Analyze APA results from scAPAtrap with the movAPA package

Xiaohui Wu

Last modified 2023-10-10

Contents

1 Overview

This documentation describes how to read an external file of single-cell poly(A) sites generated by scAPAtrap and analyze it with movAPA.

Actually, PAC list with columns chr/strand/coord and counts can be easily converted as `PACdataset` and loaded into movAPA by `movAPA::createPACdataset` or `readPACds`.

In this document, "PA, PAC, or pA" all are short for a poly(A) site.

2 Data read and filtering

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.

2.1 Demo data

The demo_peaks dataset in the movAPA package contains 5w peaks in 2538 cells, including two variables `peaks.meta` and `peaks.demo`. This file was obtained by scAPAtrap.

```
library(movAPA)
#>
#>     'movAPA'
#> The following object is masked from 'package:base':
#>
#>     rbind
# load demo peaks generated by scAPAtrap,
# which contains a list(peaks.meta, peaks.count)
data("demo_peaks")
peaks=demo_peaks
names(peaks)
#> [1] "peaks.meta"  "peaks.count"
head(peaks$peaks.meta)
#>                    peakID   chr     start       end strand     coord
#> peak_83012     peak_83012  chr1 237148255 237148422      + 237148422
#> peak_627325   peak_627325 chr10 104147771 104148105      + 104148105
#> peak_1566732 peak_1566732  chr8  18856854  18856917      -  18856854
#> peak_983111   peak_983111 chr20  25228760  25229062      +  25229062
#> peak_2087967 peak_2087967  chrX  48698679  48698884      -  48698679
#> peak_87658     peak_87658  chr1 248867076 248867152      + 248867152
head(peaks$peaks.count[, 1:10])
#> 6 x 10 sparse Matrix of class "dgCMatrix"
#>   [[ suppressing 10 column names 'AAACCCACAAATTGCC', 'AAACCCAGTCAATGGG', 'AAACCCATCTTAGCAG' ... ]]
#>
#> peak_83012   . . . . . .
#> peak_627325  . . . . . .
#> peak_1566732 . . . . . .
#> peak_983111  . . . . . .
#> peak_2087967 . . . . . .
#> peak_87658   . . . . . .
```

Then we can create a PACdataset object.

```
PACds=createPACdataset(counts=peaks$peaks.count, anno=peaks$peaks.meta)
PACds
#> PAC# 50000
#> sample# 2538
#> AAACCCACAAATTGCC AAACCCAGTCAATGGG AAACCCATCTTAGCAG AAACGAAAGATTGAGT AAACGCTAGGAGTCTG ...
#> groups:
```

```
#> @colData...[2538 x 1]
#>                     group
#> AAACCCACAAATTGCC group1
#> AAACCCAGTCAATGGG group1
#> @counts...[50000 x 2538]
#> 2 x 10 sparse Matrix of class "dgCMatrix"
#>   [[ suppressing 10 column names 'AAACCCACAAATTGCC', 'AAACCCAGTCAATGGG', 'AAACCCATCTTAGCAG' ... ]]
#>
#> peak_83012  . . . . . .
#> peak_627325 . . . . . .
#> @colData...[2538 x 1]
#>                     group
#> AAACCCACAAATTGCC group1
#> AAACCCAGTCAATGGG group1
#> @anno...[50000 x 6]
#>                   peakID   chr     start       end strand     coord
#> peak_83012   peak_83012  chr1 237148255 237148422      + 237148422
#> peak_627325 peak_627325 chr10 104147771 104148105      + 104148105
rm(peaks)
```

2.2 Data filtering

This dataset contains many PAs (also called peaks in scAPAtrap), which may not be suitable for downstream analysis. We can first remove extremely lowly expressed peaks.

First, we make summary of the PACdataset. It seems that >50% of PAs with less than 2 reads.

```
summary(PACds)
#> PAC# 50000
#> sample# 2538
#> summary of expression level of each PA
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#>     1.00     1.00     2.00    11.63     7.00 16419.00
#> summary of expressed sample# of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    1.00    1.00    2.00   10.03    6.00 1944.00
```

Here we remove PAs with <10 tags supported by all cells and PAs that are expressed in <10 cells. This filtering removed ~80% PAs.

```
PACds=subsetPACds(PACds, totPACtag = 10, minExprConds = 10, verbose=TRUE)
#>                       count
#> before subsetPACds 50000
#> totPACtag>=10       9449
#> minExprConds>=10    9141
summary(PACds)
#> PAC# 9141
#> sample# 2538
#> summary of expression level of each PA
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#>    10.00    14.00    22.00    51.09    43.00 16419.00
#> summary of expressed sample# of each PA
```

```
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   10.00   13.00   21.00   42.79   40.00 1944.00
```

3 Validate PA

3.1 Remove internal priming artifacts

After read the data into a PACdataset, users can use many functions in movAPA for removing internal priming artifacts, annotating PACs, polyA signal analysis, etc. Please follow the vignette of "movAPA_on_rice_tissues" for more details.

For example, users can remove internal priming artifacts with `removePACdsIP`. Before starting, it is better to check the chromosome names are consistent between the `BSgenome` and `PACdataset`. We need make sure the chromosome name of your PAC data is the same as the BSgenome.

```
library(BSgenome.Hsapiens.UCSC.hg38, quietly = TRUE, verbose = FALSE)
bsgenome<-BSgenome.Hsapiens.UCSC.hg38

head(GenomeInfoDb::seqnames(bsgenome))
#> [1] "chr1" "chr2" "chr3" "chr4" "chr5" "chr6"
head(unique(PACds@anno$chr))
#> [1] "chr5"  "chr17" "chr2"  "chr1"  "chr14" "chr11"
```

```
PACdsIP=removePACdsIP(PACds, bsgenome, returnBoth=TRUE,
                      up=-10, dn=10, conA=6, sepA=7)
#> 5013 IP PACs; 4128 real PACs
length(PACdsIP$real)
#> [1] 4128
length(PACdsIP$ip)
#> [1] 5013

# summary of IPs and real PAs
summary(PACdsIP$real)
#> PAC# 4128
#> sample# 2538
#> summary of expression level of each PA
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#>    10.00    13.75    21.00    53.94    38.00 16419.00
#> summary of expressed sample# of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   10.00   13.00   20.00   42.51   36.00 1944.00

summary(PACdsIP$ip)
#> PAC# 5013
#> sample# 2538
#> summary of expression level of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   10.00   15.00   24.00   48.73   46.00 3405.00
#> summary of expressed sample# of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   10.00   14.00   23.00   43.01   43.00 1341.00
```

Nearly half of the PACs are internal priming artifacts. We can use the real PAs for further analysis. **However, removing internal priming is a non-trifle task, which should be done in caution.**

```
# use real PA for further analysis
PACds=PACdsIP$real
```

3.2 Remove internal priming using reference PAs

Another way to remove internal priming (IP) artifacts is to use known PAs. We can retain those IP sites that supported by known PAs as real. The following example shows how to use known human PAs from PolyA_DB v3 for IP removing.

First we loaded the known PAs.

```
annodb=read.table("Human_hg38.PolyA_DB.bed", sep="\t", header = FALSE)
head(annodb)
annodb=annodb[,c(1,2,6)]
colnames(annodb)=c("chr","coord","strand")
annodb=readPACds(annodb, colDataFile = NULL)
```

Next, we determine the overlap between PACds' IP and known PAs. The PAs of scAPAtrap is represented by peak, with peak start and end. We believe that if a peak overlaps with any reference PA, then the PA of that peak is considered real.

The `findOverapPACds` of movAPA use `UPA_Start` and `UPA_end` as the column names, so here we first modify the start and end column names of peak to UPA_Start and UPA_end.

```
PACdsIP$ip@anno$UPA_start=PACdsIP$ip@anno$start
PACdsIP$ip@anno$UPA_end=PACdsIP$ip@anno$end

PACdsIP$real@anno$UPA_start=PACdsIP$real@anno$start
PACdsIP$real@anno$UPA_end=PACdsIP$real@anno$end

## find overlapping IPs
IP.ovp=findOvpPACds(qryPACds=PACdsIP$ip, sbjPACds=annodb,
                    d=50,
                    qryMode='region', sbjMode='point',
                    returnNonOvp=TRUE)
```

Merge IPs supported by reference PAs and the original real PAs as the PAs for subsequent analysis.

```
anno=rbind(PACdsIP$real@anno, IP.ovp$ovp@anno)
count=rbind(PACdsIP$real@counts, IP.ovp$ovp@counts)
PACds=createPACdataset(counts=count, anno=anno)
summary(PACds)
```

3.3 Annotate genomic regions for PACs

```
library(TxDb.Hsapiens.UCSC.hg38.knownGene, quietly = TRUE, verbose = FALSE)
txdb=TxDb.Hsapiens.UCSC.hg38.knownGene
```

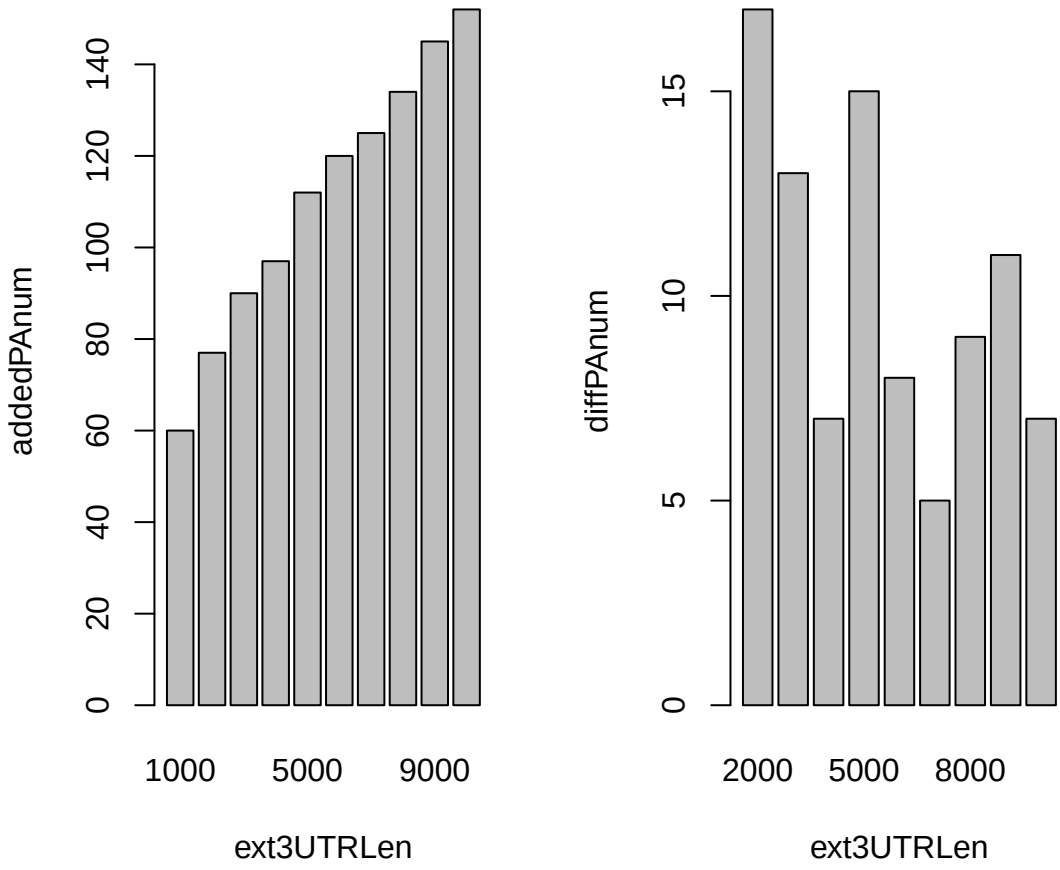
```
# annotate PAs
PACds=annotatePAC(PACds, txdb)
#> Warning: remove PAs in [character(0)] genes 266 rows

# after annotation, the gene and ftr information are present in PACds@anno.
summary(PACds)
#> PAC# 3862
#> sample# 2538
#> summary of expression level of each PA
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#>    10.00    14.00    21.00    55.65    39.00 16419.00
#> summary of expressed sample# of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   10.00   13.00   20.00   43.62   37.00 1944.00
#> gene# 2430
#>            nPAC
#> 3UTR        392
#> 5UTR         13
#> CDS         104
#> exon         31
#> intergenic  307
#> intron     3015
```

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(PACds, seq(1000, 10000, by=1000))
```




```
#>    ext3UTRLen addedPAnum
#> 1        1000         60
#> 2        2000         77
#> 3        3000         90
#> 4        4000         97
#> 5        5000        112
#> 6        6000        120
#> 7        7000        125
#> 8        8000        134
#> 9        9000        145
#> 10      10000        152
```

You can take a look at the length of the 3'UTRs PACds again to make a rough judgment. Here, we only select the length of the 3'UTRs where PA is located for approximate calculation.

```
utrid=which(PACds@anno$ftr=='3UTR')
utrs=unique(PACds@anno[utrid, c('ftr_end','ftr_start')])
summary(utrs$ftr_end-utrs$ftr_start+1)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>      34     559    1802    2682    3775   25446
```

Here we extended 3'UTR length for 2000 bp. After extension, 100+ PACs in intergenic region are now in extended 3'UTRs.

```
# extend 3UTR by 1000bp
PACds=ext3UTRPACds(PACds, 1000)
#> 60 PACs in extended 3UTR (ftr=intergenic >> ftr=3UTR)
#> Get 3UTR length (anno@toStop) for 3UTR/extended 3UTR PACs

# after 3UTR extension
summary(PACds)
#> PAC# 3862
#> sample# 2538
#> summary of expression level of each PA
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#>    10.00    14.00    21.00    55.65    39.00 16419.00
#> summary of expressed sample# of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   10.00   13.00   20.00   43.62   37.00 1944.00
#> gene# 2480
#>            nPAC
#> 3UTR        452
#> 5UTR         13
#> CDS         104
#> exon         31
#> intergenic  247
#> intron     3015
```

3.4 Retain 3'UTR PAs

For single cell data, we suggest analyzing only 3'UTR PAs and discarding PAs from other regions.

```
PACds=PACds[PACds@anno$ftr=='3UTR']
summary(PACds)
#> PAC# 452
#> sample# 2538
#> summary of expression level of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    10.0    20.0    42.0   155.8   130.5  3548.0
#> summary of expressed sample# of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    10.0    19.0    40.0   114.0   120.2  1412.0
#> gene# 440
#>      nPAC
#> 3UTR  452
```

3.5 Base compostions

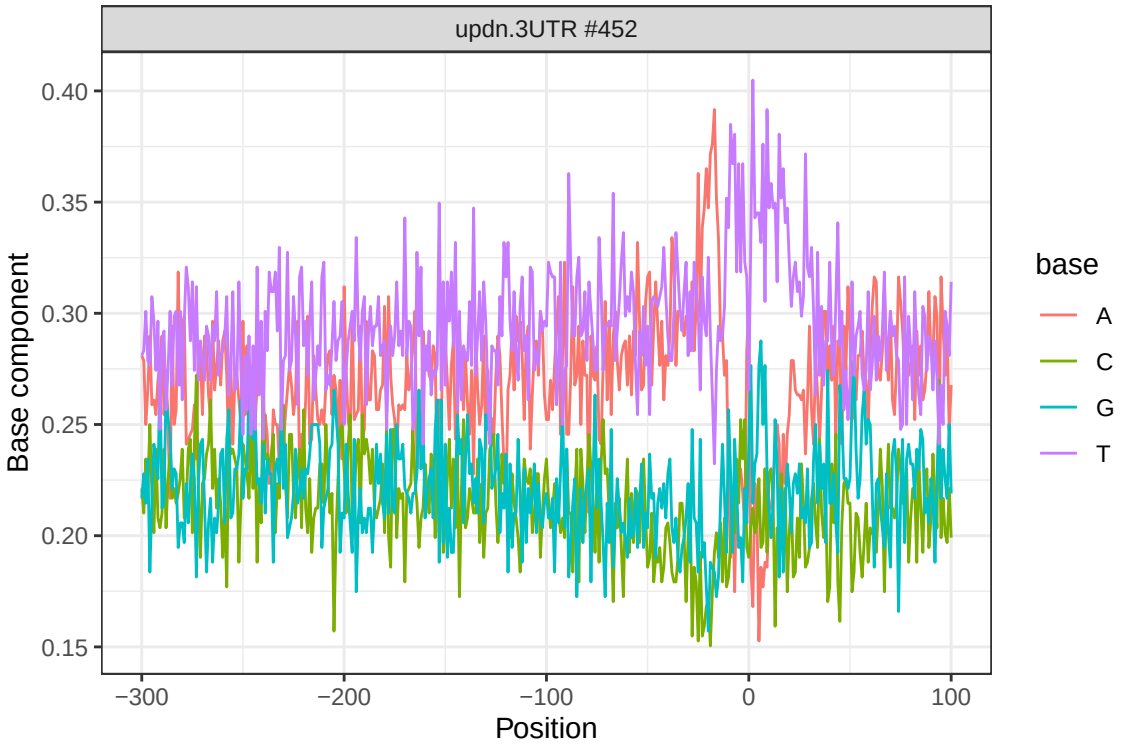
The function *plotATCGforFAfile* is for plotting single nucleotide profiles for given fasta file(s), which is particularly useful for discerning base compositions surrounding PACs.

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

```
faFiles=faFromPACds(PACds, bsgenome, what='updn', fapre='updn',
                    up=-300, dn=100, byGrp='ftr')
#> 452 >>> updn.3UTR.fa
```

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

```
plotATCGforFAfile(faFiles, ofreq=FALSE, opdf=FALSE,
                  refPos=301, mergePlots = TRUE)
```



3.6 PolyA signals

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

Here we show another example to scan known human polyA signals. 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.

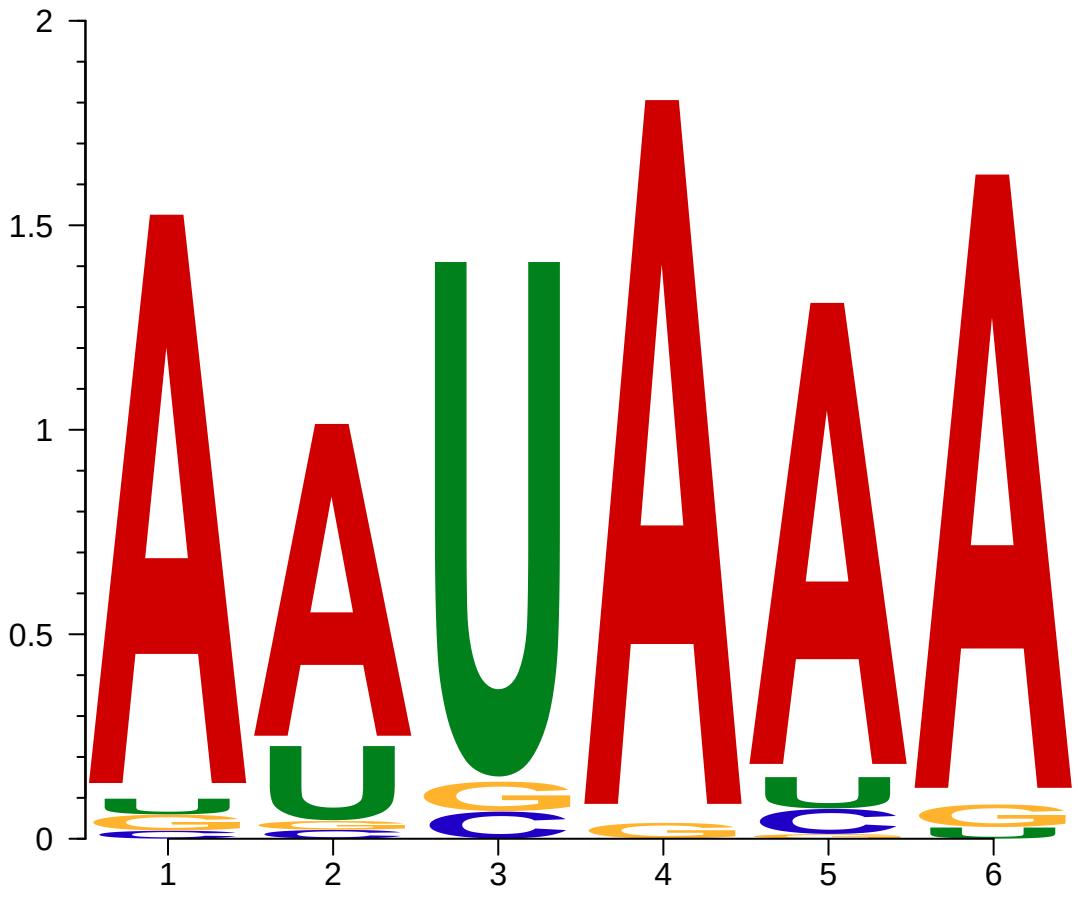
```
PACdsMM=annotateByPAS(PACds, bsgenome, grams=v,
                      priority=priority,
                      from=-50, to=-1, label='mm')
```

```
table(PACdsMM@anno$mm_gram)
#>
#> AACAAA AACAAG AAGAAA AATAAA AATAAG AATAAT AATACA AATAGA AATATA AATGAA ACTAAA
#>      5      6     12    117      2      4      7      2      9      5      5
#> AGTAAA ATTAAA ATTACA ATTATA CATAAA GATAAA TATAAA
#>      5     34      4      5      3      6      6

## percent of PA without PAS
noPAS=sum(is.na(PACdsMM@anno$mm_gram))
noPAS/length(PACdsMM)
#> [1] 0.4756637
```

Plot signal logos.

```
pas=PACdsMM@anno$mm_gram[!is.na(PACdsMM@anno$mm_gram)]
plotSeqLogo(pas)
```



4 Merge multiple samples

4.1 Merge multiple samples

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.

Here we show a multi-sample merging using the reference PA. We directly copy one more PACds for this example.

```
## the two PACds for merging
PACdsList=list(ds1=PACds, ds2=PACds)
```

```
pacds.merge=mergePACds(PACdsList, d=24, by='coord')
#> mergePACds: there are 2538 duplicated sample names in the PACdsList, will add .N to sample names of
#> mergePACds: total 904 redundant PACs from 2 PACds to merge
#> mergePACds without refPACds: 904 separate PACs reduce to 452 PACs (d=24nt)
#> mergePACds: melted all counts tables, total 103012 triplet rows
#> mergePACds: link 904 old PA IDs to 452 new PA IDs by merge
#> mergePACds: convert 103012 triplets to dgCMatrix
#> mergePACds: construct Matrix[PA, sample], [452, 5050]

# summary of PACds#1
summary(PACds)
#> PAC# 452
#> sample# 2538
#> summary of expression level of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    10.0    20.0    42.0   155.8   130.5  3548.0
#> summary of expressed sample# of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    10.0    19.0    40.0   114.0   120.2  1412.0
#> gene# 440
#>      nPAC
#> 3UTR  452

# summary of the merged PACds
summary(pacds.merge)
#> PAC# 452
#> sample# 5050
#> summary of expression level of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    20.0    40.0    84.0   311.5   261.0  7096.0
#> summary of expressed sample# of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    20.0    38.0    80.0   227.9   240.5  2824.0

head(pacds.merge@counts[, 1:10])
#> 6 x 10 sparse Matrix of class "dgCMatrix"
#>   [[ suppressing 10 column names 'AAACCCACAAATTGCC.1', 'AAACCCAGTCAATGGG.1', 'AAACCCATCTTAGCAG.1' ..
#>
#> M1 . . . . . 1 . . .
#> M2 . . . . . .
#> M3 . . . . . 1 .
```

```
#> M4 . . . . . .
#> M5 . . . . . .
#> M6 2 . . . . . .
head(pacds.merge@anno)
#>     chr strand    coord UPA_start  UPA_end
#> M1 chr1      +  2308755   2308755  2308755
#> M2 chr1      +  9368021   9368021  9368021
#> M3 chr1      +  9614872   9614872  9614872
#> M4 chr1      + 20555007  20555007 20555007
#> M5 chr1      + 20618815  20618815 20618815
#> M6 chr1      + 46220370  46220370 46220370
```

4.2 Merge multiple samples with a reference 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.

Given a reference PACds, `buildRefPACdsAnno` can be used to combine multiple PACds to build a more complete reference. First we loaded the known PAs.

```
annodb=read.table("./Human_hg38.PolyA_DB.bed", sep="\t", header = FALSE)
head(annodb)
#>     V1     V2     V3              V4         V5 V6
#> 1 chr1 629219 629219 ENSG00000225972 Pseudogene  +
#> 2 chr1 629249 629249 ENSG00000225972 Pseudogene  +
#> 3 chr1 629284 629284 ENSG00000225972 Pseudogene  +
#> 4 chr1 629328 629328 ENSG00000225972 Pseudogene  +
#> 5 chr1 629572 629572 ENSG00000225972 Pseudogene  +
#> 6 chr1 629626 629626 ENSG00000225972 Pseudogene  +
annodb=annodb[,c(1,2,6)]
colnames(annodb)=c("chr","coord","strand")
annodb=readPACds(annodb, colDataFile = NULL)
#> 155808 PACs
```

```
## we can build a reference PACds with human known PA and the given PACds list
## or we can use annodb only
refPA=buildRefPACdsAnno(refPACds=annodb, PACdsList=PACdsList,
                        by='coord', d=24,
                        min.counts = 50, min.smps=10, max.width=500,
                        verbose=TRUE)
#> ds1: total=452 --> good=0 [>=min.counts(50) & >=min.smps(10)) & <=max.width(500)]
#> ds2: total=452 --> good=0 [>=min.counts(50) & >=min.smps(10)) & <=max.width(500)]
#> buildRefPACds: 0 highQuality PACs (filtered by min.counts=50, min.smps=10, max.width=500)
#> buildRefPACds: 155808 [155808 ref + 0 highQuality PACs] reduce to 149362 PACs (d=24nt)
#> buildRefPACds: 149362 reference PACs returned
```

Because the two PACds merged for this example are the same, the number of PACs remains unchanged after merging, but the sample size doubles.

```r
pacds.merge=mergePACds(PACdsList, d=24, by='coord', refPACds=refPA)
#> mergePACds: there are 2538 duplicated sample names in the PACdsList, will add .N to sample names of
#> mergePACds: total 904 redundant PACs from 2 PACds to merge
#> mergePACds with refPACds: 149726 merged PACs (d=24nt)
#> mergePACds: melted all counts tables, total 103012 triplet rows
#> mergePACds: link 904 old PA IDs to 452 new PA IDs by merge
#> mergePACds: convert 103012 triplets to dgCMatrix
#> mergePACds: construct Matrix[PA, sample], [452, 5050]
#> Warning: in mergePACds, the number of rows (PAs) or rownames after merging is not the same between a
#> This may be because refPACds is used. Will use common PAs between anno and counts (452 PAs)

# summary of PACds#1
summary(PACds)
#> PAC# 452
#> sample# 2538
#> summary of expression level of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    10.0    20.0    42.0   155.8   130.5  3548.0
#> summary of expressed sample# of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    10.0    19.0    40.0   114.0   120.2  1412.0
#> gene# 440
#>      nPAC
#> 3UTR  452

# summary of the merged PACds
summary(pacds.merge)
#> PAC# 452
#> sample# 5050
#> summary of expression level of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    20.0    40.0    84.0   311.5   261.0  7096.0
#> summary of expressed sample# of each PA
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    20.0    38.0    80.0   227.9   240.5  2824.0

head(pacds.merge@counts[, 1:10])
#> 6 x 10 sparse Matrix of class "dgCMatrix"
#>   [[ suppressing 10 column names 'AAACCCACAAATTGCC.1', 'AAACCCAGTCAATGGG.1', 'AAACCCATCTTAGCAG.1' ..
#>
#> M100273 . . . . . .
#> M100417 1 . . . . . .
#> M100519 . . . . . .
#> M10060  . . . . . .
#> M10092  . . . . . .
#> M101558 . . . . . .
head(pacds.merge@anno)
#>          chr strand     coord UPA_start   UPA_end
#> M100273 chr3      - 114321139 114321139 114321139
#> M100417 chr3      - 121664101 121664101 121664101
#> M100519 chr3      - 124968500 124968500 124968500
#> M10060  chr1      - 108931856 108931856 108931856
#> M10092  chr1      - 109399042 109399042 109399042
```

```
#> M101558 chr3      - 165186766 165186766 165186766
```

The original annotation information of the merged data will be removed and needs to be re-annotated.

```
# annotate PAs
pacds.merge=annotatePAC(pacds.merge, txdb)

# after annotation, the gene and ftr information are present in PACds@anno.
summary(pacds.merge)
```

5 smart RUD index

Get APA index using the smart RUD method (available in movAPA v0.2.0).

The smartRUD indicator is provided in movAPA v0.2.0, and it is recommended to use it. Pay attention to setting clearPAT=1 to remove cases of PAs with only 1 read count. At the same time, check the distance between the two PAs first to select a suitable dist for filtering the proximal and distal PAs of 3'UTR.

```
s=getNearbyPAdist(PACds)
#> Nearby pA distance summary:
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   281.0   867.8  1602.0 13879.8 30425.5 54144.0
```

You can see that the average distance is 10K nt, but the median distance is only 3000 nt, so setting 5000 nt is probably enough.

```
# get proximal and distal PA for each gene
pd=get3UTRAPApd(pacds=PACds, minDist=50, maxDist=5000, minRatio=0.05,
                fixDistal=FALSE, addCols='pd')
#> get3UTRAPApd: filtering by minRatio: gene# before: 440 ; after: 11 ; remove: 429
#> get3UTRAPApd: filtering pd (dist between): gene# before: 11 ; after: 7 ; remove: 4
#> get3UTRAPApd: add four columns to pacds@anno: pdWhich, pdScore, pdRatio, pdDist

# get smartRUD index
rud=movAPAindex(pd, method="smartRUD", sRUD.oweight=TRUE, clearPAT=1)
#> clearPAT>0: set elements in @counts <= 1 to 0!
#> movAPAindex (smartRUD): sRUD.oweight=TRUE, output a list(rud, weight)
head(rud$rud[, 1:5])
#> 6 x 5 Matrix of class "dgeMatrix"
#>        AAACCCACAAATTGCC AAACCCAGTCAATGGG AAACCCATCTTAGCAG AAACGAAAGATTGAGT
#> 205564              NaN              NaN              NaN              NaN
#> 112399              NaN              NaN              NaN              NaN
#> 25966               NaN              NaN              NaN              NaN
#> 51710               NaN              NaN              NaN              NaN
#> 6654                NaN              NaN              NaN              NaN
#> 84896               NaN              NaN              NaN              NaN
#>        AAACGCTAGGAGTCTG
#> 205564              NaN
#> 112399              NaN
#> 25966               NaN
#> 51710               NaN
```

```
#> 6654                NaN
#> 84896               NaN
head(rud$weight[, 1:5])
#> 6 x 5 sparse Matrix of class "dgCMatrix"
#>        AAACCCACAAATTGCC AAACCCAGTCAATGGG AAACCCATCTTAGCAG AAACGAAAGATTGAGT
#> 205564                .                .                .                .
#> 112399                .                .                .                .
#> 25966                 .                .                .                .
#> 51710                 .                .                .                .
#> 6654                  .                .                .                .
#> 84896                 .                .                .                .
#>        AAACGCTAGGAGTCTG
#> 205564                .
#> 112399                .
#> 25966                 .
#> 51710                 .
#> 6654                  .
#> 84896                 .

## we can also calculate the RUD index
## rud1=movAPAindex(pd, method="RUD")
```

6 Visualizing PAs with vizAPA

The *vizAPA* package is a comprehensive package for visualization of APA dynamics in single cells. vizAPA imports various types of APA data and genome annotation sources through unified data structures. vizAPA also enables identification of genes with differential APA usage (also called APA markers). Four unique modules are provided in vizAPA for visualizing APA dynamics across cell groups and at the single-cell level.

Here we use vizAPA to show the gene structure and PA location.

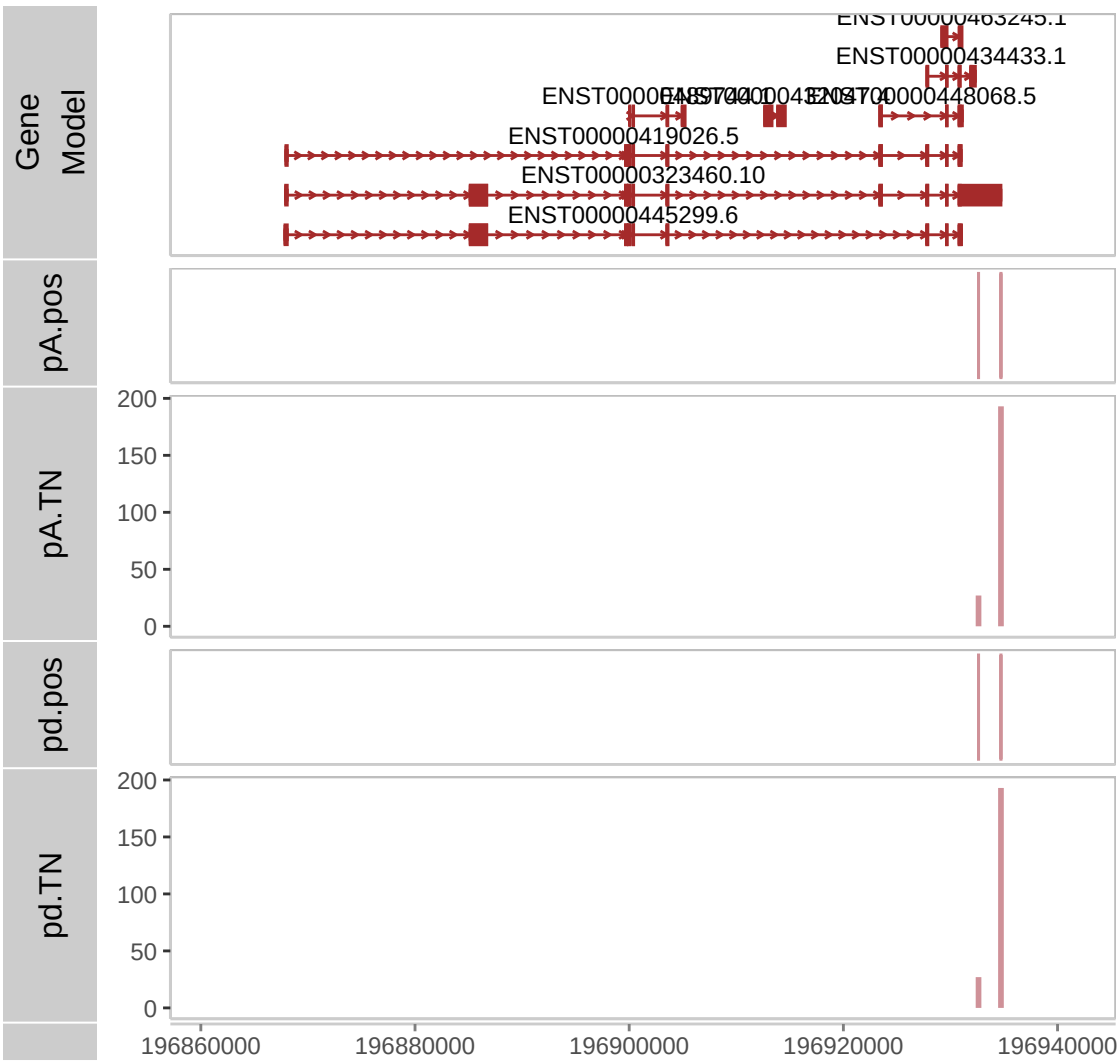
```
library(vizAPA, quietly = TRUE)
#> Registered S3 method overwritten by 'GGally':
#>   method from
#>   +.gg   ggplot2
#>
#>     'vizAPA'
#> The following object is masked from 'package:movAPA':
#>
#>     scPACds

# to choose one gene, first we get the total counts of each PA
PACds@anno$TN=rowSums(PACds@counts)
pd@anno$TN=rowSums(pd@counts)

# construct the genome annotation object
annoSource=new("annoHub")
annoSource=addAnno(annoSource, txdb)

gene=pd@anno$gene[1]
vizTracks(gene=gene,
          PACds.list=list(pA=PACds, pd=pd),
```

```
          PA.show=c("pos","TN"),
          annoSource=annoSource,
          PA.columns="coord", PA.width=10,
          space5=1000, space3=1000)
#> The region may be too large to plot [68858 nt], you can set genomicRegion to plot a smaller region
#> Plot tracks for region: chr3:+:196866856:196935714
#> Get gene model track from annoSource[ txdb ]...
#> Parsing transcripts...
#> Parsing exons...
#> Parsing cds...
#> Parsing utrs...
#> ------exons...
#> ------cdss...
#> ------introns...
#> ------utr...
#> aggregating...
#> Done
#> Constructing graphics...
#> Get PACds track...
#> chr3:+:196866856:196935714
#> Get PACds track...
#> chr3:+:196866856:196935714
```



The gene IDs between different annotations may not be consistent, and we can also use vizAPA to map different IDs.

```
library(Homo.sapiens)
#>      OrganismDbi
#>      GO.db
#>      org.Hs.eg.db
#>
#>      TxDb.Hsapiens.UCSC.hg19.knownGene
orgdb=Homo.sapiens
genes=getAnnoGenes(orgdb)
head(genes)
#>     chr strand     start       end gene_entrezid    gene_ensembl gene_symbol
#> 1 chr19      -  58858172  58874214             1 ENSG00000121410        A1BG
#> 2  chr8      +  18248755  18258723            10 ENSG00000156006        NAT2
#> 3 chr20      -  43248163  43280376           100 ENSG00000196839         ADA
#> 4 chr18      -  25530930  25757445          1000 ENSG00000170558        CDH2
#> 5  chr1      - 243651535 244006886         10000 ENSG00000117020        AKT3
#> 6  chrX      +  49217763  49233491     100008586 ENSG00000236362     GAGE12F
```

```
genes[genes$gene_entrezid==gene, ]
#>       chr strand     start       end gene_entrezid    gene_ensembl gene_symbol
#> 6278 chr3      + 196594727 196661584        205564 ENSG00000119231       SENP5
genes[genes$gene_entrezid==10003, ]
#>      chr strand    start      end gene_entrezid    gene_ensembl gene_symbol
#> 10 chr11      + 89867818 89925779         10003 ENSG00000077616     NAALAD2
```

7 Session Information

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

```
sessionInfo()
#> R version 4.2.2 (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] stats4    stats     graphics  grDevices utils     datasets  methods
#> [8] base
#>
#> other attached packages:
#>  [1] Homo.sapiens_1.3.1
#>  [2] TxDb.Hsapiens.UCSC.hg19.knownGene_3.2.2
#>  [3] org.Hs.eg.db_3.16.0
#>  [4] GO.db_3.16.0
#>  [5] OrganismDbi_1.40.0
#>  [6] vizAPA_0.1.0
#>  [7] TxDb.Hsapiens.UCSC.hg38.knownGene_3.16.0
#>  [8] GenomicFeatures_1.50.2
#>  [9] AnnotationDbi_1.60.0
#> [10] Biobase_2.58.0
#> [11] BSgenome.Hsapiens.UCSC.hg38_1.4.5
#> [12] BSgenome_1.66.2
#> [13] rtracklayer_1.58.0
#> [14] Biostrings_2.66.0
#> [15] XVector_0.38.0
#> [16] GenomicRanges_1.50.1
#> [17] GenomeInfoDb_1.34.9
#> [18] IRanges_2.32.0
#> [19] S4Vectors_0.36.0
#> [20] BiocGenerics_0.44.0
#> [21] movAPA_0.2.0
```

```
#>
#> loaded via a namespace (and not attached):
#>   [1] backports_1.4.1             Hmisc_5.0-0
#>   [3] BiocFileCache_2.6.0         plyr_1.8.8
#>   [5] lazyeval_0.2.2              sp_1.5-1
#>   [7] splines_4.2.2               BiocParallel_1.32.1
#>   [9] listenv_0.9.0               ggplot2_3.4.0
#>  [11] TFBSTools_1.36.0            digest_0.6.30
#>  [13] ensembldb_2.22.0            htmltools_0.5.3
#>  [15] fansi_1.0.3                 checkmate_2.1.0
#>  [17] magrittr_2.0.3              memoise_2.0.1
#>  [19] grImport2_0.2-0             cluster_2.1.4
#>  [21] tzdb_0.3.0                  globals_0.16.2
#>  [23] readr_2.1.3                 annotate_1.76.0
#>  [25] matrixStats_0.63.0          R.utils_2.12.2
#>  [27] ggbio_1.46.0                prettyunits_1.1.1
#>  [29] jpeg_0.1-10                 colorspace_2.0-3
#>  [31] blob_1.2.3                  rappdirs_0.3.3
#>  [33] xfun_0.35                   dplyr_1.0.10
#>  [35] crayon_1.5.2                RCurl_1.98-1.9
#>  [37] graph_1.76.0                progressr_0.12.0
#>  [39] TFMPvalue_0.0.9             VariantAnnotation_1.44.0
#>  [41] survival_3.4-0              glue_1.6.2
#>  [43] gtable_0.3.1                zlibbioc_1.44.0
#>  [45] DelayedArray_0.24.0         future.apply_1.10.0
#>  [47] scales_1.2.1                DBI_1.1.3
#>  [49] GGally_2.1.2                Rcpp_1.0.9
#>  [51] htmlTable_2.4.1             xtable_1.8-4
#>  [53] progress_1.2.2              foreign_0.8-83
#>  [55] bit_4.0.5                   Formula_1.2-4
#>  [57] htmlwidgets_1.5.4           httr_1.4.4
#>  [59] RColorBrewer_1.1-3          ellipsis_0.3.2
#>  [61] pkgconfig_2.0.3             reshape_0.8.9
#>  [63] XML_3.99-0.12               R.methodsS3_1.8.2
#>  [65] farver_2.1.1                nnet_7.3-18
#>  [67] dbplyr_2.2.1                deldir_1.0-6
#>  [69] utf8_1.2.2                  tidyselect_1.2.0
#>  [71] labeling_0.4.2              rlang_1.0.6
#>  [73] reshape2_1.4.4              munsell_0.5.0
#>  [75] tools_4.2.2                 cachem_1.0.6
#>  [77] cli_3.4.1                   DirichletMultinomial_1.40.0
#>  [79] generics_0.1.3              RSQLite_2.2.18
#>  [81] ade4_1.7-22                 evaluate_0.18
#>  [83] stringr_1.4.1               fastmap_1.1.0
#>  [85] yaml_2.3.6                  knitr_1.41
#>  [87] bit64_4.0.5                 caTools_1.18.2
#>  [89] AnnotationFilter_1.22.0     KEGGREST_1.38.0
#>  [91] RBGL_1.74.0                 future_1.30.0
#>  [93] R.oo_1.25.0                 poweRlaw_0.70.6
#>  [95] pracma_2.4.2                xml2_1.3.3
#>  [97] biomaRt_2.54.0              compiler_4.2.2
#>  [99] rstudioapi_0.14             filelock_1.0.2
#> [101] curl_4.3.3                  png_0.1-7
```

```
#> [103] tibble_3.1.8                stringi_1.7.8
#> [105] highr_0.9                   lattice_0.20-45
#> [107] ProtGenerics_1.30.0         CNEr_1.34.0
#> [109] Matrix_1.5-3                vctrs_0.5.1
#> [111] pillar_1.8.1                lifecycle_1.0.3
#> [113] BiocManager_1.30.19         data.table_1.14.6
#> [115] bitops_1.0-7                R6_2.5.1
#> [117] BiocIO_1.8.0                latticeExtra_0.6-30
#> [119] gridExtra_2.3               parallelly_1.33.0
#> [121] motifStack_1.42.0           codetools_0.2-18
#> [123] dichromat_2.0-0.1           MASS_7.3-58.1
#> [125] gtools_3.9.4                assertthat_0.2.1
#> [127] seqLogo_1.64.0              SummarizedExperiment_1.28.0
#> [129] rjson_0.2.21                withr_2.5.0
#> [131] SeuratObject_4.1.3          GenomicAlignments_1.34.0
#> [133] Rsamtools_2.14.0            GenomeInfoDbData_1.2.9
#> [135] parallel_4.2.2              hms_1.1.2
#> [137] rpart_4.1.19                grid_4.2.2
#> [139] rmarkdown_2.18              MatrixGenerics_1.10.0
#> [141] biovizBase_1.46.0           base64enc_0.1-3
#> [143] interp_1.1-3                restfulr_0.0.15
```