Overview of fusion analysis

Robert Scharpf

March 23, 2018

Contents

1 Introduction

Nearly all sequence junctions formed by a rearrangement occur in non-coding regions of the genome. With whole genome sequencing, we can identify the location of the noncoding sequence junctions. When these junctions occur within a gene (usually intronic), we can evaluate whether the new transcript produced by the rearrangement is in-frame. This vignette provides an overview, extending the rearrangement analyses in the `svrearrange` package.

Our analysis to identify in-frame fusions begin with a `RearrangementList` for ovarian cell line `CGOV11T\_1` created by and available from the `svrearrange` package.

2 Analysis pipeline

```
suppressPackageStartupMessages(library(trellis))
extdata <- system.file("extdata", package="trellis")
rfile <- file.path(extdata, "CGOV11T_1.bam.rds")
rlist <- readRDS(rfile)
rlist
## An object of class 'RearrangementList'
##   number rearrangement objects:  26
##   Use '[[i]]' to return a single Rearrangement object'
```

The above `rlist` contains 68 rearrangements. We begin by selecting only the rearrangements in which both ends of a sequence junction are within 5kb of a known transcript.

```
build <- "hg19"
near.coding <- seqJunctionNearTx(rlist, build)
rlist <- rlist[ near.coding ]
length(rlist)
## [1] 26
```

As discussed in the vignette for rearrangements, each candidate rearrangement has two possible 5-prime to 3-prime orientations that can be inferred by the orientation of the rearranged read pairs (RRPs) and split reads (SRs). We use the function `fiveTo3List` to order each rearrangement in the `rlist` object by these two orientations. In doing so, the length of the `rlist` object doubles.

```
rlist2 <- fiveTo3List(rlist, build)
length(rlist2)
## [1] 52
```

Having established the 5-prime to 3-prime orientations in the `rlist2` object, we make the definition of the sequence junctions more precise.

```
jxns <- seqJunctions_Rlist(rlist2)
jxns
## GRanges object with 52 ranges and 2 metadata columns:
##      seqnames                 ranges strand |         rid                 3p
##         <Rle>              <IRanges>  <Rle> | <character>          <GRanges>
##   5p     chr3 [ 58908808,  58908808]      - | 10546-10582  chr3:59949878:-
##   5p     chr3 [ 59949878,  59949878]      + | 10582-10546  chr3:58908808:+
##   5p     chr3 [171752966, 171752966]      - | 13607-44076 chr15:99550846:-
##   5p    chr15 [ 99550785,  99550785]      - | 44076-13607 chr3:171753007:-
##   5p     chr3 [187389262, 187389262]      - | 14161-14186 chr3:187988635:-
##   ..      ...                    ...    ... .         ...              ...
##   5p     chr2 [213914447, 213914447]      + |   9181-9136 chr2:213286255:+
##   5p     chr2 [218936235, 218936235]      - |  9258-33360 chr11:62154698:+
##   5p    chr11 [ 62154698,  62154698]      - |  33360-9258 chr2:218936235:+
##   5p     chr2 [219352749, 219352749]      + |  9281-33369 chr11:62362005:-
##   5p    chr11 [ 62362005,  62362005]      + |  33369-9281 chr2:219352749:-
##   -------
##   seqinfo: 23 sequences from an unspecified genome
```

In addition, we require the 3-prime genomic region of the junction to lie strictly within a transcript, while the 5-prime genomic region forming the junction can occur either in the promoter or within the 5-prime transcript. Operationally, we define the promoter as the genomic region 5kb upstream of the transcription start site of the 5-prime gene. We refer to the remaining junctions as `coding_jxns`. As the coding junctions are named by the rearrangement ids, we can use these names to subset the `rlist2` object, excluding those rearrangements that do not meet the above criteria.

```
coding_jxns <- codingJunctions(jxns, build)
coding_jxns
## GRanges object with 44 ranges and 3 metadata columns:
##                seqnames                 ranges strand |         rid
##                   <Rle>              <IRanges>  <Rle> | <character>
##   10546-10582     chr3 [ 58908808,  58908808]      - | 10546-10582
##   10582-10546     chr3 [ 59949878,  59949878]      + | 10582-10546
##   13607-44076     chr3 [171752966, 171752966]      - | 13607-44076
##   14161-14186     chr3 [187389262, 187389262]      - | 14161-14186
##   16502-16504     chr5 [ 59150971,  59150971]      + | 16502-16504
##           ...      ...                    ...    ... .         ...
##     9181-9136     chr2 [213914447, 213914447]      + |   9181-9136
##    9258-33360     chr2 [218936235, 218936235]      - |  9258-33360
##    33360-9258    chr11 [ 62154698,  62154698]      - |  33360-9258
##    9281-33369     chr2 [219352749, 219352749]      + |  9281-33369
##    33369-9281    chr11 [ 62362005,  62362005]      + |  33369-9281
##                              3p                tx_name
##                       <GRanges>            <character>
##   10546-10582  chr3:59949878:-                NM_198463
##   10582-10546  chr3:58908808:+    NM_001166243,NM_002012
##   13607-44076 chr15:99550846:-        promoter_NM_022763
##   14161-14186 chr3:187988635:-        promoter_NM_001048
##   16502-16504  chr5:59646153:+ NM_001104631,NM_001165899
##           ...              ...                       ...
##     9181-9136 chr2:213286255:+    NM_016260,NM_001079526
##    9258-33360 chr11:62154698:+                 NM_198483
##    33360-9258 chr2:218936235:+    NM_001083926,NM_025080
##    9281-33369 chr11:62362005:-                 NM_020935
##    33369-9281 chr2:219352749:-        promoter_NM_022830
##   -------
##   seqinfo: 23 sequences from hg19 genome
rlist2 <- rlist2[ names(coding_jxns) ]
```

Next, we use the `fuseCDS_Rlist` function to extract for each rearrangement the full CDS of the fused sequence in the somatic genome (`fusions`), the partial CDS of the 5-prime (`tum.5p`) and 3-prime (`tum.3p`) transcripts that are involved in the fusion, and the full CDS of the 5-prime (`ref.5p`) and 3-prime transcripts in the reference genome. To speed up computation in the remaining portions of this vignette, we restrict the `RearrangementList` object to a rearrangement involving the fusion of the tyrosine kinase IKAROS to a known driver ERBB4. Note, we use a single bracket `[` when subsetting the `RearrangementList` so that the resulting object is still an instance of `RearrangementList` (albeit a length-one list).

```
ikaros.erbb4 <- "9136-9181"
coding_jxns <- coding_jxns[ikaros.erbb4]
cds.list <- fuseCDS_Rlist(rlist2[ikaros.erbb4], coding_jxns)
cds.list
## List of length 6
## names(6): fusions tum.5p tum.3p ref.5p ref.3p coding_junctions
```

For each fusion, we translate the CDS in each of the three possible reading frames and partition the amino acid sequence of the fusion protein into the sequences derived from the 5-prime and 3-prime transcripts. In particular, by partioning the amino acid sequence of the fusion into its 5-prime and 3-prime parts, we can compare the 5-prime partition to the amino acid sequence of the 5-prime reference protein and the 3-prime partition to the amino acid sequence of the 3-prime reference protein. The function `translateCDS` translates the CDS in the 3 possible reading frames.

```
proteins <- translateCDS(cds.list)
##
## Attaching package: 'Biostrings'
## The following object is masked from 'package:trellis':
##
##     type
## The following object is masked from 'package:base':
##
##     strsplit
proteins
## List of length 5
## names(5): fusion tum5p tum3p ref5p ref3p
```

The amino acid sequence obtained by translating the CDS involved in the 5-prime and 3-prime transcripts are represented as `AAStringSet` objects. Since a single gene can have multiple transcripts, we translate every possible combination of 5-prime and 3-prime transcripts in each of the 3 frames. Here, we show the amino acid sequences for the two possible combinations of transcripts involving IKAROS and ERBB4:

```
proteins$fusion[[1]]
## $frame1
##   A AAStringSet instance of length 2
##     width seq                                           names
## [1]   363 MKPATGLWVWVSLLVAAGTVQP...RSQDRYEFSSHIVRGEHTFH* NM_001042599-NM_0...
## [2]   363 MKPATGLWVWVSLLVAAGTVQP...RSQDRYEFSSHIVRGEHTFH* NM_001042599-NM_0...
```

```
##
## $frame2
##   A AAStringSet instance of length 2
##     width seq                                            names
## [1]   362 *SRRQDFGSG*AFSWRRGPSSP...TEARTVMSFHHTLFEGSTHST NM_001042599-NM_0...
## [2]   362 *SRRQDFGSG*AFSWRRGPSSP...TEARTVMSFHHTLFEGSTHST NM_001042599-NM_0...
##
## $frame3
##   A AAStringSet instance of length 2
##     width seq                                            names
## [1]   362 EAGDRTLGLGEPSRGGGDRPAQ...QKPGPL*VFITHCSRGAHIPL NM_001042599-NM_0...
## [2]   362 EAGDRTLGLGEPSRGGGDRPAQ...QKPGPL*VFITHCSRGAHIPL NM_001042599-NM_0...
```

We can also list the amino acid sequences derived from the 5-prime gene and the 3-prime gene:

```
proteins$tum5p[[1]]
## $frame1
##   A AAStringSet instance of length 2
##     width seq                                            names
## [1]    27 MKPATGLWVWVSLLVAAGTVQPSDSQS                    NM_001042599-NM_0...
## [2]    27 MKPATGLWVWVSLLVAAGTVQPSDSQS                    NM_001042599-NM_0...
##
## $frame2
##   A AAStringSet instance of length 2
##     width seq                                            names
## [1]    26 *SRRQDFGSG*AFSWRRGPSSPAILS                     NM_001042599-NM_0...
## [2]    26 *SRRQDFGSG*AFSWRRGPSSPAILS                     NM_001042599-NM_0...
##
## $frame3
##   A AAStringSet instance of length 2
##     width seq                                            names
## [1]    26 EAGDRTLGLGEPSRGGGDRPAQRFSV                     NM_001042599-NM_0...
## [2]    26 EAGDRTLGLGEPSRGGGDRPAQRFSV                     NM_001042599-NM_0...
proteins$tum3p[[1]]
## $frame1
##   A AAStringSet instance of length 2
##     width seq                                            names
## [1]   335 WVNLTSATTVDEATSSAVHWRS...TEARTVMSFHHTLFEGSTHST NM_016260
## [2]   335 WVNLTSATTVDEATSSAVHWRS...TEARTVMSFHHTLFEGSTHST NM_001079526
##
## $frame2
##   A AAStringSet instance of length 2
##     width seq                                            names
## [1]   334 G*TSQVQLLWTKLQAAQFTGGA...LQKPGPL*VFITHCSRGAHIP NM_016260
## [2]   334 G*TSQVQLLWTKLQAAQFTGGA...LQKPGPL*VFITHCSRGAHIP NM_001079526
##
## $frame3
##   A AAStringSet instance of length 2
##     width seq                                            names
## [1]   334 GKPHKCNYCGRSYKQRSSLEEH...YRSQDRYEFSSHIVRGEHTFH NM_016260
## [2]   334 GKPHKCNYCGRSYKQRSSLEEH...YRSQDRYEFSSHIVRGEHTFH NM_001079526
```

The function `partitionAASequence` reorganizes the list of amino acid sequences by frame to facilitate downstream computation.

```
partition.fusion <- partitionAASequence(proteins)
partition.fusion
## List of length 3
## names(3): frame1 frame2 frame3
```

For a fusion to be in-frame, we require that the amino acid sequence derived from the 5- and 3-prime transcripts to be subsequences of the full amino acid sequence of the reference genome that was translated in the same frame. In addition, we require that there be no premature stop codons. The following code accomplishes these two tasks, with the `ref.frames` object below containing the full amino acid sequence of the transcripts associated with IKAROS and ERBB4.

```
nostop.list <- noPrematureStop(partition.fusion)
nostop.list
## $frame1
## LogicalList of length 1
## [["9136-9181"]] TRUE TRUE
##
## $frame2
## LogicalList of length 1
## [["9136-9181"]] FALSE FALSE
##
## $frame3
## LogicalList of length 1
## [["9136-9181"]] FALSE FALSE
ref.frames <- organizeReferenceByFrame(proteins)
inframe.list <- inFrameList(fusion.frames=partition.fusion,
                            ref.frames=ref.frames)
inframe.list
## $frame1
## LogicalList of length 1
## [["9136-9181"]] TRUE TRUE
##
## $frame2
## LogicalList of length 1
## [["9136-9181"]] FALSE FALSE
##
## $frame3
## LogicalList of length 1
## [["9136-9181"]] FALSE FALSE
```

Each element of the `nostop.list` and `inframe.list` lists are an object of class `LogicalList`, each evaluated from a different reading frame. The `LogicalList` objects are named by the rearrangement identifier. The helper function `inFrameNoStop` combines these to lists to create a single list that is `TRUE` for rearrangements that are in-frame and have no premature stop codons.

```
inframe.nostop <- inFrameNoStop(nostop.list, inframe.list)
inframe.nostop
## $frame1
## LogicalList of length 1
## [["9136-9181"]] TRUE TRUE
##
## $frame2
## LogicalList of length 1
## [["9136-9181"]] FALSE FALSE
##
## $frame3
## LogicalList of length 1
## [["9136-9181"]] FALSE FALSE
```

Finally, `validFusions` returns a list object containing only the fusions that are in-frame and have no premature stops. For each such fusion, the amino acid sequence, CDS of the fusion, partial CDS of the 5-prime and 3-prime transcripts, and the full CDS of the reference transcripts are available.

```
valid.fusions <- validFusions(partition.fusion,
                              cds.list,
                              inframe.nostop,
                              ref.frames)
valid.fusions
## List of length 11
## names(11): amino.acids tum_aa.5p tum_aa.3p ... ref.3p coding_junctions
```

The above data can be summarized in tabular format using the `fusionTable2` function:

```
tab <- fusionTable2(valid.fusions)
head(tab)
## DataFrame with 2 rows and 15 columns
##   gene.5prime gene.3prime    tx.5prime    tx.3prime junction.5prime
##   <character> <character>  <character>  <character>     <character>
## 1       ERBB4       IKZF2 NM_001042599    NM_016260  chr2:213286255
## 2       ERBB4       IKZF2 NM_001042599 NM_001079526  chr2:213286255
##   junction.3prime exons.5prime exons.3prime strand.5prime strand.3prime
##       <character>  <character>  <character>   <character>   <character>
## 1  chr2:213914447   exons: 1-1   exons: 5-7             -             -
## 2  chr2:213914447   exons: 1-1   exons: 5-7             -             -
##   rearrangement.id aa.5prime.start aa.5prime.end aa.3prime.start
##        <character>       <numeric>     <integer>       <numeric>
## 1        9136-9181               1            27             192
## 2        9136-9181               1            27             166
##   aa.3prime.end
##       <integer>
## 1           527
## 2           501
```

Note, this table provides both the genomic coordinates of the fusion (junction.5prime, junction.3prime), but also the portion of the amino acid sequence that was retained by the fusion (aa.5prime.start, aa.5prime.end, aa.3prime.start, aa.3prime.end). In the following code chunk, we construct amino acid ranges for the genes involved in the fusion and the protein domains for these genes in the reference genome.

```
  tumor_aa_ranges <- aa_granges(tab)
  extdata2 <- system.file("extdata", package="trellis")
  up <- readRDS(file.path(extdata2, "uniprot.rds"))
  up2 <- uniprotFeatures(up, tab, strwrap.width=20)
  domain_aa_ranges <- GRanges(up2$hugo, IRanges(up2$start, up2$end),
                              chromosome=up2$seqnames,
                              description=up2$description,
                              short.desc=up2$short.desc,
                              aa_len=up2$aa_len)
  domains <- subsetByOverlaps(domain_aa_ranges, tumor_aa_ranges, type="within")
  domains
## GRanges object with 4 ranges and 4 metadata columns:
##       seqnames     ranges strand | chromosome description  short.desc
##          <Rle>  <IRanges>  <Rle> |   <factor> <character> <character>
##   [1]    IKZF2 [168, 190]      * |       chr2 C2H2-type 3 C2H2-type 3
##   [2]    IKZF2 [196, 219]      * |       chr2 C2H2-type 4 C2H2-type 4
##   [3]    IKZF2 [471, 493]      * |       chr2 C2H2-type 5 C2H2-type 5
##   [4]    IKZF2 [499, 523]      * |       chr2 C2H2-type 6 C2H2-type 6
##          aa_len
##       <integer>
##   [1]       526
##   [2]       526
##   [3]       526
##   [4]       526
##   -------
##   seqinfo: 2 sequences from an unspecified genome; no seqlengths
```