on poly(A) sites (or called poly(A) site clusters, PACs) from mouse sperm cells. Poly(A) sites from three stages of differentiation process were

obtained from the previous study (Shulman and Elkon, 2019), including early stage (spermatocytes, SC), intermediate stage (round spermatids, RS), and late stage (elongating spermatids, ES). We used a small dataset containing 3' UTR poly(A) sites from the chromosome 12 for demonstration.

2 Preparations

2.1 PAC data of mouse sperm cells

movAPA is highly scalable and flexible in that the dataset of APA sites in single cells can be readily represented by the generic object of *PACdataset* where cells of the same cell type are regarded as replicates of a biological sample.

The moveAPA package includes an example single cell PAC dataset stored as a *PACdataset* object, containing 771 PACs from 396 genes located in chromosome 12. There are total 2042 cells from three cell types. This dataset contains the gene *Psen1* (ENSMUSG00000019969) presented in Shulman et al, 2019.

```
## library(movAPA, warn.conflicts = FALSE, quietly=TRUE)
data(scPACds)
head(scPACds@counts[1:2,1:5])
#>        AAACCTGAGAGGGCTT AAACCTGAGCTTATCG AAACCTGCATACGCCG AAACCTGGTTGAGTTC
#> PA3443                0                0                0                0
#> PA3446                0                0                0                0
#>        AAACCTGTCAACGAAA
#> PA3443                0
#> PA3446                0
head(scPACds@anno, n=2)
#>          chr strand     coord   peakID  ftr gene_type ftr_start   ftr_end
#> PA3443 chr12      - 100125475 peak3443 3UTR      <NA> 100125452 100125605
#> PA3446 chr12      - 100549890 peak3446 3UTR      <NA> 100549778 100551443
#>                       gene gene_start  gene_end gene_stop_codon upstream_id
#> PA3443 ENSMUSG00000021179  100125452 100159653       100125606        <NA>
#> PA3446 ENSMUSG00000021180  100549778 100725028       100551444        <NA>
#>        upstream_start upstream_end downstream_id downstream_start
#> PA3443             NA           NA          <NA>               NA
#> PA3446             NA           NA          <NA>               NA
#>        downstream_end three_UTR_length three_extend
#> PA3443             NA              131           NA
#> PA3446             NA             1554           NA
head(scPACds@colData, n=2)
#>                             group celltype        tsn1        tsn2
#> AAACCTGAGAGGGCTT AAACCTGAGAGGGCTT       SC 22.54797966 4.077467845
#> AAACCTGAGCTTATCG AAACCTGAGCTTATCG       RS 1.138437608 -32.9317999
levels(scPACds@colData$celltype)
#> [1] "ES" "RS" "SC"
```

2.2 Reference genome

The reference genome is not necessary for this case study, 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. To use *BSgenome* object, please refer to the *BSgenome* package for obtaining a *BSgenome* object for your species.

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* is used to parse the given annotation and the processed annotation can be saved into an rdata object for further use. The genome annotation file is not necessary for this case study as the information has been stored in scPACds.

Process the genome annotation of mm10 represented as TxDb object.

```
library(TxDb.Mmusculus.UCSC.mm10.ensGene)
txdbmm10 <- parseGenomeAnnotation(TxDb.Mmusculus.UCSC.mm10.ensGene)
table(txdbmm10$anno.need$type)
gff=txdbmm10
save(gff, file='txdbmm10.rda')
```

3 Preprocessing of PAC data

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

```
table(scPACds@anno$ftr)
#>
#>       3UTR        CDS intergenic     intron
#>        644          3         94         30
scPACds=ext3UTRPACds(scPACds, ext3UTRlen=2000)
#> 70 PACs in extended 3UTR (ftr=intergenic >> ftr=3UTR)
#> Get 3UTR length (anno@toStop) for 3UTR/extended 3UTR PACs
table(scPACds@anno$ftr)
#>
#>       3UTR        CDS intergenic     intron
#>        714          3         24         30
```

3.2 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).

Here is an exmple to normalize the data using the TPM method.

```
scPACdsNorm2=normalizePACds(scPACds, method='TPM')
head(colSums(scPACdsNorm2@counts))
#> AAACCTGAGAGGGCTT AAACCTGAGCTTATCG AAACCTGCATACGCCG AAACCTGGTTGAGTTC
#>              737              696             2425              558
#> AAACCTGTCAACGAAA AAACCTGTCGCGGATC
#>              789              152
```

3.3 Filter PACs or cells

Filter PACs with total counts>=20 and remove intergenic PACs.

```
scPACdsFlt=subsetPACds(scPACds, totPACtag=20, choosePA=NULL,
                       noIntergenic=TRUE, verbose=TRUE)
#>                     txt
#> before subsetPACds 771
#> noItg              747
#> totPACtag>=20      682
```

Filter only PACs in 3'UTR and obtain PACs in 3'UTRs with >=2 PACs.

```
scPACdsFlt=get3UTRAPAds(scPACdsFlt, sortPA=TRUE, choose2PA=NULL)
length(scPACdsFlt)
#> [1] 411
```

4 Statistics of PACds

4.1 PAC distributions among cell types

To make statistics of PACs among cell types, first we pool cells of the same cell type.

```
scPACdsCt=subsetPACds(scPACds, group='celltype', pool=TRUE)
```

Make statistics of PAC distributions in each cell type.

```
scPACdsCtStat=movStat(scPACdsCt, minPAT=c(20, 40), ofilePrefix=NULL)
```

Statistical results of PACs with total read counts >=20.

```
scPACdsCtStat$pat20
#>       nPAC    nPAT nGene nAPAgene APAextent 3UTR_nPAT CDS_nPAT intergenic_nPAT
#> SC     567  715602   354      142 0.4011299    706741      989            1816
```

```
#> RS     564  810878   366      134 0.3661202    795569      586            2513
#> ES     277  118707   225       45 0.2000000    116090        0             835
#> total  669 1645187   403      167 0.4143921   1618400     1575            5164
#>       intron_nPAT 3UTR_nPAC CDS_nPAC intergenic_nPAC intron_nPAC
#> SC           6056       534        2              13          18
#> RS          12210       534        1              11          18
#> ES           1782       266        0               3           8
#> total       20048       626        2              19          22
```

Plot statistical results by various barplots. Results showed that there are more PACs expressed in RS and SC than in ES.

```
plotPACdsStat(scPACdsCtStat, pdfFile=NULL)
```

4.2 PAC distributions in single cells

Make statistics for PAT and PAC distributions in each cell, using PAT cutoffs 1 and 5.

```
scPACdsStat=movStat(scPACds, minPAT=c(1,5), ofilePrefix='scPACds.stat')
#> >>> scPACds.stat.pat1.stat
#> >>> scPACds.stat.pat5.stat
```

Statistics of pooled data

```
scPACdsStat$pat1['total',]
#>       nPAC    nPAT nGene nAPAgene APAextent 3UTR_nPAT CDS_nPAT intergenic_nPAT
#> total  771 1650337   436      197 0.4518349   1622965     1618            5442
#>       intron_nPAT 3UTR_nPAC CDS_nPAC intergenic_nPAC intron_nPAC
#> total       20312       714        3              24          30
```

Summary of PAC# in each cell, ranging from 56 PACs per cell to 354 PACs per cell.

```
summary(scPACdsStat$pat1$nPAC[1:(nrow(scPACdsStat$pat1)-1)])
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>      56     128     147     154     169     354
```

Summary of PAT# (read count) in each cell, ranging from 154 PACs per cell to 5712 PACs per cell.

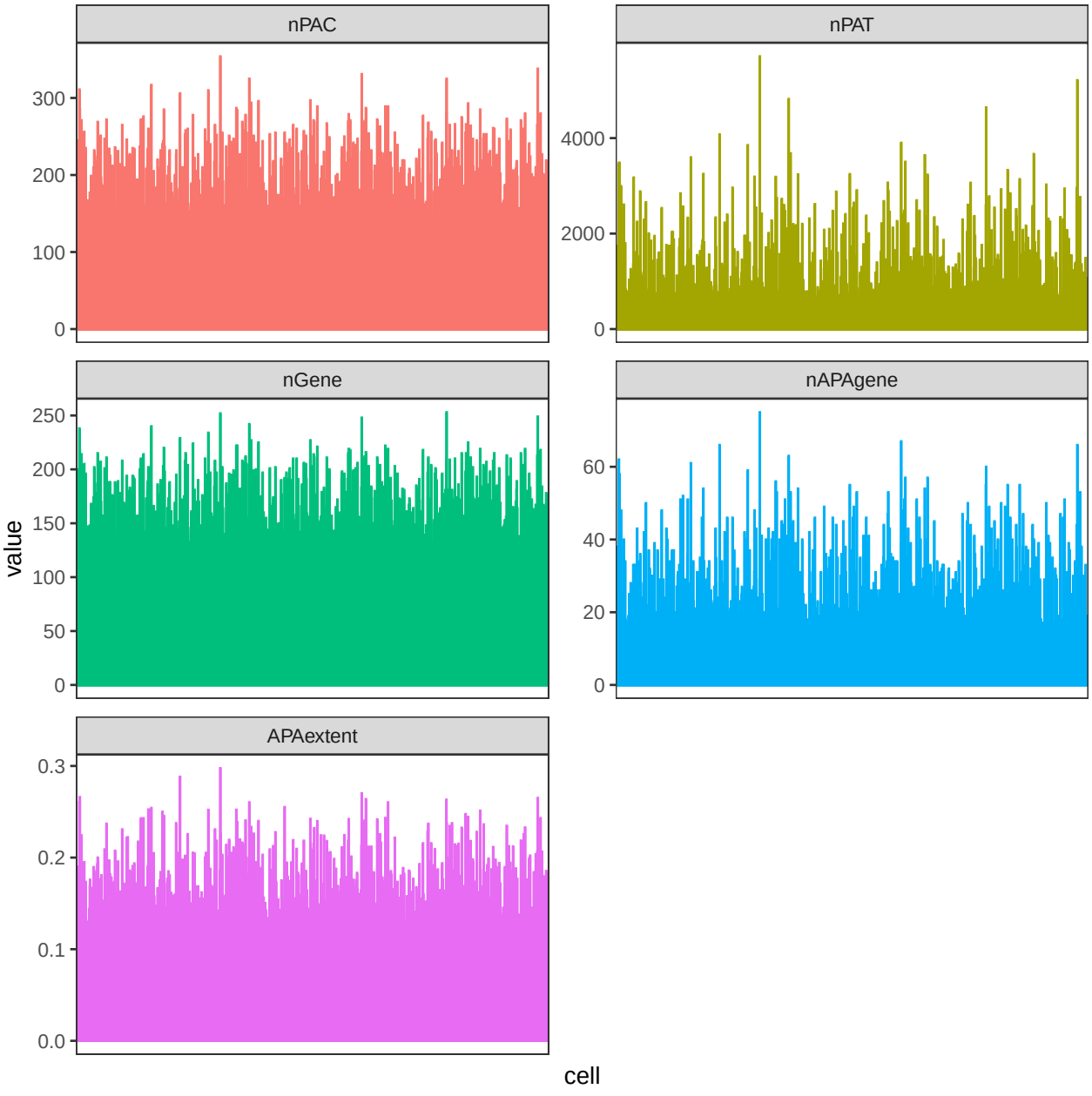
```
summary(scPACdsStat$pat1$nPAT[1:(nrow(scPACdsStat$pat1)-1)])
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   154.0   503.2   644.5   808.2   847.8  5712.0
```

Here we plot barplots showing distributions of PACs and PATs among cells. First we create the data for plot using all PACs (PAT cutoff=1), and remove the 'total' line.

```
d=scPACdsStat$pat1[, c('nPAC','nPAT','nGene','nAPAgene','APAextent')]
d$cell=rownames(d)
d=d[1:(nrow(d)-1), ]
d=melt(d)
```

Plot barplots to Show the distribution of PAT#, PAT#, gene#, APA gene#, and APA gene%.

```
sp <- ggplot(data=d, aes(x=cell, y=value, color=variable)) +
  geom_bar(stat='identity', width=1.1)
sp = sp + theme_bw() +  guides(color=FALSE) +
  theme(axis.ticks.x = element_blank(), axis.text.x = element_blank(),
        panel.grid.minor=element_blank(),
        panel.grid.major=element_blank())
sp + facet_wrap( ~ variable, ncol=2, scales="free")
```



5 Analyses of APA dynamics

5.1 Detecting 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"). The strategy of chi-squared test was used in the study on single cell APA for detecting differential usage of PACs among cells (Shulman and Elkon, 2019). For single cell data, we highly recommand the chisq method because it is much faster than the other two methods.

Detecting DE PACs using chisq method for genes with total counts>=50.

```
DEqPAC=movDEPAC(scPACdsCt, method='chisq', group='celltype',
                minSumPAT=50, chisqPadjust=TRUE)
```

Make statistics of the DE PAC result from the chisq 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.

```
stat=movStat(object=DEqPAC, padjThd=0.05, valueThd=0.95)
#> All cond pairs in heat@colData, get de01 and deNum
stat$nsig
#>       sig.num
#> SC.RS     167
#> SC.ES     140
#> RS.ES      85
head(stat$ovp)
#>          pair n1.sig.num n2.sig.num novp.sig.num
#> 1 SC.RS-SC.ES        167        140          115
#> 2 SC.RS-RS.ES        167         85           65
#> 3 SC.ES-RS.ES        140         85           78
head(stat$siglist[[1]])
#> [1] "PA11112" "PA11113" "PA11288" "PA11291" "PA11375" "PA11467"
```

Output full list of DE PACs.

```
sel=movSelect(aMovRes=DEqPAC, condpair='SC.RS',
              padjThd=0.05, valueThd=0.95,
              out='full', PACds=scPACdsCt)
head(sel, n=2)
#>        PA   chr strand    coord    peakID  ftr gene_type ftr_start  ftr_end
#> 1 PA11112 chr12      + 86963789 peak11112 3UTR      <NA>  86962668 86965362
#> 2 PA11113 chr12      + 86965333 peak11113 3UTR      <NA>  86962668 86965362
#>                 gene gene_start gene_end gene_stop_codon upstream_id
#> 1 ENSMUSG00000034157   86947343 86965362        86962667        <NA>
#> 2 ENSMUSG00000034157   86947343 86965362        86962667        <NA>
#>   upstream_start upstream_end downstream_id downstream_start downstream_end
#> 1             NA           NA          <NA>               NA             NA
#> 2             NA           NA          <NA>               NA             NA
```

```
#>   three_UTR_length three_extend toStop  SC  RS ES         padj value
#> 1             1122           NA   1122 507 573 14 6.218416e-08     1
#> 2             2666           NA   2666 586 269  7 3.042999e-10     1
```

For all DE PACs in all condtions pairs, examine the DE status of each PAC.

```
head(stat$tf01)
#>         SC.RS SC.ES RS.ES
#> PA11105     0     0     1
#> PA11112     1     0     0
#> PA11113     1     0     0
#> PA11167     0     0     1
#> PA11169     0     1     1
#> PA11288     1     1     1
## Output stat results into files: "DEqPAC.plots.pdf" and 'DEqPAC.stat'.
## outputHeatStat(heatStats=stat, ostatfile='DEqPAC.stat', plotPre='DEqPAC')
```

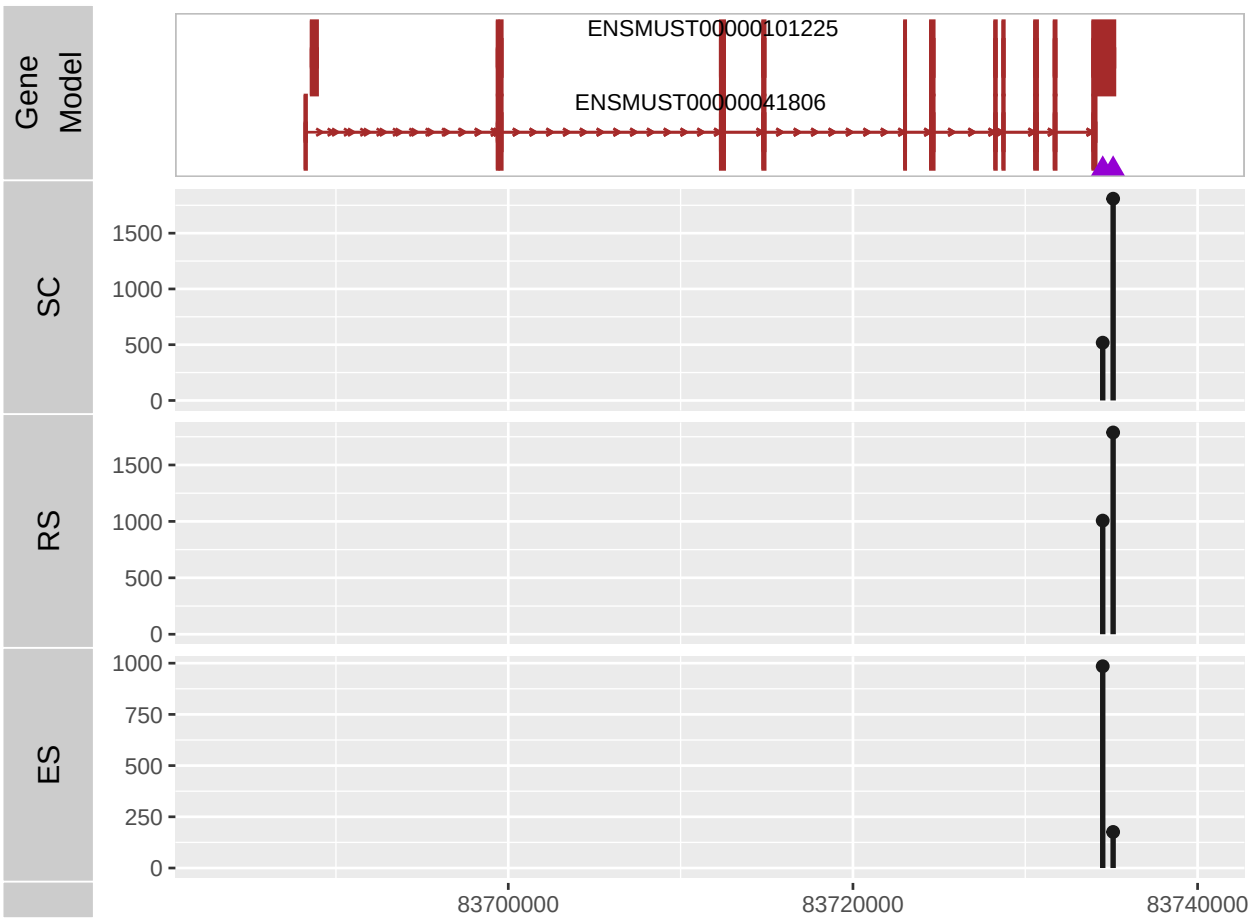
Visualize a DE PAC in gene ENSMUSG00000019969 by *movViz*.

First, we examine all PACs in this gene. There are two 3'UTR PACs (PA2503 and PA2504).

```
gene='ENSMUSG00000019969'
gp=scPACds[scPACds@anno$gene==gene, ]
cbind(gp@anno$ftr, rowSums(gp@counts))
#>        [,1]   [,2]
#> PA2503 "3UTR" "2510"
#> PA2504 "3UTR" "3773"
```

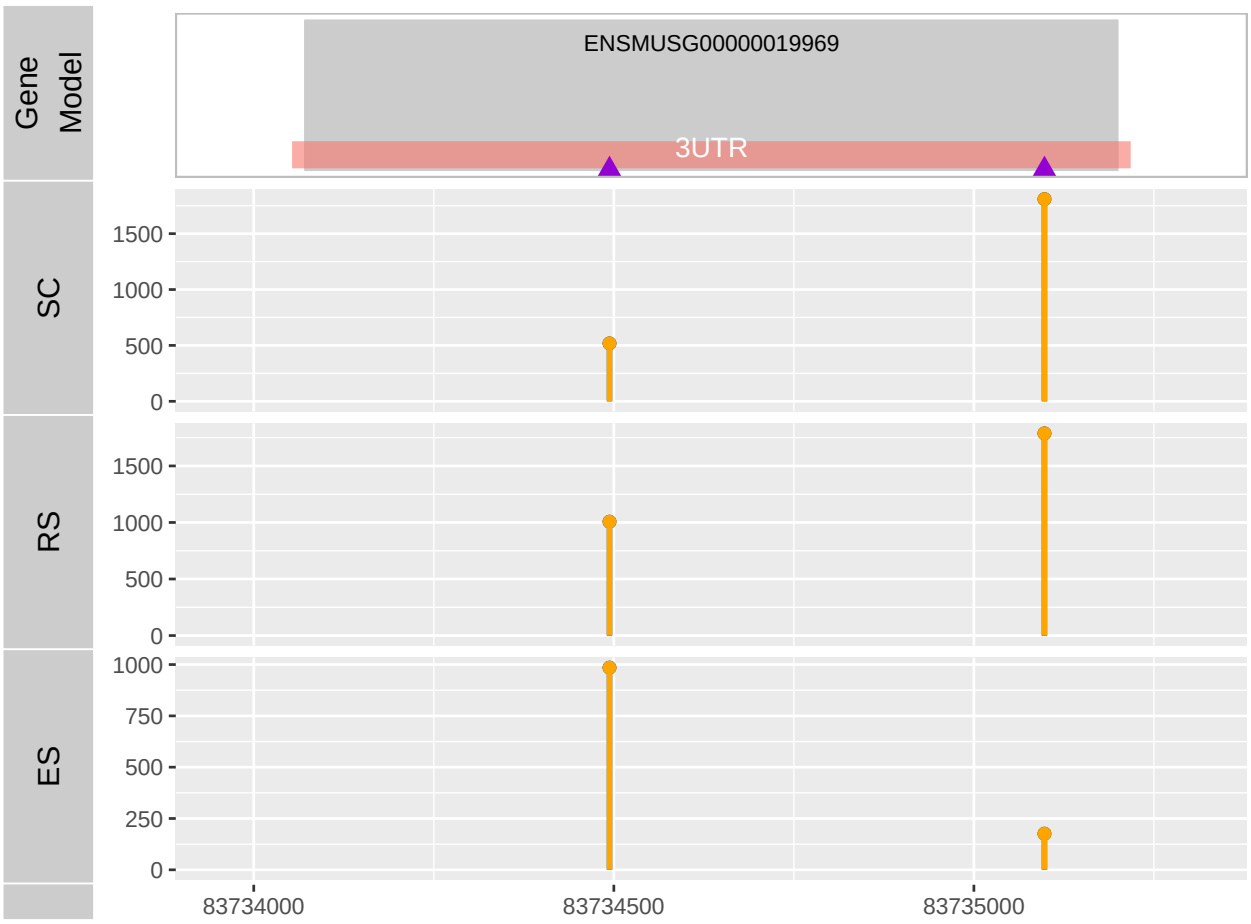
Plot all PACs in this gene. Here we used scPACdsCt instead of scPACds to plot the total expression levels of PACs in a cell type.

```
movViz(object=scPACdsCt, gene=gene, txdb=gff, group='celltype')
```



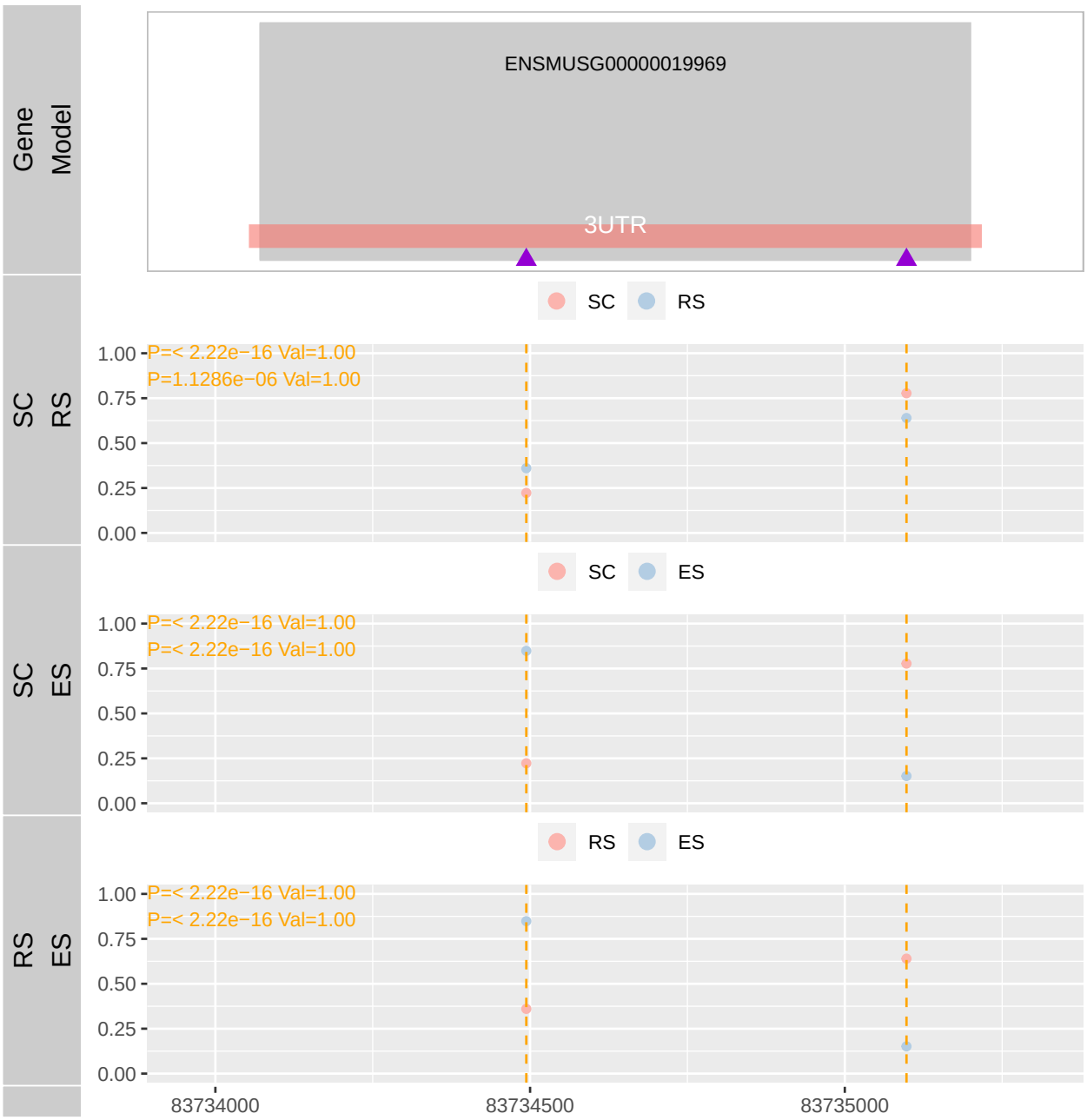
Visualize PACs of this gene in individual cell types. In the plot, the Y-axis is read count, the scale of which is different among conditions. Only show DE PACs with padj < padjThd and show 3'UTR region instead of the gene model.

```
movViz(object=DEqPAC, gene=gene, txdb=NULL,
       PACds=scPACdsCt,
       collapseConds=FALSE, padjThd=0.01,
       showRatio=FALSE, showAllPA=FALSE)
```



Show condition pairs in individual tracks. If padjThd is given, then the DE PACs (padj < padjThd) will be highlighted (dashed yellow line).

```
movViz(object=DEqPAC, gene=gene, txdb=NULL, PACds=scPACdsCt, collapseConds=T,
       padjThd=0.01, showPV=TRUE, showRatio=TRUE, showAllPA=T)
```



5.2 Fisher's exact test for APA genes

Here we detect genes with dynamic APA usages among cell types using Fisher's exact test. This is similar to the method (*test_apa*) used in (Shulman and Elkon, 2019). The Fisher's exact test is performed for genes with at least two 3'UTR PACs.

```
sw=movAPAswitch(PACds=scPACds, group='celltype',
                avgPACtag=0, avgGeneTag=0,
                only3UTR=TRUE, mergeReps='pool',
                aMovDEPACRes=NULL, DEPAC.padjThd=NULL, nDEPAC=0,
                mindist=0, fisherThd=0.05, logFCThd=0, cross=FALSE,
```

```
                selectOne='fisherPV')
#> SC.RS
#> SC.ES
#> RS.ES

head(sw@fullList$SC.RS, n=2)
#>                  gene nPAC geneTag1 geneTag2 avgUTRlen1 avgUTRlen2     fisherPV
#> 1 ENSMUSG00000002996    2       41      139   822.1707   429.2518 4.229427e-07
#> 2 ENSMUSG00000007411    2       74      807   317.4865   212.8761 5.571850e-24
#>      logFC change    PA1    PA2 dist nDEPA nSwitchPair                PAs1
#> 1 -2.82639     -1  PA535  PA536  856     0           2    PA535=8;PA536=33
#> 2 -4.47032     -1 PA1207 PA1208  248     0           1 PA1207=40;PA1208=34
#>                    PAs2
#> 1    PA535=91;PA536=48
#> 2 PA1207=778;PA1208=29
## Filter results with padj<0.05.
swstat=movStat(object=sw, padjThd=0.05, valueThd=0)
#> All cond pairs in heat@colData, get de01 and deNum
## Number of significant switching genes between cell types.
swstat$nsig
#>       sig.num
#> SC.RS     125
#> SC.ES     106
#> RS.ES      79
head(swstat$siglist$SC.RS)
#> [1] "ENSMUSG00000002996" "ENSMUSG00000007411" "ENSMUSG00000014905"
#> [4] "ENSMUSG00000017843" "ENSMUSG00000019969" "ENSMUSG00000020561"
```

5.3 Detecting 3'UTR switching genes

This is similar to the above Fisher's exact test, while it is stricter. The switching criteria: at least 1 DEqPAC; fisher's test of the two PACs pvalue=logFCThd.

If more than one switching pair was found in a gene, only the pair with the smallest Fisher's test's pvalue (selectOne='fisherPV') was returned.

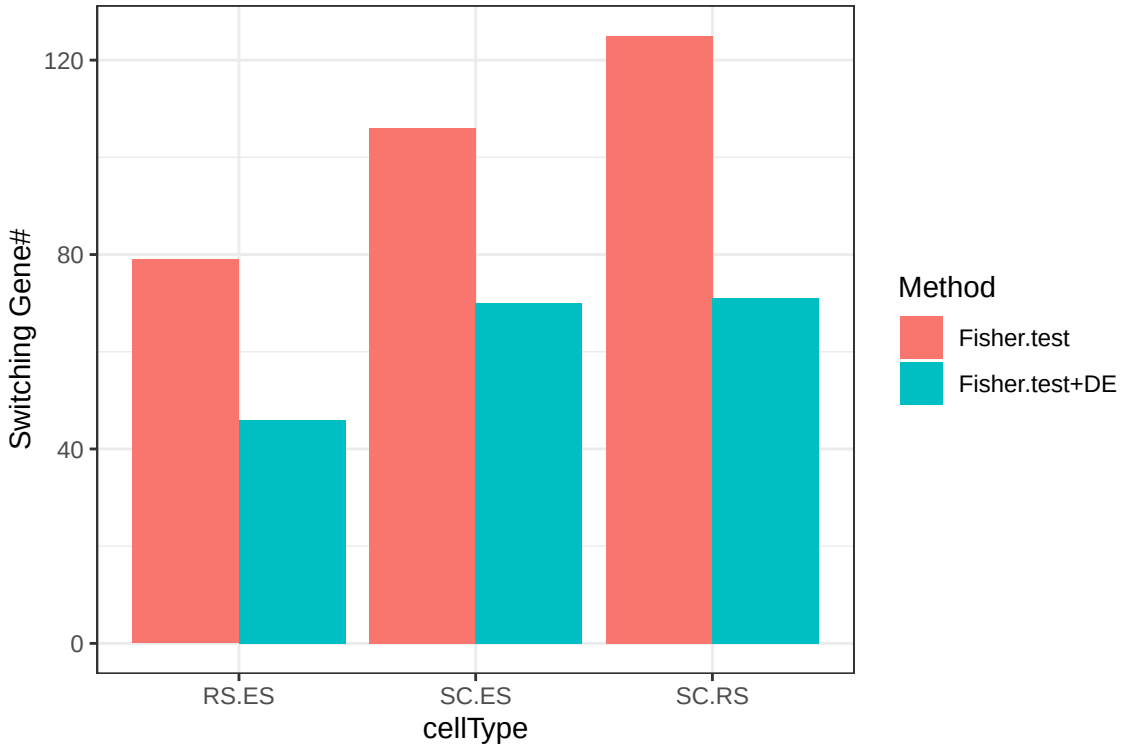
```
swDEq=movAPAswitch(PACds=scPACds, group='celltype',
                   avgPACtag=0, avgGeneTag=0,
                   only3UTR=TRUE, mergeReps='pool',
                   aMovDEPACRes=DEqPAC, DEPAC.padjThd=0.05, nDEPAC=1,
                   mindist=50, fisherThd=0.05, logFCThd=1,
                   cross=FALSE, selectOne='fisherPV')
#> SC.RS
#> SC.ES
#> RS.ES
```

Get 3'UTR switching results.

```
swDEqStat=movStat(object=swDEq, padjThd=0.01, valueThd=1, upThd=NULL, dnThd=NULL)
#> All cond pairs in heat@colData, get de01 and deNum
swDEqStat$nsig
#>       sig.num
#> SC.RS      71
#> SC.ES      70
#> RS.ES      46
```

Compare results from the two 3'UTR switching methods.

```
nsig=as.data.frame(cbind(swstat$nsig, swDEqStat$nsig))
colnames(nsig)=c('Fisher.test', 'Fisher.test+DE')
nsig$cellType=rownames(nsig)
nsig=melt(nsig, variable.name='Method', value.name="SwitchingEvents")
ggplot(data=nsig, aes(x=cellType, y=SwitchingEvents, fill=Method)) +
  geom_bar(stat="identity", position=position_dodge()) +
  ylab("Switching Gene#") + theme_bw()
```



Plot heatmap to view switching status of each gene among all cell types. First convert the *movRes* object to a heatmap object.

```
heat=movRes2heatmapResults(swDEq)
```

Filter switching genes.

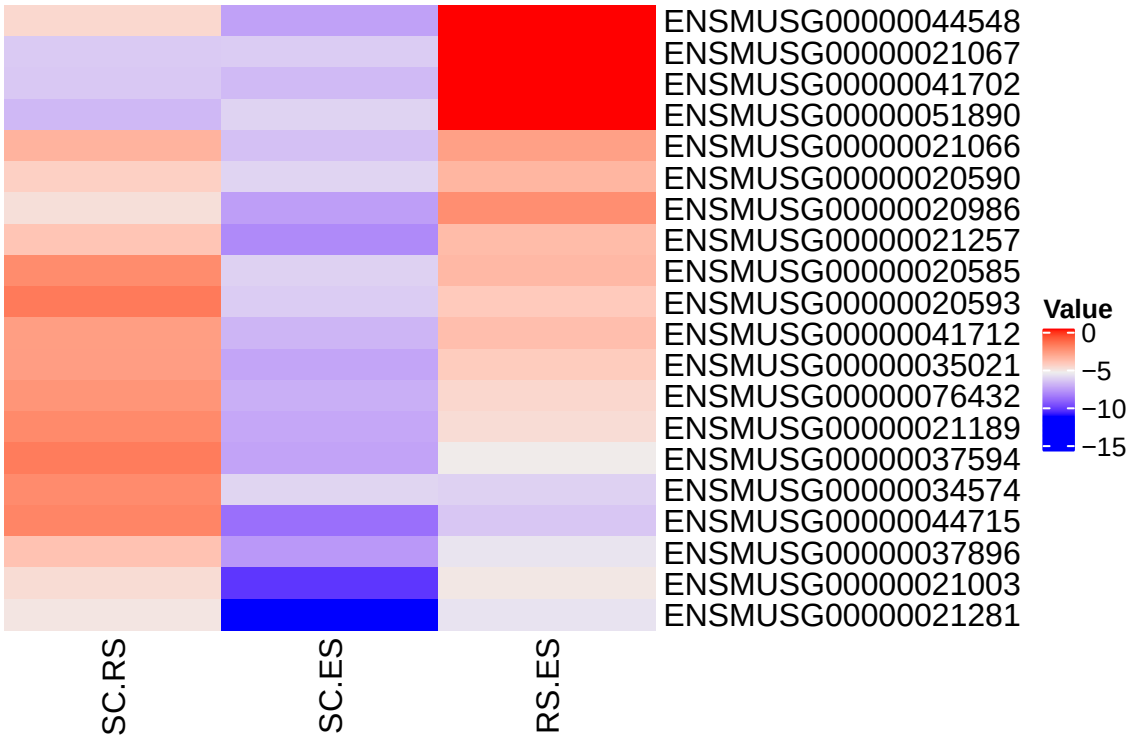
```
heat=subsetHeatmap(heat, padjThd=0.001, valueThd=2)
nrow(heat@value)
#> [1] 79
```

Select top 20 genes for the plot.

```
heat@value=heat@value[rowSums(is.na(heat@value))==0, ]
heat@value=heat@value[order(rowMeans(abs(heat@value)), decreasing =T ), ]
nrow(heat@value)
#> [1] 79
```

From the heatmap, we can see gene ENSMUSG00000021281 is shorter from SC to RS (value=-5), from SC to ES (value=-14), and from RS to ES (value=-5). This means that the length of 3'UTR of this gene is ES < RS < SC, which is consistent with the 3UTR shortening during sperm cell differentiation found in the previous study.

```
plotHeatmap(heat@value[1:20, ],  show_rownames=TRUE, plotPre=NULL)
```




```
heat@value['ENSMUSG00000021281',]
#>                         SC.RS     SC.ES     RS.ES
#> ENSMUSG00000021281 -5.161861 -10.87117 -5.709308
```

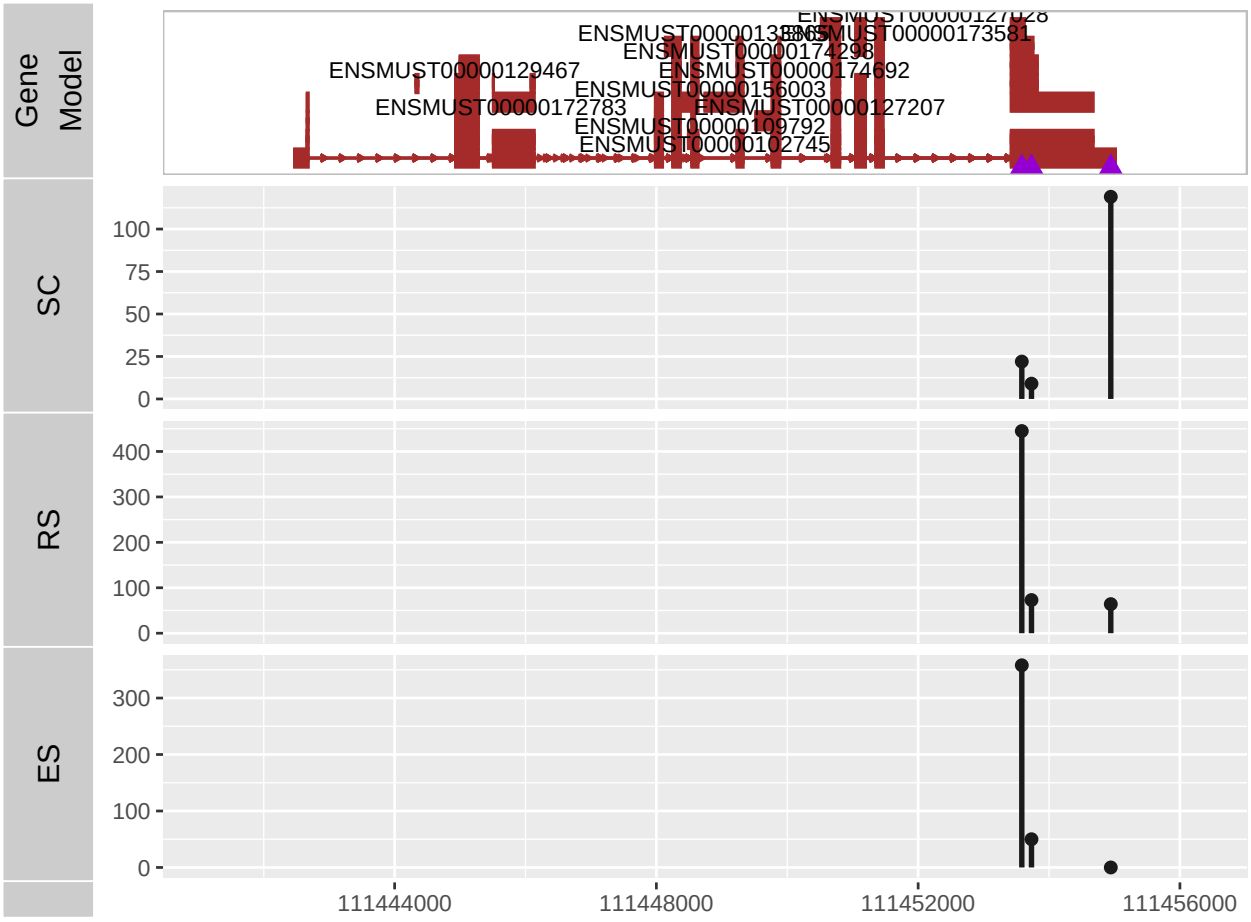
Get the APA switching list between SC and RS for this gene. This gene has three pairs of switching APA sites between SC and RS. But because we set selectOne='fisherPV' in the movAPAswitch function, only the pair with smallest pvalue was returned.

```
swDEq@fullList$SC.RS[swDEq@fullList$SC.RS$gene=='ENSMUSG00000021281',]
#>                   gene nPAC geneTag1 geneTag2 avgUTRlen1 avgUTRlen2     fisherPV
#> 36 ENSMUSG00000021281    2      141      509   1169.801    193.002 8.388516e-59
#>        logFC change    PA1    PA2 dist nDEPA nSwitchPair                  PAs1
#> 36 -5.161861     -1 PA3541 PA3543 1361     2           3 PA3541=22;PA3543=119
#>                    PAs2
#> 36 PA3541=445;PA3543=64
```

5.4 Visualization of 3'UTR switching genes

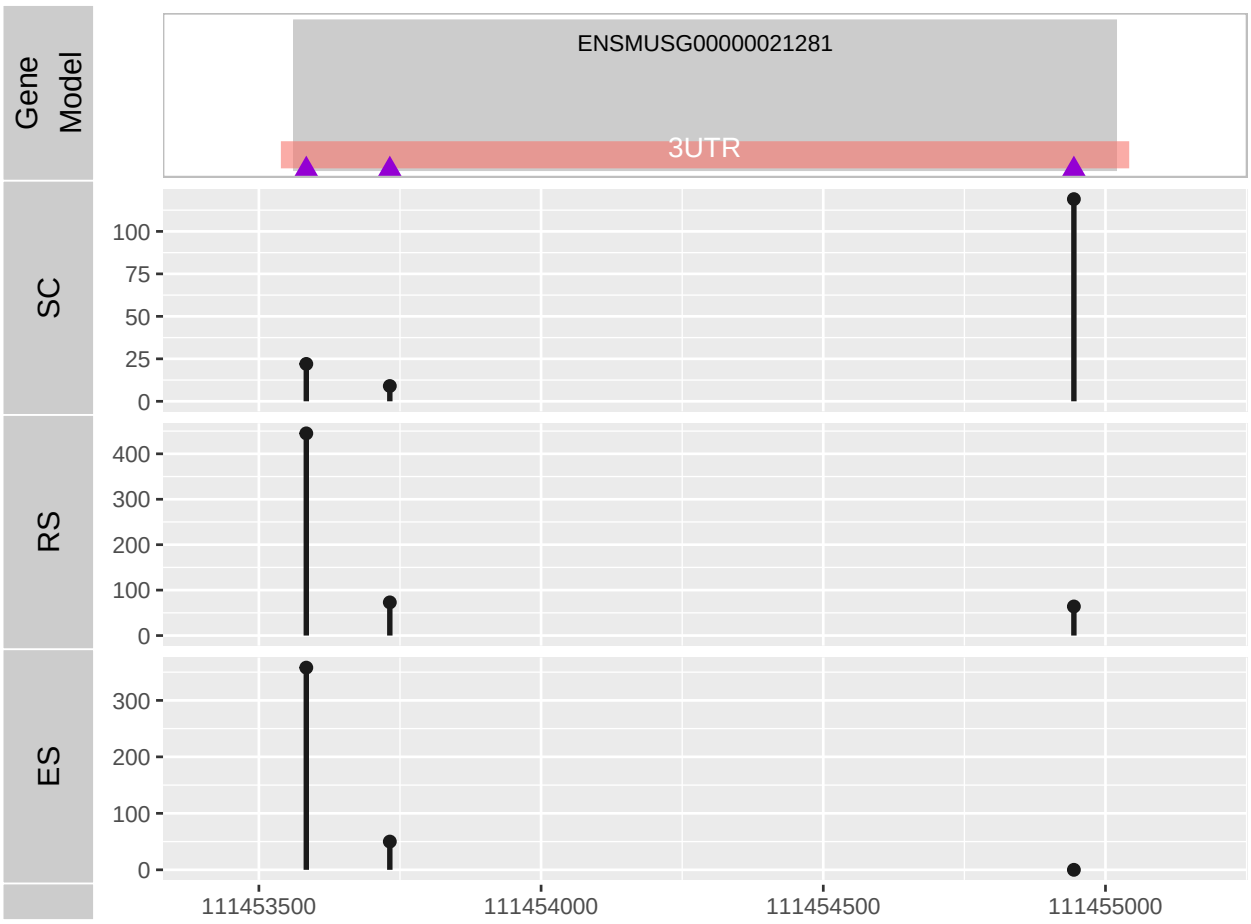
Use *movViz* to show this gene which has three 3'UTR PACs.

```
gene='ENSMUSG00000021281'
movViz(object=swDEq, gene=gene, txdb=gff, PACds=scPACdsCt)
```



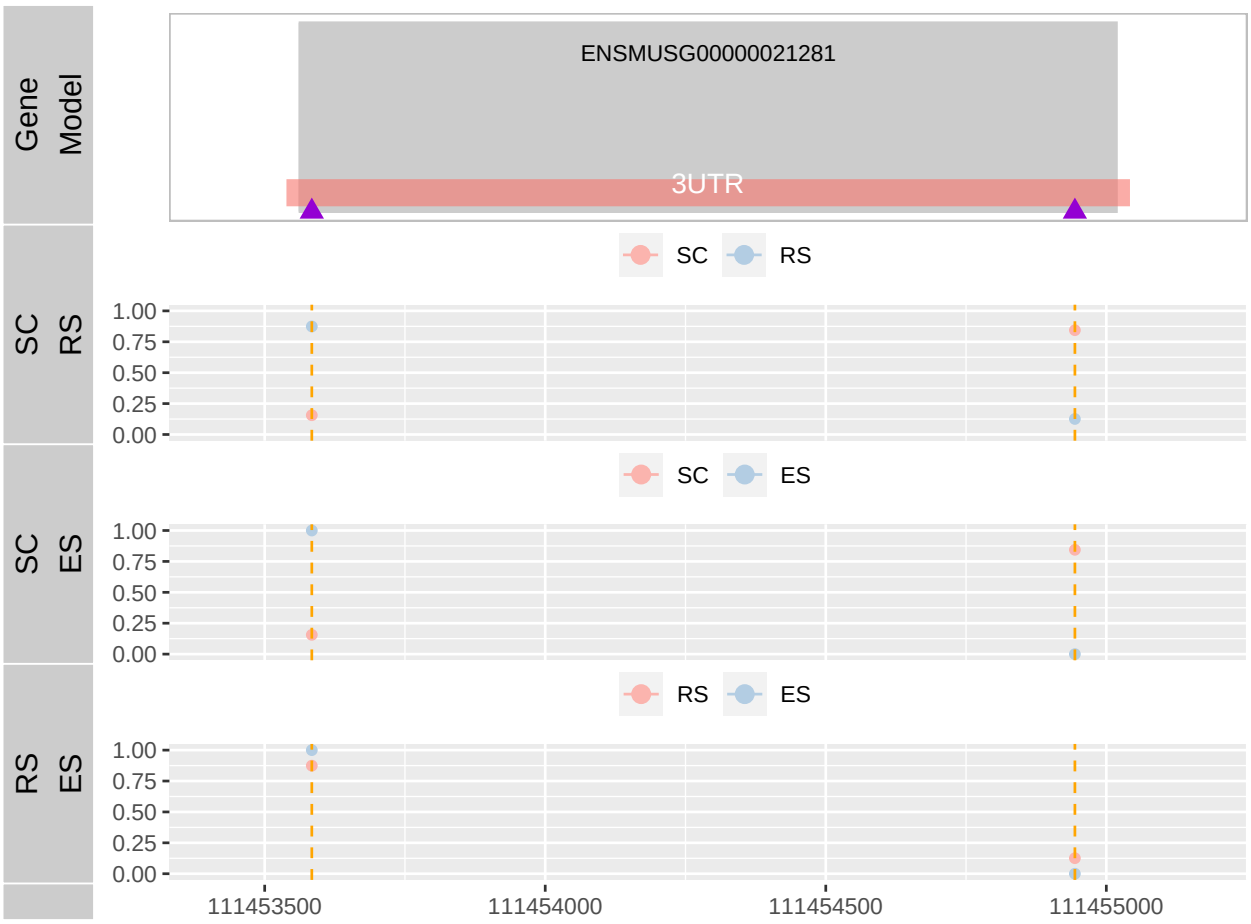
Just show the 3'UTR region.

```
movViz(object=swDEq, gene=gene, txdb=NULL, PACds=scPACdsCt)
```



Show only PACs involved in the 3'UTR switching.

```
movViz(object=swDEq, gene=gene, txdb=NULL, PACds=scPACdsCt, collapseConds=TRUE,
       conds=NULL, highlightConds=NULL, showRatio=TRUE, linkPAs=TRUE,
       padjThd=0.01, showAllPA=FALSE, showPV=FALSE)
```



5.5 Proximal PAC's GPI index

Here we calculate GPI index of each APA gene for each cell. GPI of a gene is the "geo"" score of the proximal poly(A) site. The "geo" metric measures the usage of a poly(A) site by the geometric mean, which was used for measuring poly(A) site usage in single cells (Shulman et al, 2019). First, filter 3'UTR's proximal and distal PACs.

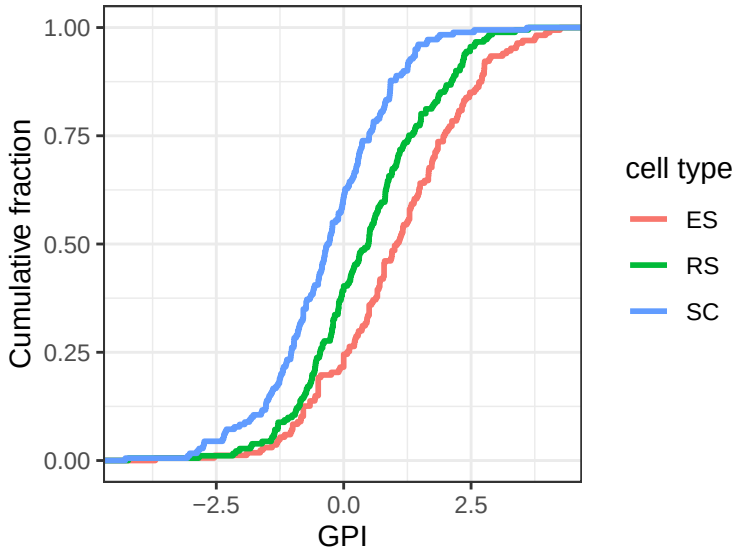
```
ds=get3UTRAPAds(scPACds, sortPA=TRUE, choose2PA='PD')
gpi=movAPAindex(ds, method="GPI")
head(gpi[1:5, 1:5])
gpi=gpi[rowSums(is.na(gpi))==0, ]
```

Calculate GPI for each cell type.

```
ds2=subsetPACds(scPACdsCt, group='celltype', pool=TRUE)
ds2=get3UTRAPAds(ds2, sortPA=TRUE, choose2PA='PD')
gpi2=movAPAindex(ds2, method="GPI")
summary(gpi2)
```

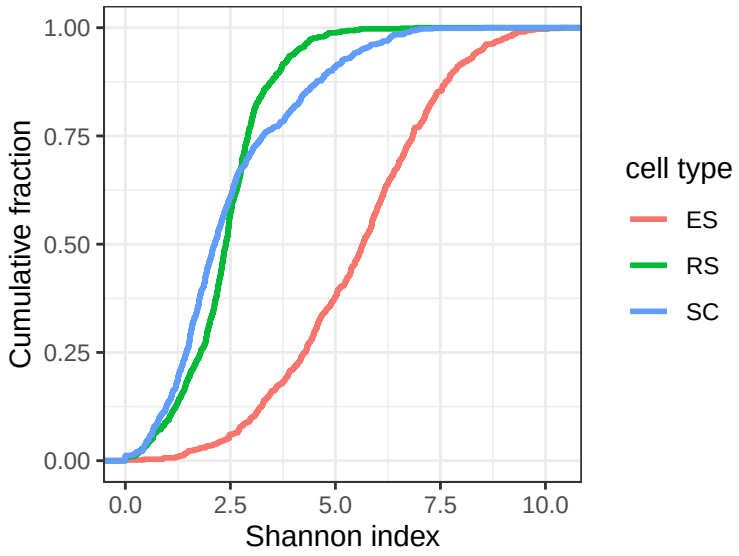
The plot of the distribution of GPI values is similar to the Fig. 3D of Shulman et al, 2019.

```
plotCummPAindex(PAindex=gpi2, groupName='cell type', xlab='GPI')
```



The plot of the distribution of Shannon index (tissue-specificity) for each PAC.

```
shan=movPAindex(scPACdsCt, method="shan")
#> Using count for Shannon.
#> Tissue-specific PAC's H_cutoff (mean-2*sd):  0.2826073
#> Tissue-specific PAC's Q_cutoff (mean-2*sd):  0.2489532
#> Tissue-specific PAC# (H<H_cutoff):  33
#> Tissue-specific PAC# (Q<Q_cutoff):  24
#> Constitutive PAC's H_cutoff (mean+2*sd):  1.585916
#> Constitutive PAC's Q_cutoff (mean+2*sd):  2.726938
#> Constitutive PACs (H>H_cutoff):  0
#> Constitutive PACs (Q>Q_cutoff):  8
plotCummPAindex(PAindex=shan[, -c(1:3)],
                groupName='cell type', xlab='Shannon index')
```



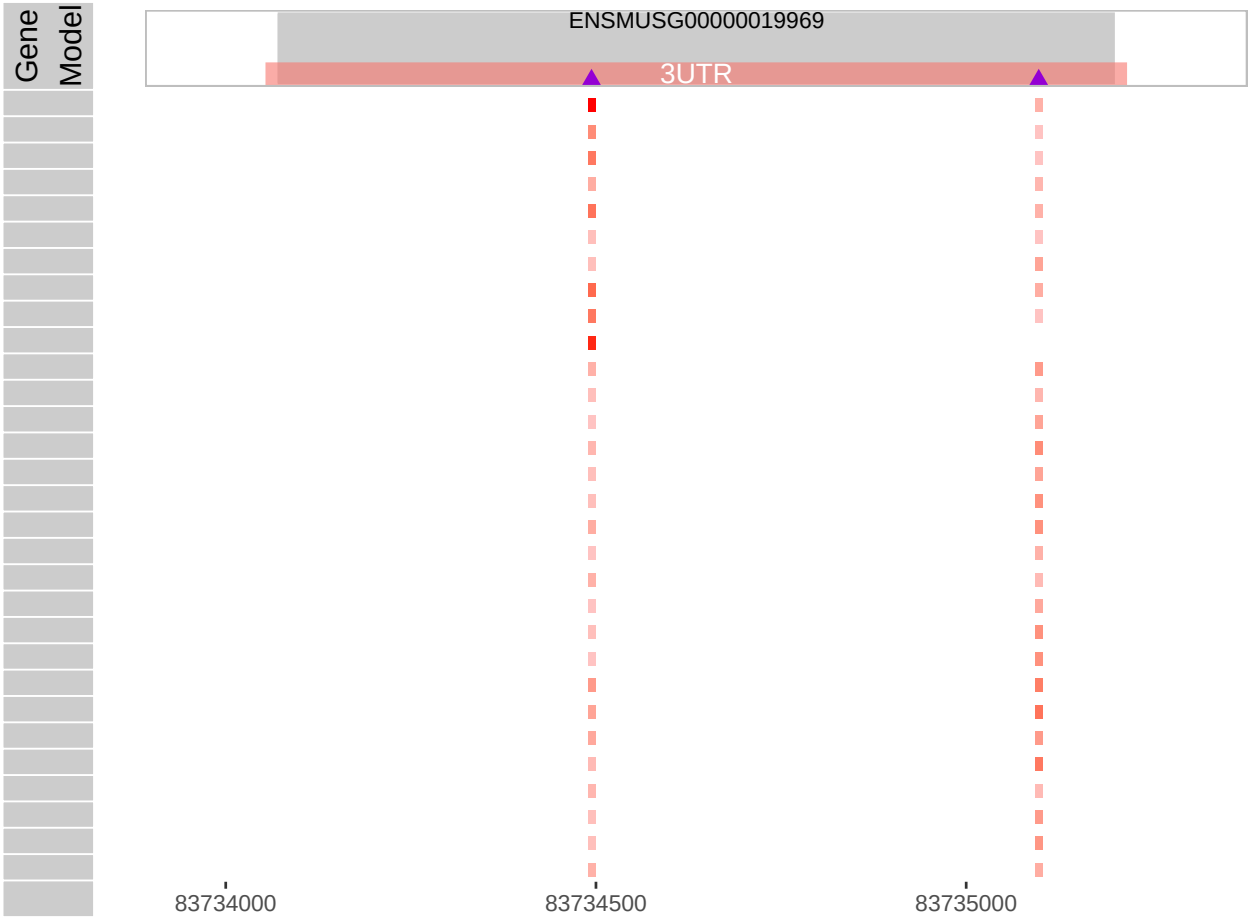
6 Visualize PACs in single cells

Here we first used the *movViz* function to visualize expression levels of PACs among single cells. For each cell type, first we chose top 10 cells with the highest number of read counts from each cell type for the plot.

```
cellCounts=colSums(scPACds@counts)
cellCounts=as.data.frame(cbind(cell=colnames(scPACds@counts),
                               celltype=as.character(scPACds@colData$celltype),
                               count=cellCounts))
cellCounts$count=as.integer(cellCounts$count)
head(cellCounts)
#>                              cell celltype count
#> AAACCTGAGAGGGCTT AAACCTGAGAGGGCTT       SC   743
#> AAACCTGAGCTTATCG AAACCTGAGCTTATCG       RS   664
#> AAACCTGCATACGCCG AAACCTGCATACGCCG       RS  1744
#> AAACCTGGTTGAGTTC AAACCTGGTTGAGTTC       RS   548
#> AAACCTGTCAACGAAA AAACCTGTCAACGAAA       RS   669
#> AAACCTGTCGCGGATC AAACCTGTCGCGGATC       ES   232
shownCells=cellCounts %>% dplyr::group_by(celltype) %>%
  dplyr::filter(rank(dplyr::desc(count)) <=10)
shownCells=shownCells[order(shownCells$celltype), ]
unique(shownCells$celltype)
#> [1] "ES" "RS" "SC"
shownCells=shownCells$cell
```

For single cells, we simplified the plot by setting simple=TRUE, geneHeight=0.1, etc. The first 10 cells are ES, then RS, the last 10 cells are SC. We can see from the plot that the proximal PAC seems to be expressed higher in ES.

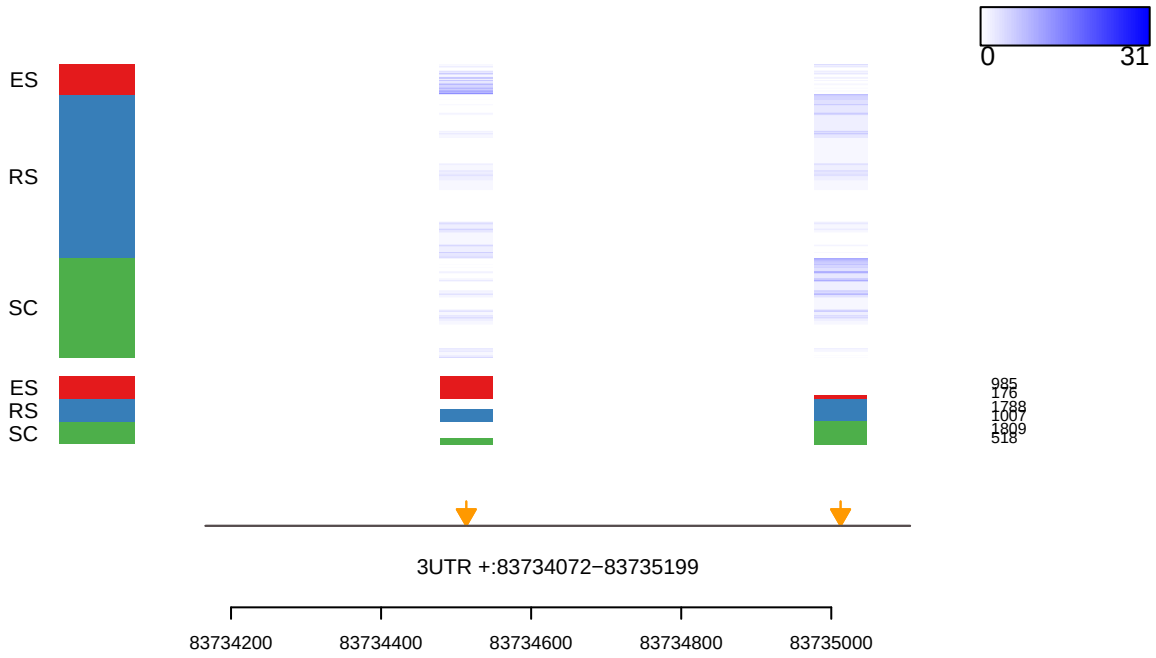
```
gene='ENSMUSG00000019969'
movViz(scPACds[, shownCells], gene=gene, txdb=NULL,
       group='group', collapseConds=FALSE,
       simple=TRUE, showYaxis=FALSE, geneHeight=0.1,
       trackOrder=shownCells, ylimits=c(NA,NA),
       showBox=TRUE, boxWidth=10, showRatio=F, title=NULL)
#> Rectangle color scale is 0 31
#> Ordering tracks by trackOrder...
```



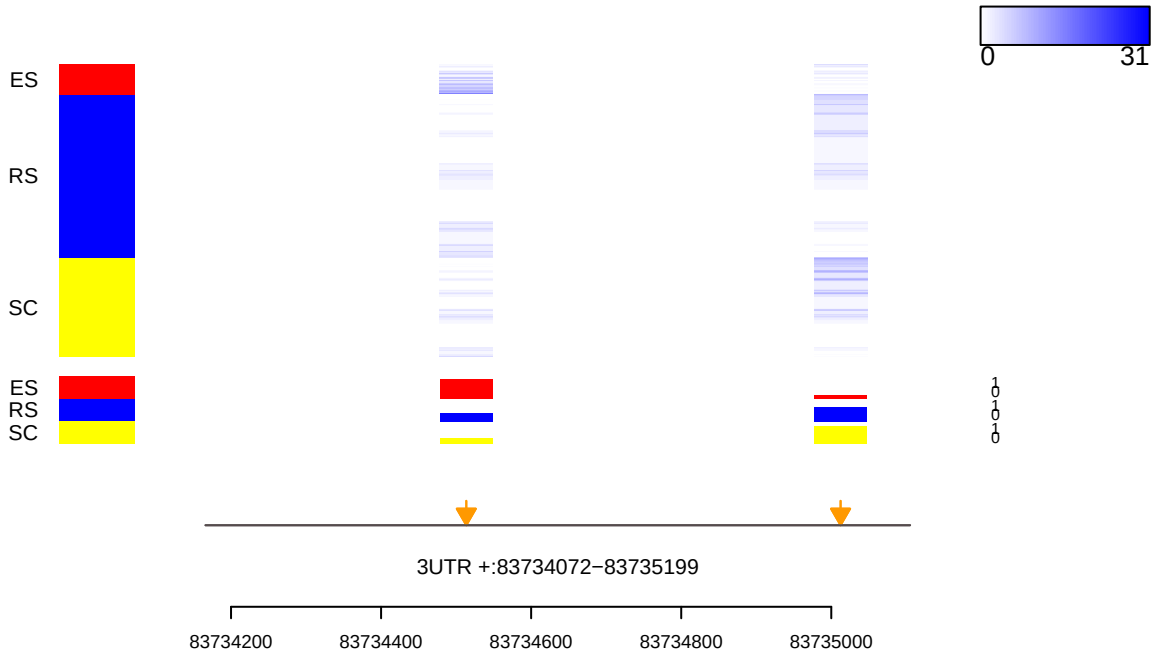
We can also use *movVizSC* function which utilizes the R packages millefy (Ozaki et al., 2020) for a more complex plot.

```
gene='ENSMUSG00000019969'

movVizSC(scPACds, gene, cellGroupName='celltype', txdb=NULL,
         cellGroupColors=NULL, showRatio = F)
#> [1] "Begin Plot: 2020-07-22 20:29:15"
#> There was a problem when running diffusion map. Trying PCA instead...
#> The standard deviations of PC1: 1.091806
```




```
## Plot the expression levels of cell types as ratios, and specify colors for cell types.
movVizSC(scPACds, gene, cellGroupName='celltype', txdb=NULL,
         cellGroupColors=c(SC="yellow", ES="red",RS="blue"), showRatio = T)
#> [1] "Begin Plot: 2020-07-22 20:29:16"
#> There was a problem when running diffusion map. Trying PCA instead...
#> The standard deviations of PC1: 1.091806
```



7 Session Information

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

```
sessionInfo()
#> R version 4.0.0 (2020-04-24)
#> Platform: x86_64-pc-linux-gnu (64-bit)
#> Running under: Ubuntu 16.04.5 LTS
#>
#> Matrix products: default
#> BLAS:   /usr/lib/atlas-base/atlas/libblas.so.3.0
#> LAPACK: /usr/lib/atlas-base/atlas/liblapack.so.3.0
#>
#> locale:
#>  [1] LC_CTYPE=en_GB.UTF-8       LC_NUMERIC=C
#>  [3] LC_TIME=zh_CN.UTF-8        LC_COLLATE=C
#>  [5] LC_MONETARY=zh_CN.UTF-8    LC_MESSAGES=en_GB.UTF-8
#>  [7] LC_PAPER=zh_CN.UTF-8       LC_NAME=C
#>  [9] LC_ADDRESS=C               LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=zh_CN.UTF-8 LC_IDENTIFICATION=C
#>
#> attached base packages:
#>  [1] grid      stats4    parallel  stats     graphics  grDevices utils
#>  [8] datasets  methods   base
#>
#> other attached packages:
#>  [1] movAPA_0.1.0                devtools_2.3.0
#>  [3] usethis_1.6.1               edgeR_3.30.3
#>  [5] limma_3.44.3                millefy_0.1.9
#>  [7] ComplexHeatmap_2.4.2        motifStack_1.32.0
#>  [9] ade4_1.7-15                 MotIV_1.43.0
#> [11] grImport2_0.2-0             DEXSeq_1.34.1
#> [13] BiocParallel_1.22.0         DESeq2_1.28.1
#> [15] SummarizedExperiment_1.18.1 DelayedArray_0.14.0
#> [17] matrixStats_0.56.0          GenomicFeatures_1.40.0
#> [19] AnnotationDbi_1.50.0        Biobase_2.48.0
#> [21] ggbio_1.36.0                BSgenome_1.56.0
#> [23] rtracklayer_1.48.0          Biostrings_2.56.0
#> [25] XVector_0.28.0              ggplot2_3.3.1
#> [27] data.table_1.12.8           RColorBrewer_1.1-2
#> [29] GenomicRanges_1.40.0        GenomeInfoDb_1.24.2
#> [31] IRanges_2.22.2              S4Vectors_0.26.1
#> [33] BiocGenerics_0.34.0         reshape2_1.4.4
#> [35] dplyr_1.0.0
#>
#> loaded via a namespace (and not attached):
#>   [1] rappdirs_0.3.1               ggthemes_4.2.0
```

```
#>   [3] GGally_2.0.0                tidyr_1.1.0
#>   [5] acepack_1.4.1               bit64_0.9-7
#>   [7] knitr_1.28                  irlba_2.3.3
#>   [9] rpart_4.1-15                hwriter_1.3.2
#>  [11] RCurl_1.98-1.2              AnnotationFilter_1.12.0
#>  [13] generics_0.0.2              callr_3.4.3
#>  [15] RSQLite_2.2.0               proxy_0.4-24
#>  [17] bit_1.1-15.2                assertthat_0.2.1
#>  [19] xfun_0.14                   hms_0.5.3
#>  [21] evaluate_0.14               DEoptimR_1.0-8
#>  [23] fansi_0.4.1                 progress_1.2.2
#>  [25] dbplyr_1.4.4                readxl_1.3.1
#>  [27] DBI_1.1.0                   geneplotter_1.66.0
#>  [29] htmlwidgets_1.5.1           reshape_0.8.8
#>  [31] purrr_0.3.4                 ellipsis_0.3.1
#>  [33] RSpectra_0.16-0             backports_1.1.8
#>  [35] annotate_1.66.0             biomaRt_2.44.0
#>  [37] vctrs_0.3.1                 SingleCellExperiment_1.10.1
#>  [39] remotes_2.1.1               ensembldb_2.12.1
#>  [41] TTR_0.23-6                  abind_1.4-5
#>  [43] RcppEigen_0.3.3.7.0         withr_2.2.0
#>  [45] robustbase_0.93-6           checkmate_2.0.0
#>  [47] vcd_1.4-7                   GenomicAlignments_1.24.0
#>  [49] rGADEM_2.36.0               xts_0.12-0
#>  [51] prettyunits_1.1.1           cluster_2.1.0
#>  [53] lazyeval_0.2.2              seqLogo_1.54.3
#>  [55] laeken_0.5.1                crayon_1.3.4
#>  [57] genefilter_1.70.0           pkgconfig_2.0.3
#>  [59] labeling_0.3                pkgload_1.1.0
#>  [61] ProtGenerics_1.20.0         nnet_7.3-14
#>  [63] rlang_0.4.6                 lifecycle_0.2.0
#>  [65] BiocFileCache_1.12.0        dichromat_2.0-0
#>  [67] cellranger_1.1.0            rprojroot_1.3-2
#>  [69] RcppHNSW_0.2.0              lmtest_0.9-37
#>  [71] graph_1.66.0                Matrix_1.2-18
#>  [73] carData_3.0-4               boot_1.3-25
#>  [75] zoo_1.8-8                   base64enc_0.1-3
#>  [77] GlobalOptions_0.1.2         processx_3.4.2
#>  [79] png_0.1-7                   rjson_0.2.20
#>  [81] bitops_1.0-6                blob_1.2.1
#>  [83] shape_1.4.4                 stringr_1.4.0
#>  [85] jpeg_0.1-8.1                scales_1.1.1
#>  [87] memoise_1.1.0               magrittr_1.5
#>  [89] plyr_1.8.6                  hexbin_1.28.1
#>  [91] zlibbioc_1.34.0             compiler_4.0.0
#>  [93] pcaMethods_1.80.0           clue_0.3-57
#>  [95] Rsamtools_2.4.0             cli_2.0.2
```

```
#>  [97] ps_1.3.3                    htmlTable_2.0.1
#>  [99] Formula_1.2-3               ggplot.multistats_1.0.0
#> [101] MASS_7.3-51.6               tidyselect_1.1.0
#> [103] stringi_1.4.6               forcats_0.5.0
#> [105] highr_0.8                   yaml_2.2.1
#> [107] askpass_1.1                 locfit_1.5-9.4
#> [109] latticeExtra_0.6-29         VariantAnnotation_1.34.0
#> [111] tools_4.0.0                 rio_0.5.16
#> [113] circlize_0.4.10             rstudioapi_0.11
#> [115] foreign_0.8-76              gridExtra_2.3
#> [117] smoother_1.1                farver_2.0.3
#> [119] scatterplot3d_0.3-41        digest_0.6.25
#> [121] BiocManager_1.30.10         Rcpp_1.0.4.6
#> [123] car_3.0-8                   OrganismDbi_1.30.0
#> [125] httr_1.4.1                  biovizBase_1.36.0
#> [127] colorspace_1.4-1            XML_3.99-0.3
#> [129] fs_1.4.1                    ranger_0.12.1
#> [131] splines_4.0.0               RBGL_1.64.0
#> [133] statmod_1.4.34              sp_1.4-2
#> [135] sessioninfo_1.1.1           xtable_1.8-4
#> [137] testthat_2.3.2              destiny_3.2.0
#> [139] R6_2.4.1                    Hmisc_4.4-0
#> [141] pillar_1.4.4                htmltools_0.5.0
#> [143] glue_1.4.1                  VIM_6.0.0
#> [145] class_7.3-17                codetools_0.2-16
#> [147] pkgbuild_1.0.8              lattice_0.20-41
#> [149] tibble_3.0.1                curl_4.3
#> [151] zip_2.0.4                   openxlsx_4.1.5
#> [153] openssl_1.4.1               survival_3.1-12
#> [155] rmarkdown_2.3               desc_1.2.0
#> [157] munsell_0.5.0               e1071_1.7-3
#> [159] GetoptLong_1.0.2            GenomeInfoDbData_1.2.3
#> [161] haven_2.3.1                 gtable_0.3.0
```

8 References

[1] 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.

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

[3] 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.

[4] Ozaki, H., Hayashi, T., Umeda, M. et al. Millefy: visualizing cell-to-cell heterogeneity in read coverage of single-cell RNA sequencing datasets. BMC Genomics 21, 177 (2020).