SCANVIS - SCoring, ANnotating and VISualzing splice junctions

Phaedra Agius

May 1, 2024

This document describes the main functions in SCANVIS as well as how and when to execut the functions.

1 Introduction

SCANVIS is a tool for scoring and annotating splice junctions (SJs) with gene names, junction type and frame-shifts. It has a visualization function for generating static sashimi plots with annotation details differentiated by color and line types, and which can be overlaid with variants and read profiles. Samples in one cohort can also be merged into one figure for quick contrast to another cohort. To score and annotate a sample, SCANVIS requires two main inputs: SJ details (coordinates and split-read support) and SCANVIS-readable gene annotations. A function for extracting SCANVIS-readable annotation from a GTF file of choice is included in the package (see below). SCANVIS processes one sample at a time, the output being a matrix containing all original SJ details (coordinaes and read support) together with the following annotation details: *(i)* a Relative Read Support (RRS) score *(ii)* the RRS genomic interval, and *(iii)* names of gene/s that overlap the SJ. Unannotated SJs (USJs) are further described by *(iv)* frame-shfits and *(v)* junction type, this being either *exon-skip, alt5p, alt3p, IsoSwitch, Unknown* or *Novel Exon (NE)*. USJs described as *IsoSwitch* are SJs that straddle two mutually exclusive isoforms while *Unknown* USJs are contained in annotated intronic regions. A RRS genomic interval is defined as the minimal interval containing at least one gene overlapping the query SJ and at least one annotated SJ (ASJ). The RRS score is the ratio of x to $x+y$, where x is the query junction read support and y is the median read support of ASJs in the RRS genomic interval. This approaach keeps RRS scores free from undue influence of USJs which tend to be frequent, have poor read support and may be alignment artifacts. Once all SJs are scored and annotated, SCANVIS looks for potential *NE*s defined by USJs coinciding in annotated intronic regions. *NE*s are scored by the mean RRS of all SJs landing on the *NE* start/end coordinates. If the BAM file is supplied (optional, see details below) SCANVIS also computes a Relative Read-Coverage (RRC) score for NEs only, defined as $c/(c5+c+c3)$ where c is the mean *NE* read coverage, and $c5$ and $c3$ are mean read coverages for flanking regions, both defined as intervals 0.2 times the *NE* interval. Note that RRCs are only computed when users supply the BAM file, an optional feature in SCANVIS since BAMs are not always accessible to users.

Also note that processing a sample with the bam file takes significantly longer and requires more compute memory.

2 SCANVISAnnotation data

SCANVIS functions require a SCANVIS-readable gene annotation file. Users can generate their own annotation file using the `SCANVISannotation` function with a suitable GTF url. We provide a portion of the human gencode v19 as output by the `SCANVISannotation` function with the data examples attached to the package, which can be uploaded likeso:

```
library(SCANVIS)
data(SCANVISexamples)
names(gen19)

## [1] "EXONS"        "INTRONS"       "GENES"         "GENES.merged"
```

A full version can easily be generated using `SCANVISannotation` by simply pointing it to a suitable url to a GTF file. Note that `gen19` is a list with the following components: GENES, GENES.merged, EXONS and IN-TRONS where GENES contains all gene names/ids and full genomic coordinates, GENES.merged is the union of the intervals in GENES so that overlapping genes are consolidated into one interval, EXONS contains full isoform and exon numbers/ids and coordinates and INTRONS contains genomic coordinates of intronic regions that do not overlap any known exons within this annotation. Once this gene annotation object is obtained, it can be used to process any number of samples that were aligned to the same reference genome as that used for the GTF source.

3 SCANning: SCoring and ANnotating splice junctions

With the annotation file at hand, users can now execute the `SCANVISscan` function to SCore and ANnotate each SJ in a sample. The package contains a few examples that users can load up and reference. Specifically, the exemplary data include 2 LUSC (lung squamous cell carcinoma), 3 GBM (glioblastoma) and 2 LUAD (lung adenocarcinoma) samples, all derived from STAR [Dobin *et al*] alignments to TCGA samples. Due to package space restrictions, SJ details for select genomic regions only are included. Some samples are the output of `SCANVISscan` and are included for referenced in the `SCANVISvisual` documentation, while one one sample named `gbm3` is prepared in the required format for `SCANVISscan` and can be processed likeso:

```
head(gbm3)

##       chr    start      end      uniq.reads
## [1,] "chr6" "46672434" "46672889" "18"
## [2,] "chr6" "46673039" "46675727" "13"
```

```
## [3,] "chr6" "46675899" "46677063" "8"
## [4,] "chr6" "46677156" "46678281" "6"
## [5,] "chr6" "46678396" "46679232" "12"
## [6,] "chr6" "46679357" "46680005" "16"

gbm3.scn<-SCANVISscan(sj=gbm3,gen=gen19,Rcut=5,bam=NULL,samtools=NULL)

## [1] "*** Categorizing unannotated splice junctions ... ***"
## [1] "*** DONE: Categorizing unannotated splice junctions ***"
## [1] "*** Computing RRS by median of local ASJ reads ... ***"
## [1] "*** DONE: Computing RRS by median of local ASJ reads ***"
## [1] "*** Designating gene names to sj coordinates ... ***"
## [1] "*** DONE: Designating gene names to sj coordinates ***"
## [1] "*** Querying inFrameStatus ... ***"
## [1] "*** DONE: Querying FrameStatus ***"
## [1] "*** Collecting and scoring all potential NEs ... ***"
## [1] "*** DONE: Collecting and scoring all potential NEs ... ***"

head(gbm3.scn)

##       chr      start       end       uniq.reads JuncType RRS
## [1,] "chr6" "46672434" "46672889" "18"         "annot"  "0.6"
## [2,] "chr6" "46673039" "46675727" "13"         "annot"  "0.52"
## [3,] "chr6" "46675899" "46677063" "8"          "annot"  "0.4"
## [4,] "chr6" "46677156" "46678281" "6"          "annot"  "0.333333333333333"
## [5,] "chr6" "46678396" "46679232" "12"         "annot"  "0.5"
## [6,] "chr6" "46679357" "46680005" "16"         "annot"  "0.571428571428571"
##      genomic_interval        gene_name FrameStatus
## [1,] "chr6:46671938-46703430" "PLA2G7"  "NA"
## [2,] "chr6:46671938-46703430" "PLA2G7"  "NA"
## [3,] "chr6:46671938-46703430" "PLA2G7"  "NA"
## [4,] "chr6:46671938-46703430" "PLA2G7"  "NA"
## [5,] "chr6:46671938-46703430" "PLA2G7"  "NA"
## [6,] "chr6:46671938-46703430" "PLA2G7"  "NA"
```

If users have acess to the corresponding BAM file, they can supply the BAM url and an executable SAMTOOLS url to derive RRC scores for any Novel Exons detected. Note that the default settings for `bam` and `samtools` is NULL, and if `bam` points to a url then users must define `samtools` as the path to the executable samtools so that read depths can be estimated from the BAM file.

4   Mapping variants to SCANned junctions

Once a sample has been SCANned, users can map variants (if available) to SJs using the `SCANVISlinkvar` function. While this function is optional, it should be executed prior to using `SCANVISmerge` or `SCANVISvisual` functions if variants are available. The `SCANVISlinkvar` function maps a variant to a SJ if the variant is located in the same genomic interval described by the gene/s overlapping the SJ. In the examples included with the package there is a set of

toy variants in the required format which can be mapped to the gbm3 example likeso:

```
head(gbm3.vcf)

##   chr     start       end        passedMUT
## 1 "chr6" " 46778280" " 46778282" "ZZ>T"
## 2 "chr6" " 46820148" " 46820149" "Z>AA"
## 3 "chr6" " 46821228" " 46821229" "Z>T"
## 4 "chr9" "139870063" "139870065" "ZZ>C"
## 5 "chr9" "139870263" "139870266" "ZZZ>G"
## 6 "chr9" "139871956" "139871957" "Z>A"

gbm3.scnv<-SCANVISlinkvar(gbm3.scn,gbm3.vcf,gen19)

## [1] "Mapping variants to SJs for chr12"
## [1] "Mapping variants to SJs for chr9"
## [1] "Mapping variants to SJs for chr6"

gbm3.scnv[10:23,]

##         chr     start       end        uniq.reads JuncType RRS
##  [1,] "chr6" "46821809" "46822451" "8"        "annot"  "0.432432432432432"
##  [2,] "chr6" "46821809" "46823710" "19"       "annot"  "0.644067796610169"
##  [3,] "chr6" "46822519" "46823710" "8"        "annot"  "0.432432432432432"
##  [4,] "chr6" "46823796" "46824454" "27"       "annot"  "0.72"
##  [5,] "chr6" "46824515" "46824603" "28"       "annot"  "0.727272727272727"
##  [6,] "chr6" "46824646" "46825865" "22"       "annot"  "0.676923076923077"
##  [7,] "chr6" "46827261" "46828451" "19"       "annot"  "0.644067796610169"
##  [8,] "chr6" "46828632" "46830624" "9"        "annot"  "0.461538461538462"
##  [9,] "chr6" "46830834" "46832778" "12"       "annot"  "0.533333333333333"
## [10,] "chr6" "46832935" "46834661" "8"        "annot"  "0.432432432432432"
## [11,] "chr6" "46834875" "46836619" "6"        "annot"  "0.363636363636364"
## [12,] "chr6" "46839751" "46845938" "6"        "annot"  "0.363636363636364"
## [13,] "chr6" "46846143" "46847554" "12"       "annot"  "0.533333333333333"
## [14,] "chr6" "46851403" "46851831" "5"        "annot"  "0.32258064516129"
##       genomic_interval         gene_name FrameStatus passedMUT
##  [1,] "chr6:46820249-46922680" "GPR116"  "NA"        "chr6:46821228;Z>T"
##  [2,] "chr6:46820249-46922680" "GPR116"  "NA"        "chr6:46821228;Z>T"
##  [3,] "chr6:46820249-46922680" "GPR116"  "NA"        "chr6:46821228;Z>T"
##  [4,] "chr6:46820249-46922680" "GPR116"  "NA"        "chr6:46821228;Z>T"
##  [5,] "chr6:46820249-46922680" "GPR116"  "NA"        "chr6:46821228;Z>T"
##  [6,] "chr6:46820249-46922680" "GPR116"  "NA"        "chr6:46821228;Z>T"
##  [7,] "chr6:46820249-46922680" "GPR116"  "NA"        "chr6:46821228;Z>T"
##  [8,] "chr6:46820249-46922680" "GPR116"  "NA"        "chr6:46821228;Z>T"
##  [9,] "chr6:46820249-46922680" "GPR116"  "NA"        "chr6:46821228;Z>T"
## [10,] "chr6:46820249-46922680" "GPR116"  "NA"        "chr6:46821228;Z>T"
## [11,] "chr6:46820249-46922680" "GPR116"  "NA"        "chr6:46821228;Z>T"
## [12,] "chr6:46820249-46922680" "GPR116"  "NA"        "chr6:46821228;Z>T"
## [13,] "chr6:46820249-46922680" "GPR116"  "NA"        "chr6:46821228;Z>T"
## [14,] "chr6:46820249-46922680" "GPR116"  "NA"        "chr6:46821228;Z>T"
```

When multiple variants map to the same gene/s overlapping a SJ, these are
"—" separated. Some examples of this occur at the end of the output matrix:

```
gbm3.scnv[38:51,]

##        chr    start       end         uniq.reads JuncType   RRS
##  [1,] "chr9" "139872145" "139873444" "1522"     "annot"    "0.993472584856397"
##  [2,] "chr9" "139872145" "139874397" "22"       "exon.skip" "0.6875"
##  [3,] "chr9" "139873585" "139873674" "1784"     "annot"    "0.994425863991081"
##  [4,] "chr9" "139873601" "139873674" "6"        "alt5p"    "0.375"
##  [5,] "chr9" "139873752" "139874397" "1799"     "annot"    "0.994472084024323"
##  [6,] "chr9" "139873854" "139874397" "46"       "annot"    "0.821428571428571"
##  [7,] "chr9" "139874515" "139874634" "2018"     "annot"    "0.995069033530572"
##  [8,] "chr9" "139874515" "139875284" "52"       "exon.skip" "0.838709677419355"
##  [9,] "chr9" "139874737" "139875131" "7"        "annot"    "0.411764705882353"
## [10,] "chr9" "139874737" "139875284" "1753"     "annot"    "0.994327850255247"
## [11,] "chr9" "139875308" "139876009" "7"        "alt3p"    "0.411764705882353"
## [12,] "chr9" "139875308" "139876030" "1431"     "annot"    "0.993060374739764"
## [13,] "chr9" "139879016" "139879487" "5"        "annot"    "0.333333333333333"
## [14,] "chr9" "139879590" "139879848" "5"        "annot"    "0.333333333333333"
##       genomic_interval          gene_name
##  [1,] "chr9:139871956-139880862" "PTGDS"
##  [2,] "chr9:139871956-139880862" "PTGDS"
##  [3,] "chr9:139871956-139880862" "PTGDS"
##  [4,] "chr9:139871956-139880862" "PTGDS"
##  [5,] "chr9:139871956-139880862" "PTGDS"
##  [6,] "chr9:139871956-139880862" "PTGDS"
##  [7,] "chr9:139871956-139880862" "PTGDS"
##  [8,] "chr9:139871956-139880862" "PTGDS"
##  [9,] "chr9:139871956-139880862" "PTGDS"
## [10,] "chr9:139871956-139880862" "PTGDS"
## [11,] "chr9:139871956-139880862" "PTGDS"
## [12,] "chr9:139871956-139880862" "PTGDS"
## [13,] "chr9:139871956-139880862" "LCNL1,PTGDS"
## [14,] "chr9:139871956-139880862" "LCNL1,PTGDS"
##       FrameStatus
##  [1,] "NA"
##  [2,] "ENST00000224167.2:OUTframe,ENST00000457950.1:OUTframe,ENST00000371625.3:OUTframe
##  [3,] "NA"
##  [4,] "INframe"
##  [5,] "NA"
##  [6,] "NA"
##  [7,] "NA"
##  [8,] "ENST00000224167.2:INframe,ENST00000457950.1:OUTframe,ENST00000371625.3:INframe,E
##  [9,] "NA"
## [10,] "NA"
## [11,] "ENST00000224167.2:OUTframe,ENST00000371625.3:OUTframe,ENST00000471521.1:INframe,
## [12,] "NA"
## [13,] "NA"
```

```
## [14,] "NA"
##       passedMUT
##  [1,] "chr9:139871956;Z>A|chr9:139872544;Z>C"
##  [2,] "chr9:139871956;Z>A|chr9:139872544;Z>C"
##  [3,] "chr9:139871956;Z>A|chr9:139872544;Z>C"
##  [4,] "chr9:139871956;Z>A|chr9:139872544;Z>C"
##  [5,] "chr9:139871956;Z>A|chr9:139872544;Z>C"
##  [6,] "chr9:139871956;Z>A|chr9:139872544;Z>C"
##  [7,] "chr9:139871956;Z>A|chr9:139872544;Z>C"
##  [8,] "chr9:139871956;Z>A|chr9:139872544;Z>C"
##  [9,] "chr9:139871956;Z>A|chr9:139872544;Z>C"
## [10,] "chr9:139871956;Z>A|chr9:139872544;Z>C"
## [11,] "chr9:139871956;Z>A|chr9:139872544;Z>C"
## [12,] "chr9:139871956;Z>A|chr9:139872544;Z>C"
## [13,] "chr9:139871956;Z>A|chr9:139872544;Z>C"
## [14,] "chr9:139871956;Z>A|chr9:139872544;Z>C"
```

Users have the option to allow some padding or relaxation via the parameter p (default $p = 0$) which maps SJs to variants that are $\leq p$ base pairs away from the borders of any genes overlapping the SJs. If we set $p = 100$ we now see SJs mapped to a variant chr6:46820148;Z¿AA that was not previously mapped:

```
gbm3.scnvp<-SCANVISlinkvar(gbm3.scn,gbm3.vcf,gen19,p=100)

## [1] "Mapping variants to SJs for chr12"
## [1] "Mapping variants to SJs for chr9"
## [1] "Mapping variants to SJs for chr6"

table(gbm3.scnv[,'passedMUT'])

##
##                                                     chr6:46821228;Z>T
##                                      103                            14
## chr9:139871956;Z>A|chr9:139872544;Z>C
##                                       14

table(gbm3.scnvp[,'passedMUT'])

##
##                                chr6:46820148;Z>AA|chr6:46821228;Z>T
##                                      103                            14
## chr9:139871956;Z>A|chr9:139872544;Z>C
##                                       14
```

The `SCANVISlinkvar` function may be executed any number of times for multiple variant sets/calls in the same sample. With each run, a new column is added indicating any variants mapping to the overlapping genes.

5 VISualizing splice junctions in colorful sashimi plots

Once variants (if any) have been linked, users can visualize SJs in regions of interest. To execute `SCANVISvisual` users will need to define a region of interest `roi` parameter since the function generates a visual of a select genomic region. The `roi` parameter can either be defined as a single gene name OR a vector with multiple gene names OR a 3-bit vector with chr,start,end identifying the precise region of interest. The function will then generate a sashimi plot showing annotated SJs in grey and unannotated SJs in various colors to indicate the type of junction. Frame-shifting junctions are indicated with dotted lines, with junctions that induce a frame-shift across all isoforms having a slightly different line type than those that induce frame-shifts in some isoforms. SJ arcs are overlaid with variants when the sample is variant-mapped. Using two lung cell squamous carcinoma samples from TCGA that have been mapped to splice variants in the example data, we can visualize the PPA2 gene likeso:

```
par(mfrow=c(2,1),mar=c(1,1,1,1))
vis.lusc1<-SCANVISvisual('PPA2',gen19,LUSC[[1]],TITLE=names(LUSC)[1],full.annot=TRUE,bam=N

## [1] "*** Sashimi plot  spanning 105004 base pairs is being generated"

vis.lusc2<-SCANVISvisual('PPA2',gen19,LUSC[[2]],TITLE=names(LUSC)[2],full.annot=TRUE,bam=N

## [1] "*** Sashimi plot  spanning 105004 base pairs is being generated"
```
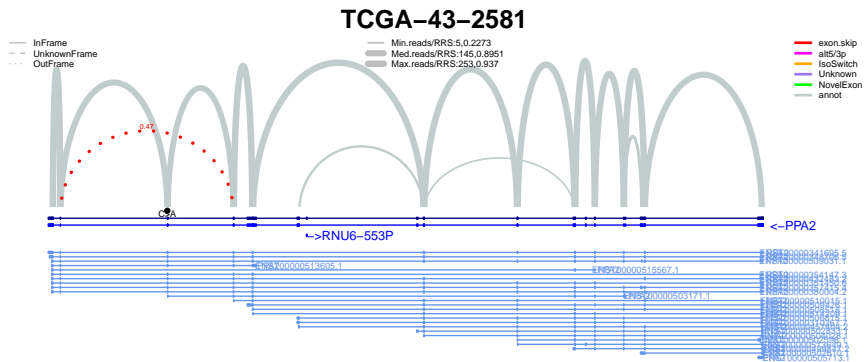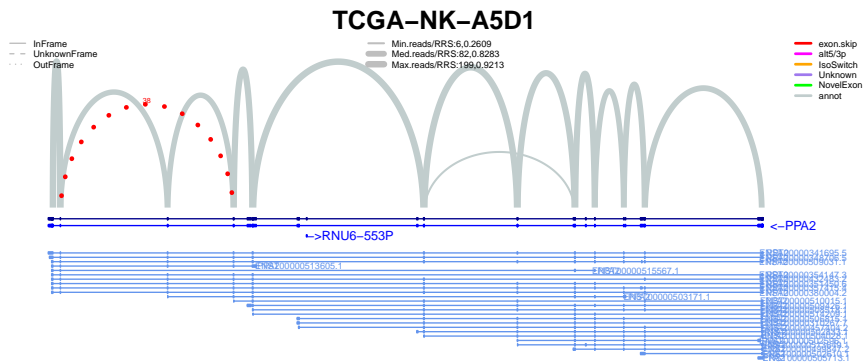
If users wish to overlay a sashimi plots with a read profile, then suitable BAM and SAMTOOLS urls must be provided (default settings are both NULL). Users may also highlight any SJs of interest. This is particularly useful for annotated SJs which are not as distinguishable as the unannotated SJs that appear in color. To do this, users can specify SJs of interest via the SJ.special parameter likeso:

```
ASJ<-tail(vis.lusc2[[2]])[,1:3]
c2<-SCANVISvisual('PPA2',gen19,LUSC[[2]],TITLE=names(LUSC)[2],full.annot=TRUE,SJ.special=A

## [1] "*** Sashimi plot  spanning 105004 base pairs is being generated"
```
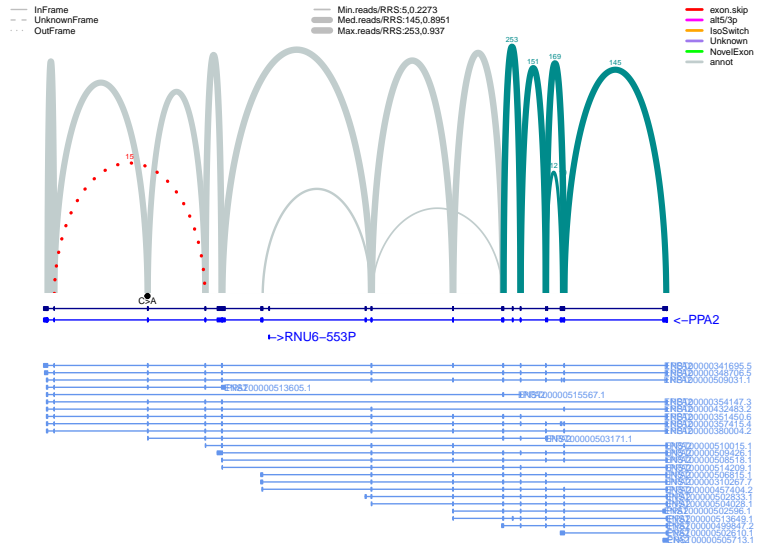
TCGA–43–2581

To visualize multiple samples merged into one figure, submit the samples as
a list likeso:

```
vis.lusc.merged<-SCANVISvisual('PPA2',gen19,LUSC,TITLE='Two LUSC samples, merged',full.ann

## [1] "*** Sashimi plot  spanning 105004 base pairs is being generated"
```

Two LUSC samples, merged


```
## [1] "plotting mutations ..."
```

More parameter descriptions and different examples on how to use `SCANVISvisual` can be found in the `help` manual.

```
sessionInfo()
```

```
## R version 4.4.0 beta (2024-04-15 r86425)
## Platform: x86_64-pc-linux-gnu
## Running under: Ubuntu 22.04.4 LTS
##
## Matrix products: default
## BLAS:   /home/biocbuild/bbs-3.19-bioc/R/lib/libRblas.so
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
```

```
##  [9] LC_ADDRESS=C                LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## time zone: America/New_York
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] SCANVIS_1.18.0
##
## loaded via a namespace (and not attached):
##  [1] Matrix_1.7-0              jsonlite_1.8.8
##  [3] highr_0.10               compiler_4.4.0
##  [5] rjson_0.2.21             crayon_1.5.2
##  [7] plotrix_3.8-4            SummarizedExperiment_1.34.0
##  [9] Biobase_2.64.0           Rsamtools_2.20.0
## [11] GenomicRanges_1.56.0     bitops_1.0-7
## [13] Biostrings_2.72.0        GenomicAlignments_1.40.0
## [15] parallel_4.4.0           IRanges_2.38.0
## [17] BiocParallel_1.38.0      yaml_2.3.8
## [19] lattice_0.22-6           R6_2.5.1
## [21] XVector_0.44.0           S4Arrays_1.4.0
## [23] curl_5.2.1               GenomeInfoDb_1.40.0
## [25] knitr_1.46               BiocGenerics_0.50.0
## [27] XML_3.99-0.16.1          DelayedArray_0.30.0
## [29] MatrixGenerics_1.16.0    GenomeInfoDbData_1.2.12
## [31] xfun_0.43                SparseArray_1.4.0
## [33] zlibbioc_1.50.0          grid_4.4.0
## [35] rtracklayer_1.64.0       S4Vectors_0.42.0
## [37] evaluate_0.23            codetools_0.2-20
## [39] abind_1.4-5              stats4_4.4.0
## [41] RCurl_1.98-1.14          restfulr_0.0.15
## [43] httr_1.4.7               matrixStats_1.3.0
## [45] tools_4.4.0              BiocIO_1.14.0
## [47] UCSC.utils_1.0.0
```

References

[Dobin *et al*] Dobin, Alexander *et al.* (2013) STAR: ultrafast universal RNA-seq aligner, *B*ioinformatics, 29(1) 15-21.