Chimera

Raffaele A Calogero, Matteo Carrara, Marco Beccuti, Francesca Cordero

December 2 2014

1 Introduction

The discovery of novel gene fusions can lead to a better comprehension of cancer progression and development. The emergence of deep sequencing of transcriptome, known as RNAseq, has opened many opportunities for the identification of this class of genomic alterations, leading to the discovery of novel chimeric transcripts in melanomas, breast cancers and lymphomas. Nowadays, various computational approaches have been developed for the detection of chimeric transcripts. Although all of these computational methods allow to detect fusions events, each one producing its own type of output. Outputs generated by fusion finders do not follow any standard structure and, as far as we know, no tools are available to analyse and manipulate, these output. Thus, we have developed *chimera*, which is a package for downstream processing of data obtained by the following fusion detection tools:

1. bellerophontes,
2. deFuse,
3. FusionFinder,
4. FusionHunter,
5. mapSplice,
6. tophat-fusion,
7. FusionMap
8. chimeraScan
9. STAR
10. Rsubread

1.1 Fusion finder tools

We have categorised the fusion detection algorithms into two classes:

1. Fragment-based approach
2. Pseudo-reference based approach

In the fragment based approach tools split the input reads into fragments, which are aligned with respect to a genome or whole transcriptome reference. The mapped fragments are then used to build a list of putative fusion events that are selected using several additional pieces of information or filter steps. This category includes the following tools: *FusionFinder, FusionMap, MapSlice, deFuse, chimeraScan*. Pseudo-reference based approaches are characterised by a combination of candidate fusion events, obtained after the first mapping phase, which are then used to generate a pseudo reference for chimeras detection. The candidate fusion events are filtered following several policies chosen by the authors. *TopHat-Fusion*, *deFuse* and *FusionHunter* are the tools included in this category.

1.2 Example folder

In the example folder of chimera there are a set of example data that can be used to test the chimera functionalities:

```
> dir(paste(find.package(package="chimera"),"/examples/", sep=""))
```

```
 [1] "Edgreen_fm.txt"
 [2] "Edgren_cs.txt"
 [3] "Edgren_df.tsv"
 [4] "Edgren_true.positives.txt"
 [5] "SULF2_ARFGEF2.fa"
 [6] "defuse_TP_FP.txt"
 [7] "edgren.stat.detection.txt"
 [8] "fset_ARFGEF2-SULF2.rda"
 [9] "mcf7.FMFusionReport"
[10] "mcf7_sample_1.fq"
[11] "mcf7_sample_2.fq"
[12] "mcf7_trs_accepted_hits.bam"
[13] "uc002xtx.4-272_uc010zyd.2-988.fa"
[14] "uc002xtx.4-272_uc010zyd.2-988_R1.fastq"
[15] "uc002xtx.4-272_uc010zyd.2-988_R2.fastq"
[16] "uc002xvp.1-243_uc002iyu.4-1031.fa"
[17] "uc002xvp.1-243_uc002iyu.4-1031_R1.fastq"
[18] "uc002xvp.1-243_uc002iyu.4-1031_R2.fastq"
```

Edgreen_fm.txt : it is the output generated by FusionMap on the dataset encompassing all cell lines published by Edgren in Genome Biology.

Edgren_cs.txt : it is the output generated by ChimeraScan on the dataset encompassing all cell lines published by Edgren in Genome Biology.

Edgren_df.tsv : it is the output generated by deFuse on the dataset encompassing all cell lines published by Edgren in Genome Biology.

n.stat.detection.txt : It is a file encompassing all the results obtained by us using different fusion detection tools on edgren dataset. An other set of comparisons can be found on the repository page of fusionCatcher `https://code.google.com/p/fusioncatcher/wiki/comparison`.

7.FMFusionReport : List of fSet objects generated with FusionMap output.

FGEF2-SULF2.rda : example of an fSet object.

ULF2_ARFGEF2.fa : Fasta file for fusion transcripts for the fusion SULF2:ARFGEF2

zyd.2-988_R2.fastq : Fastq to be remapped on SULF2_ARFGEF2.fa to obtain a bam file suitable coverage generation.

uc002iyu.4-1031.fa : Fasta file for fusion transcripts BCAS4:BCAS3.

yu.4-1031_R2.fastq : Fastq to be remapped on BCAS4_BCAS3.fa to obtain a bam file suitable coverage generation.

2 Data structure

The chimera finder output is loaded in R a s list, made of objects of the class *fSet* for each fusion event.

An object of the *fSet* class is characterised by the following slots:

fusionInfo : a list embedding various characteristics of the fusion. The most interesting one is the *SeedCount* slot, which contains the number of reads supporting the fusion junction.

fusionLoc : embedding a GRangesList containing two GRanges objects, one for each gene involved in the fusion. Furthermore each Granges object also contains various informations about the fusion in the *elementMetadata* slot, as *KnownGene* referring to the gene involved in the fusion and the *FusionJunctionSequence*, which is the sequence involved in the fusion.

fusionRNA : embedding a DNAStringSet encompassing the possible fusion events obtainable using the gene isoforms encompassing the exons involved in the fusion.

fusionGA : a GAlignments object containing all the positions of the reads mapping over the fusion, which is useful to generate coverage information supporting the fusion.

2.1 fSet methods

The method *fusionData*, given a fSet object, returns information for a fusion, depending on the tool used only some of the fusion description information are available (fusionTool: used for the analysis, SeedCount: number of reads supporting the fusion junction, RescuedCount: number of reads supporting the fusion globally, i.e. spanning and encompassing reads, SplicePattern: spice patter used, FusionGene: transcripts involved in the fusion, frameShift: presence of a frame shift in the fusion event) The method *fusionGRL*, given a fSet object, returns the GRangesList with the information on the genomic location of the fusion boundaries for the two genes involved in the fusion. The method *fusionRNA* given a fSet object, returns the DNAStringSet of the putative fusions encompassing the exons involved in the fusion. The method *addDNA*, given a fSet object, allows to add the DNAStringSet of the putative fusions to an fSet object. The DNAStringSet of the putative fusions can be generated using the function *chimeraSeqs*, see section below. The method *fusionGA* given a fSet object, returns the GAlignments object of the putative fusions encompassing the exons involved in the fusion. The method *addGA*, given a fSet object, allows to add the GAlignments of the putative fusions to an fSet object. The DNAStringSet of the putative fusions can be generated using the function *tophatRun*, see section below. The function *addGA* sorts, indexes and loads the bam generated by TopHat as a GAlignments object.

```
> #creating a fusion report from output of fusionMap
> library(chimera)
> tmp <- importFusionData("fusionmap", paste(find.package(package="chimera"),
+ "/examples/mcf7.FMFusionReport", sep=""), org="hg19")
> #extracting the fSet object for one of the fusions
> myset <- tmp[[13]]
> #constructing the fused sequence(s)
> trs <- chimeraSeqs(myset, type="transcripts")
> #adding the sequences to the fSet object
> myset <- addRNA(myset , trs)
> #extracting sequences from an fSet object
> tmp.seq <- fusionRNA(myset)
> #adding reads mapped on the fusion generated using tophatRun function
> myset <- addGA(myset, paste(path.package(package="chimera"),
+ "/examples/mcf7_trs_accepted_hits.bam",sep=""))
> #extracting the GAlignments from an fSet object
> ga <- fusionGA(myset)
```

3 Functions

The function *importFusionData* allows to import in a list outputs generated by bellerophontes, defuse, fusionfinder, fusionhunter, mapsplice, tophat-fusion, fusionmap, chimerascan.

The function *supportingReads* allows to extract the number of reads supporting each of the detected fusions. It is notable that the same fusion gene is detected with a different number of supporting read depending on the fusion tool used. Furthermore, each tool detects a different number of fusions when querying the same data set. same apply to the supporting reads.

```
> supporting.reads <- supportingReads(tmp, fusion.reads="encompassing")
> supporting.reads

 [1]  1  1  4 19  1  1  1  1  1  1  1  1 18
```

The function *fusionName* allows to extract fusion names from a list of fSet objects.

```
> fusion.names <- fusionName(tmp)
> fusion.names

 [1] "HMGN2:ESYT1"                   "CC2D1B:DTYMK"
 [3] "NOS1AP:C1orf226"               "GREB1:GREB1"
 [5] "RYBP:YAF2"                     "SLC30A5:AZIN1"
 [7] "chr6:30691425-30691455:KRT80" "EEF1A1:GHITM"
 [9] "HNRNPK:AATF"                   "NDUFA1:SYNJ2BP-COX16"
[11] "TASOR2:TASOR2"                 "YLPM1:ITPK1"
[13] "SULF2:ARFGEF2"
```

The function *chimeraSeqs* allows to generate the chimera nucleotide sequence. The output is a DNAStringSet object encompassing the fusions generated using all the isoforms for each gene involved in the fusion.

```
> myset <- tmp[[13]]
> trs <- chimeraSeqs(myset, type="transcripts")
```

The function *subreadRun* allows to map reads to a chimera sequence set generated by *chimeraSeqs*. The function produces a standard output bam file (accepted_hits.bam). The bam produced by this remapping on putative fusions can be used to generate the coverage plot for all the fused constructs.

Using *write.XStringSet* function from Biostring package DNAStringSet can be saved in fast format. The function *tophatRun* maps reads to a chimera sequence set, eg. *trs*.

```
> #the DNAStringSet of transcript fusions sequences is saved as fast file
> #write.XStringSet(trs, paste("SULF2_ARFGEF2.fa",sep=""), format="fasta")
> if (require(Rsubread))
```

```
+ {
+     subreadRun(ebwt=paste(find.package(package="chimera"),"/examples/SULF2_ARFGEF2
+     input1=paste(find.package(package="chimera"),"/examples/mcf7_sample_1.fq",sep=
+     input2=paste(find.package(package="chimera"),"/examples/mcf7_sample_2.fq",sep=
+     outfile.prefix="accepted_hits", alignment="se", cores=1)
+ }

        ==========     _____ _    _ ____  _____  ______          _____
        =====         / ____| |  | |  _ \|  __ \|  ____|   /\   |  __ \
          =====      | (___ | |  | | |_) | |__) | |__     /  \  | |  | |
            ====      \___ \| |  | |  _ <|  _  /|  __|   / /\ \ | |  | |
              ====    ____) | |__| | |_) | | \ \| |____ / ____ \| |__| |
        ==========   |_____/ \____/|____/|_|  \_\______/_/    \_\_____/
       Rsubread 2.0.0

//================================== setting ==================================\\
||                                                                            ||
||                Index name : chimeraDB_Tue-Oct-29-20-37-44-2019             ||
||               Index space : base space                                     ||
||               Index split : no-split                                       ||
||         Repeat threshold : 100 repeats                                     ||
||              Gapped index : no                                             ||
||                                                                            ||
||       Free / total memory : 22.7GB / 31.3GB                                ||
||                                                                            ||
||               Input files : 1 file in total                                ||
||                             o SULF2_ARFGEF2.fa                             ||
||                                                                            ||
\\============================================================================//

//================================== Running ==================================\\
||                                                                            ||
|| Check the integrity of provided reference sequences ...                    ||
|| No format issues were found                                                ||
|| Scan uninformative subreads in reference sequences ...                     ||
|| Estimate the index size...                                                 ||
||  149%,   0 mins elapsed, rate=447.2k bps/s                                 ||
||  158%,   0 mins elapsed, rate=472.1k bps/s                                 ||
||  166%,   0 mins elapsed, rate=487.5k bps/s                                 ||
||  174%,   0 mins elapsed, rate=511.9k bps/s                                 ||
||  183%,   0 mins elapsed, rate=536.3k bps/s                                 ||
||  191%,   0 mins elapsed, rate=560.7k bps/s                                 ||
||  199%,   0 mins elapsed, rate=574.2k bps/s                                 ||
```

```
||  1.3 GB of memory is needed for index building.                              ||
||  Build the index...                                                          ||
||     8%,   0 mins elapsed, rate=4.5k bps/s                                    ||
||    16%,   0 mins elapsed, rate=9.0k bps/s                                    ||
||    24%,   0 mins elapsed, rate=13.6k bps/s                                   ||
||    33%,   0 mins elapsed, rate=18.0k bps/s                                   ||
||    41%,   0 mins elapsed, rate=22.4k bps/s                                   ||
||    49%,   0 mins elapsed, rate=26.9k bps/s                                   ||
||    58%,   0 mins elapsed, rate=31.3k bps/s                                   ||
||    66%,   0 mins elapsed, rate=35.6k bps/s                                   ||
||  Save current index block...                                                 ||
||   [ 0.0% finished ]                                                          ||
||   [ 10.0% finished ]                                                         ||
||   [ 20.0% finished ]                                                         ||
||   [ 30.0% finished ]                                                         ||
||   [ 40.0% finished ]                                                         ||
||   [ 50.0% finished ]                                                         ||
||   [ 60.0% finished ]                                                         ||
||   [ 70.0% finished ]                                                         ||
||   [ 80.0% finished ]                                                         ||
||   [ 90.0% finished ]                                                         ||
||   [ 100.0% finished ]                                                        ||
||                                                                              ||
||                        Total running time: 0.0 minutes.                      ||
||       Index chimeraDB_Tue-Oct-29-20-37-44-2019 was successfully built!       ||
||                                                                              ||
\\==============================================================================//


        ==========     _____ _    _ ____  _____  ______          _____
        =====         / ____| |  | |  _ \|  __ \|  ____|   /\   |  __ \
          =====      | (___ | |  | | |_) | |__) | |__     /  \  | |  | |
            ====      \___ \| |  | |  _ <|  _  /|  __|   / /\ \ | |  | |
              ====    ____) | |__| | |_) | | \ \| |____ / ____ \| |__| |
        ==========   |_____/ \____/|____/|_|  \_\______/_/    \_\_____/
       Rsubread 2.0.0

//================================ setting ==================================\\
||                                                                            ||
|| Function      : Read alignment (RNA-Seq)                                   ||
|| Input file    : tmp_Tue-Oct-29-20-37-47-2019.fastq                         ||
|| Output file   : accepted_hits.bam (BAM)                                    ||
```

```
|| Index name    : chimeraDB_Tue-Oct-29-20-37-44-2019                    ||
||                                                                        ||
||                    ----------------------------------                  ||
||                                                                        ||
||                              Threads : 1                               ||
||                         Phred offset : 33                              ||
||                            Min votes : 3 / 10                          ||
||                       Max mismatches : 3                               ||
||                     Max indel length : 5                               ||
||            Report multi-mapping reads : yes                            ||
|| Max alignments per multi-mapping read : 1                              ||
||                                                                        ||
\\========================================================================//

//================ Running (29-Oct-2019 20:37:47, pid=25949) ================\\
||                                                                        ||
|| Check the input reads.                                                 ||
|| The input file contains base space reads.                              ||
|| Initialise the memory objects.                                         ||
|| Estimate the mean read length.                                         ||
|| The range of Phred scores observed in the data is [0,34]               ||
|| Create the output BAM file.                                            ||
|| Check the index.                                                       ||
|| Init the voting space.                                                 ||
|| Global environment is initialised.                                     ||
|| Load the 1-th index block...                                           ||
|| The index block has been loaded.                                       ||
|| Start read mapping in chunk.                                           ||
||                                                                        ||
||                          Completed successfully.                       ||
||                                                                        ||
\\==================================    ==================================//

//==============================   Summary ================================\\
||                                                                        ||
||                 Total reads : 1,386                                    ||
||                      Mapped : 1,381 (99.6%)                            ||
||              Uniquely mapped : 1,381                                   ||
||                Multi-mapping : 0                                       ||
||                                                                        ||
||                    Unmapped : 5                                        ||
||                                                                        ||
```

```
||                      Indels : 0                                          ||
||                                                                          ||
||                Running time : 0.0 minutes                                ||
||                                                                          ||
\\==========================================================================//

[1] "accepted_hits_mapped.bam"
```

The function *filterList* allows to filter a list of fSet objects on the basis of supporting reads or fusion names or presence of intronic sequence in the fusion. The rational of filtering on the basis of supporting reads is that biological effect also depends on the amount of the expressed mRNA, thus highly expressed fusions, i.e. fusions with an high number of junction-spanning reads, might have a more important role in cancer physiology. On the other hand the presence e of an intron in a fusion generates a very large transcript, which do not produce in frame transcripts. Furthermore it is possible to remove read-through events, i.e. fusion in which different exons of the same gene are recognised as a fusion being separated by extremely long introns. It is also possible to retain only fusions in which only annotated names are considered, eg. PID1:TP53 is retained and PID1:chr2:133038625-133038655 is removed.

```
> tmp1 <- filterList(tmp, type="fusion.names", fusion.names[c(1,3,7)])

filtering by fusion names needs to pass to the method vector of character names

> tmp2 <- filterList(tmp, type="spanning.reads", 2)

filtering by supporting reads needs to pass to the method a numerical reads threshold

> #tmp3 <- filterList(tmp, type="intronic")
> tmp4 <- filterList(tmp, type="annotated.genes")
> tmp5 <- filterList(tmp, type="read.through")
```

Coverage can be visualised with the function *plotCoverage*. Below three examples of the visualisation of a fusion based on exon coverage, junctions coverage or coverage at the fusion boudaries.

```
> tmp <- importFusionData("fusionmap", paste(find.package(package="chimera"),
+ "/examples/mcf7.FMFusionReport", sep=""), org="hg19")
> fusion.names <- fusionName(tmp)
> myset <- tmp[[13]]
> trs <- chimeraSeqs(myset, type="transcripts")
> myset <- addRNA(myset , trs)
> tmp.seq <- fusionRNA(myset)
> myset <- addGA(myset, paste(path.package(package="chimera"),
```

```
+ "/examples/mcf7_trs_accepted_hits.bam",sep=""))
> pdf("coverage1.pdf")
> plotCoverage(myset, plot.type="exons", col.box1="red",
+ col.box2="green", ybox.lim=c(-4,-1))
> dev.off()

null device
          1

> pdf("coverage2.pdf")
> plotCoverage(myset, plot.type="junctions", col.box1="red",
+ col.box2="yellow", ybox.lim=c(-4,-1))
> dev.off()

null device
          1

> pdf("coverage3.pdf")
> plotCoverage(myset, junction.spanning=100, fusion.only=TRUE, col.box1="red",
+ col.box2="yellow", ybox.lim=c(-4,-1))
> dev.off()

null device
          1
```

The coverage at the fusion boundary, can be generated using the parameter fusion.only sets on TRUE. To extend the region around the fusion location, it is possible to change the number of nucleotides spanning around the fusion location, using the parameter junction.spanning, which is set to 20 nts as default.

It is also possible to reconstruct the protein sequence involved in the fusion using the function *fusionPeptides*. The function a AAStringSet encompassing: fusion sequence, peptide from p1 and peptide from p2. In case the peptides are not in frame the AAStringSet will not contain the fusion sequence.

```
> mypeps <- fusionPeptides(chimeraSeq.output=trs)

None of the two genes is coding

> mypeps

NULL
```

Furthermore, the output of the *fusionPeptides*, provides a DNAStringSet encompassing the fusion transcript that can be used to be upload in Primer3, `http://simgene.com/Primer3`, to design the PCR primers to validated the fusion event or to obtain the region encompassing the fusion that can be cloned and sequenced with Sanger method.

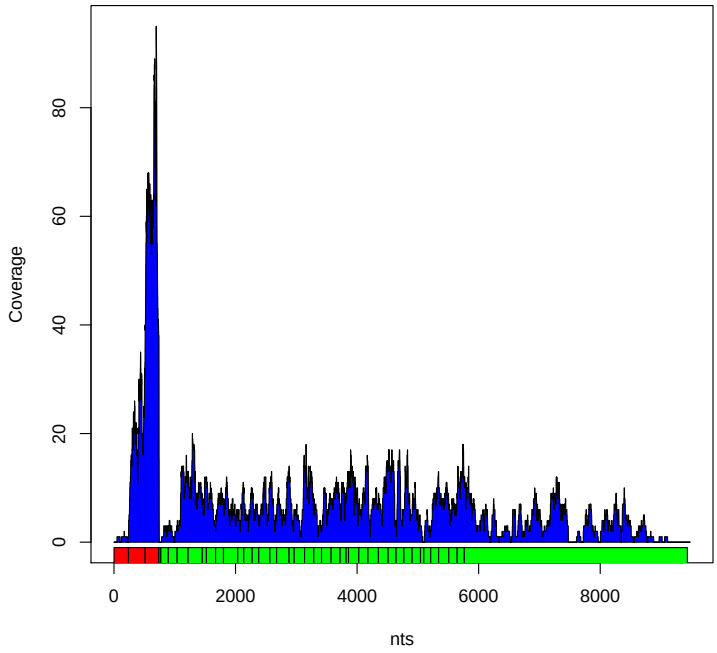

Figure 1: Exons coverage plotted with respect to the fused transcript structure. Exons of gene 1 are shown in red, exons of gene 2 are shown in yellow.

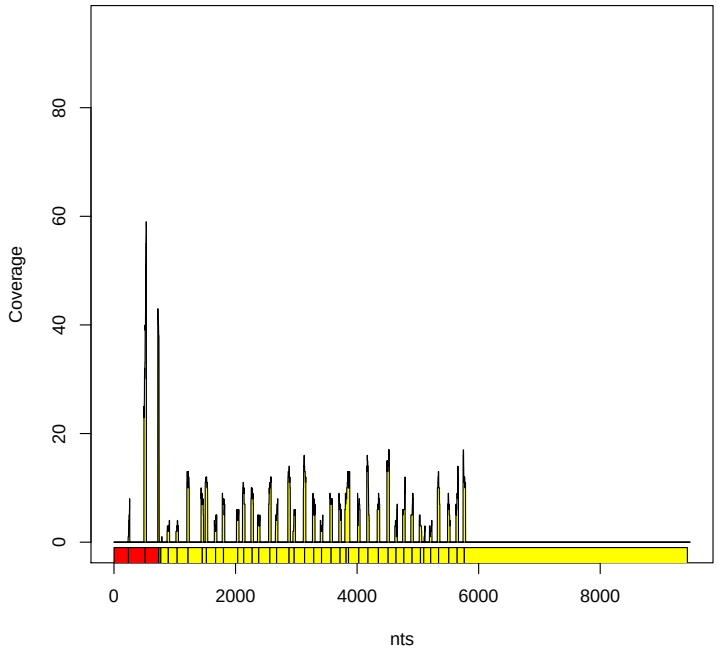

Figure 2: Exon/Exon junctions coverage plotted with respect to the fused transcript structure. Exons of gene 1 are shown in red, exons of gene 2 are shown in yellow.

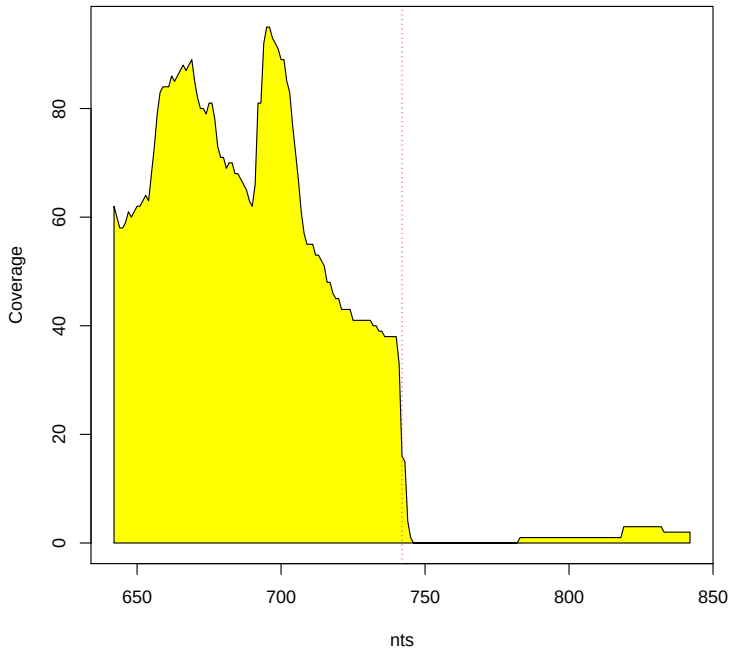

Figure 3: Fusion boundaries coverage plotted at the level at the fusion location. In this case it is clear that reads spanning over the fusion boundary (dashed red line) are not detected

4 Fusions validation

The tools used for the detection of fusion events are error prone and many of the detected fusions are false positive events. Since experimental validation is a long and realatively expensive procedure it is important to reduce as much as possible the number of false positive events. For this reason we have deviced a validation procedure based on the de novo assembly. We use the GapFiller tools, [Nadalin et al. BMC Bioinformatics. 2012;13 Suppl 14:S8], to perform this validation. GapFiller is seed-and-extend local assembler, which carefully detects reliable overlaps and operates clustering similar reads in order to reconstruct the missing part between the two ends of the same insert. Thus is perfect to see if a set of reads mapping over a putative fusion transcript are also able to de novo reconstruct the fusion junction. Below are shown the steps we suggest for the identification of a robust set of fusion events.

```
#importing fusion from your favorite program, e.g. ChimeraScan.
fusions <- importFusionData("chimerascan", 'chimeras.bedpe', org="hg19")
```

On the basis of our work [Carrara et al BMC Bioinformatics. 2013;14 Suppl 7:S2.; Carrara et al. Biomed Res Int. 2013;2013:340620] the best performing fusion detection tool, between those tested, is ChimeraScan. It detectes most of the true positive events also in the correct orientation, but brings a significant number of false positive. One of the experimental approaches we used to moderate the number of false positive was using a directional sequence protocol. Running chimerascan in a strand oriented mode significantly reduced the false positive events.

```
#retain only fusions supported by at least 1 reads
#spanning over the fusion junction.
fusions <- filterList(fusions, type="spanning.reads", 1)
#retain only fusions encompassing coding annotated genes
fusions <- filterList(fusions, type="annotated.genes")
#filter fusion events to remove those including intronic sequences.
fusions <- filterList(fusions, type="intronic", parallel=T)
```

The above mentioned filtering procedure reduce significantly the number of fusions

```
#Confirming by de novo assembly that reads are able to reconstruct the fusion juncti
#reconstructing fusions
trsx <- chimeraSeqSet(fusions, parallel=T)
#save the fusions as fasta file
writeXStringSet(trsx, paste("fusions.fa",sep=""), format="fasta")
#remap the reads initially used to detect the fusion events over
#the putative fusions, i.e. fusions.fa
subreadRun(ebwt="fusions.fa", input1="sample_R1.fastq", input2="sample_R2.fastq",
outfile.prefix="accepted_hits", alignment="se", cores=64)
```

The output bam, accepted_hits.bam, contains only the mapped reads

```
#Validating fusion junction results with GapFiller
tmp <- gapfillerWrap(chimeraSeqSet.out=trsx, bam="accepted_hits.bam", parallel=c(FAL
```

The function *gapfillerRun* returns a list of three objects:

contigs : a PairwiseAlignments object.

junction.contigs : a DNAStringSet encompassing the sequences present in the contigs object

fusion : a a DNAStringSet object encompassing the fusion transcript.

The list is returned only in case that some of de novo assemblies cover the breakpoint junction. The final step is the confirmation of a unique open reading frame encompassing the coding region of the two fused genes.

```
tmpx <- lapply(trsx,fusionPeptides)
```

5 Oncofuse

Oncofuse is a naive bayesian classifier designed to predict the oncogenic potential of fusion genes found by Next-Generation Sequencing in cancer cells (Shugay et al. Bioinformatics 2013,29:2539-2546). Its goal is to identify those fusion sequences with higher probability of being driver than passenger events. The function *oncofuseInstallation* installs the java Oncofuse program in a folder inside chimera path. The function *oncofuseRun* will execute Oncofuse using as input a list of fSets. The output provided is a dataframe with the output of Oncofuse. Of particular interest are the dataframe fields: DRIVER PROB which provides a pvalue for the probability that the fusion will be deleterious and the annotations fields, DOMAINS RETAINED and DOMAINS BROKEN providing information on the type of protein domains affected by the fusion event.

```
tmp <- importFusionData("fusionmap", paste(find.package(package="chimera"),"/example
installOncofuse()
of.out <- oncofuseRun(tmp, tissue="EPI")
```

6 Supported Fusion detection tools

6.1 FusionMap

FusionMap aligns fusion reads directly to the genome without prior knowledge of potential fusion regions. FusionMap can detect fusion events in both single- and paired-end datasets from either RNA-Seq or gDNA-Seq studies and characterize fusion junctions at base-pair resolution: `http://www.omicsoft.com/fusionmap` The software splits reads

into smaller fragments and finds fusion candidates aligning these fragments to genes annotated on genomic reference. The read alignment is performed by GSPN algorithm, integrated in the tool, with up to 2 base mismatches of tolerance. Two 25 bp seeds at each side of unmapped read are extracted and aligned to the reference. Putative fusions are reported only if both seeds align. Fusions events characterized by fusion boundaries distant less than 5 bp are aggregated and used for junction refinement. A scoring system based on canonical splicing patterns is then used to define the position of the fusion boundary. False positives are removed using the following four steps: (i) reads with break point score above a threshold are removed; (ii) fusions nearer than 5kb (partial removal of read-through events) are ignored; (iii) the fusion source sequences are concatenated, creating a pseudo-reference to be used to align unmapped reads. Fusion candidates with no reads aligned to the pseudo-reference are removed; and (iv) fusions with less than 2 reads aligned on distinct regions (PCR artifact removal) are removed.

6.2 FusionHunter

FusionHunter is an open-source software tool, which reliably identifies fusion transcripts from transcriptional analysis of paired-end RNA-seq `http://bioen-compbio.bioen.illinois.edu/FusionHunter/`. FusionHunter maps the input paired-end reads against a reference genome using Bowtie. The mapped reads are used to identify the fusion candidates. These candidates are combined to generate a pseudo reference used to identify junction-spanning reads. Two genomic regions are marked as fusion candidate if there are two transcripts enriched by input reads and the fusion junction between them is supported by at least two different paired-end reads. Each fusion candidate composed by two genes sharing significant homology is removed. Candidates are analyzed to estimate their orientation and are concatenated into a pseudo-reference. Unmapped reads are split in segments and mapped on the pseudo-reference. If one segment is correctly aligned, the tool searches for the nearest canonical splicing junction and aligns the other part of the original read with that region. Several filters are applied to (i) remove reads aligned on the break point if anchored to one gene with less than 6 bp; (ii) remove fusion events with no reads aligned on the break point: (iii) keep only one read in case multiple reads are stacked (PCR artifact removal); and (iv) remove read-through events if not available in the human EST database.

6.3 FusionFinder

FusionFinder is a Perl-based software designed to automate the discovery of candidate gene fusion partners from single-end (SE) or paired-end (PE) RNA-Seq read data. `http://bioinformatics.childhealthresearch.org.au/software/fusionfinder/`. FusionFinder splits reads into fragments and reports fusion candidates when these fragments align to genes annotated on genomic reference. The main differences with respect to FusionMap are the tools used for alignment and the filter implementation. Bowtie is

used to align reads with respect to coding reference transcriptome. Two fragments at each side of unmapped read, namely pseudo-Paired-End (PE) reads, are extracted. The pseudo-PE reads are aligned with Bowtie with respect to the coding reference transcriptome considering up to two mismatches. A list of exons involved in the fusion can be reported by the identification of the closest ENSEMBL exons. Finally, multiple filtering steps are used to refine the results: (i) pairs of seeds mapping on the same gene are removed; (ii) pairs on the same chromosome but on opposite strands (antisense transcript removal) are removed; (iii) the pairs are mapped on the genome. Pairs mapped at a distance higher than read length are removed; and (iv) all possible artifacts caused by sequence similarity are also removed.

6.4 deFuse

deFuse is a computational method for fusion discovery in tumor RNA-Seq data. Unlike existing methods that use only unique best-hit alignments and consider only fusion boundaries at the ends of known exons, deFuse considers all alignments and all possible locations for fusion boundaries. `http://sourceforge.net/apps/mediawiki/defuse/index.php?title=DeFuse`. deFuse uses reads pairs with discordant alignments to define putative fusion events on the basis of two conditions: the region covered by different reads must overlap and the shift between overlapping reads must be coherent with the fragment length. Each paired-end read with discordant alignment is then assigned to a putative fusion in order to minimize the number of reported events. For each putative fusion an estimation of fusion boundary position is used to detect encompassing reads and to map the fusion boundary position at nucleotide level. This information is also used to discard read pairs aligned located at a distance not compatible with the expected distribution of sequenced fragments distance.

6.5 mapSplice

MapSplice can be applied to both short (<75bp) and long reads (>75bp). MapSplice is not dependent on splice site features or intron length, consequently it can detect novel canonical as well as non-canonical splices. MapSplice leverages the quality and diversity of read alignments of a given splice to increase accuracy. `http://www.netlab.uky.edu/p/bioinfo/MapSplice`. MapSplice starts splitting each read in a set of consecutive segments, having size smaller than half of the read size. The exon alignment of segments is performed using Bowtie and BWA or SOAP2, BFAST and MAQ, with an input specified mismatch tolerance. For each read, MapSplice aligns segments not mapped in the previous step, exploiting the information derived by the other aligned segments. Finally, the splice junction quality of fusion events is evaluated according with two statistical measures: the ?anchor significance?, determined by an alignment that maximizes significance as a result of long anchors on each side of the splice junction, and the entropy measured by the diversity of splice junction positions.

6.6 TopHat-fusion

TopHat-Fusion is an algorithm designed to discover transcripts representing fusion gene products, which result from the breakage and re-joining of two different chromosomes, or from rearrangements within a chromosome. TopHat-Fusion is an enhanced version of TopHat, an efficient program that aligns RNA-seq reads without relying on existing annotation. Because it is independent of gene annotation, TopHat-Fusion can discover fusion products deriving from known genes, unknown genes and unannotated splice variants of known genes. `http://tophat.cbcb.umd.edu/fusion_index.html`. TopHat-Fusion uses Bowtie to detect all reads aligning entirely within exons, and creates a set of partial exons from these alignments. Then, hypothetical intron boundaries are created between the partial exons, and Bowtie is used to re-align the initially unmapped reads and find those that define introns. Each read is split into segments of 25 bp and each segment mapped on the genome. Putative fusions are reported if segments map in a way consistent with fusions (using TopHat with relaxed parameters). Several filters are applied after candidate definition to (i) remove candidate fusions on multi-copy genes or repetitive sequences; (ii) remove reads anchored with less than 13 bp on either side of the fusion; and (iii) remove candidate fusions from regions nearer than 100 kb (read-through events removal). TopHat-Fusion trims 22 bp segments flanking each fusion point, it constructs spliced fusion contigs and builds an index for them. Each segment is re-mapped on the new contigs and the results are stitched together to produce the full read alignment. The algorithm evaluates then the contradicting reads, i.e. the reads fully mapped on a single part of the fusion and overlapping the fusion boundary. Finally, the tool removes fusion events after the junction definition, if both sides are not annotated.

6.7 Bellerophontes

Bellerophontes is a fully automated framework for the detection of novel fusion transcripts in paired end RNA-Seq data `http://eda.polito.it/bellerophontes/index.html`. It detects putative fusion genes by searching those reads that discordantly matches on different genes. Then, the tool applies several modular filters in order to select those fused genes matching an accurate gene fusion model based on experimental evidences reported in recent literature. Bellerophontes runs on top of TopHat and Cufflinks tools. The analysis is based on the results of TopHat alignment and Cufflinks transcript isoform detection.

6.8 chimeraScan

ChimeraScan uses Bowtie to align paired-end reads to a combined genome-transcriptome reference. read pairs that could not be aligned concordantly are trimmed into smaller segments (default = 25 bp) and realigned. Trimming increases the chance that neither read alignment spans a chimeric junction, thereby improving sensitivity for nominating chimeras. the trimmed alignments are scanned for evidence of discordant read pairs, or

reads that align to distinct references or distant genomic locations (as determined by the fragment size range) of the same reference. Reads aligning to overlapping transcripts are not considered discordant. ChimeraScan clusters the discordant reads and produces a list of putative transcript pairs that serve as chimera candidates. after spanning reads are incorporated, ChimeraScan filters chimeras with few supporting reads and chimeras with fragment sizes far outside the range of the distribution. When isoforms of the same gene support a fusion ChimeraScan only retains the isoform(s) with highest coverage. ChimeraScan produces a tabular text file describing each chimera.

6.9 STAR

Spliced Transcripts Alignment to a Reference (STAR) software is based on a previously un-described RNA-seq alignment algorithm which utilizes sequential maximum mappable seed search in uncompressed suffix arrays followed by seed clustering and stitching procedure. STAR can discover non-canonical splices and chimeric (fusion) transcripts. If the best scoring alignment window does not cover the entire read, STAR reports chimeric connections to the other windows that cover portions of the read not covered by the main window. These chimeric connections between windows can span long distance on the same strand, or different strands on the same chromosome, or different chromosomes. The *importFusionData* allows to import the file Chimeric.out.junction. Furthermore, STAR can be installed in the chimera folder using the function *starInstallation*, and STAR can be run using *starRun*. Data import can be parallelized with option parallel sets to TRUE.

6.10 Rsubread

Rsubread package provide the function *subjunc* that allows with the parameter reportAllJunctions set to TRUE allow the detection of fusions which are reported with the extension fusion.txt. To optimize data analysis we suggest to run first STAR in paired end mode adding the parameter outReadsUnmapped Fastx, which save unmapped reads with estension mate1 and mate2. The unmapped reads are then passed to the *subjunc* function of Rsubread package to generate fusion data. Data import can be parallelized with option parallel sets to TRUE.

```
#########Example of fusion detection with Rsubread.
##STEP1: example of shell script to run STAR to remove all paired-end mapping
#not involving canonical junctions.

#! /bin/sh
STAR_FOLDER=/somewhere/STAR_2.3.1n/
STAR_GENOME=/somewhere/hg9.star
FASTQ1=../myfastq_R1.fastq
```

```
FASTQ2=../myfastq_R2.fastq
OUTFILE=mydata_
THREADS=64
$STAR_FOLDER/STARstatic --genomeDir $STAR_GENOME --readFilesIn $FASTQ1 $FASTQ2 --run

##STEP2: On the unmapped fastq generated from STAR mapping script (above)
library(Rsubread)
subjunc("/home/calogero/bin/genome/rsubread.hg19/hg19", "spike4_Unmapped.out.mate1",
tmp <- importFusionData("rsubread","spike4_subread.bam.fusion.txt", org="hg19", min.

##STEP3: Importing fusions in R
tmp <- importFusionData("rsubread","spike4_subread.bam.fusion.txt", org="hg19", min.
```

7 example

In the following table are shown the data obtained, during chimera development, using some of the fusion detection tools supported by chimera , on the datasets published by Edgen. The different cell lines fastq were combined in a unique fastq to generate the benchmark dataset and runs were done using default parameters.

```
> library(xtable)
> tp <- read.table(paste(find.package(package="chimera"),"/examples/edgren.stat.dete
> my.table <- xtable(tp, caption = 'Edgren validated fusions discovered by tools sup
> print(my.table)
```

```
% latex table generated in R 3.6.1 by xtable 1.8-4 package
% Tue Oct 29 20:38:03 2019
\begin{table}[ht]
\centering
\begin{tabular}{rlllllllllllll}
  \hline
 & cell.line & Edgren.paper & location.donor.gene & location.acceptor.gene & trueposit
  \hline
1 & BT-474 & Genome Biol. 2011;12(1):R6. & chr17:35441927-35766902 & chr17:37366789-37
  2 & SK-BR-3 & Genome Biol. 2011;12(1):R6. & chr5:139781399-139852062 & chr5:14124221
  3 & MCF-7 & Genome Biol. 2011;12(1):R6. & chr20:47538275-47653230 & chr20:46286150-4
  4 & MCF-7 & Genome Biol. 2011;12(1):R6. & chr20:49411467-49493714 & chr17:58755172-5
  5 & KPL-4 & Genome Biol. 2011;12(1):R6. & chr19:571277-583493 & chr19:13106584-13209
  6 & SK-BR-3 & Genome Biol. 2011;12(1):R6. & chr14:99977603-100070727 & chr14:9987614
  7 & BT-474 & Genome Biol. 2011;12(1):R6. & chr20:34213953-34252878 & chr20:43803540-
  8 & SK-BR-3 & Genome Biol. 2011;12(1):R6. & chr20:47662783-47713497 &  & CSE1L:ENSG0
```

```
  9 & SK-BR-3 & Genome Biol. 2011;12(1):R6. & chr17:76670130-76778376 & chr8:11765705
  10 & SK-BR-3 & Genome Biol. 2011;12(1):R6. & chr20:37590981-37668366 & chr20:3295104
  11 & BT-474 & Genome Biol. 2011;12(1):R6. & chr20:61536350-61569304 & chr20:3661142
  12 & BT-474 & Genome Biol. 2011;12(1):R6. & chr3:33038100-33138694 & chr3:32433163-3
  13 & BT-474 & Genome Biol. 2011;12(1):R6. & chr13:113951469-113977741 & chr13:11362
  14 & SK-BR-3 & Genome Biol. 2011;12(1):R6. & chr20:34256610-34287287 & chr20:4724079
  15 & KPL-4 & Genome Biol. 2011;12(1):R6. & chr9:139388896-139440238 & chr9:134000981
  16 & KPL-4 & Genome Biol. 2011;12(1):R6. & chr12:80167343-80329235 & chr2:1103003741
  17 & BT-474 & Genome Biol. 2011;12(1):R6. & chr20:56884771-56942563 & chr19:1718659
  18 & SK-BR-3 & Genome Biol. 2011;12(1):R6. & chr17:38465423-38513895 & chr8:7942833
  19 & BT-474 & Genome Biol. 2011;12(1):R6. & chr17:57970443-58027786 & chr17:4700745
  20 & MCF-7 & Genome Biol. 2011;12(1):R6. & chr17:57970443-58027786 & chr17:57784863-
  21 & BT-474 & Genome Biol. 2011;12(1):R6. & chr17:57187308-57232800 & chr17:3486891
  22 & BT-474 & Genome Biol. 2011;12(1):R6. & chr17:37793333-37820454 & chr17:3779333
  23 & SK-BR-3 & Genome Biol. 2011;12(1):R6. & chr3:4402829-4508966 & chr3:37094117-37
  24 & SK-BR-3 & Genome Biol. 2011;12(1):R6. & chr8:125500735-125551329 & chr17:380608
  25 & BT-474 & Genome Biol. 2011;12(1):R6. & chr20:56964175-57026156 & chr17:3791396
  26 & SK-BR-3 & Genome Biol. 2011;12(1):R6. & chr8:124084920-124164392 & chr8:8154068
  27 & BT-474 & Genome Biol. 2011;12(1):R6. & chr20:45838381-45985474 & chr20:3404322
   \hline
\end{tabular}
\caption{Edgren validated fusions discovered by tools supported by chimera}
\end{table}
```

Further comparison results are available at `https://code.google.com/p/fusioncatcher/wiki/comparison` The package comes with a set of data, present in the example folder, representing all the information needed to create an fSet object for the fusion SULF2:ARFGEF2 detected by FusionMap in the MCF7 dataset published by Edgren et al. Genome Biology 2011, 12:R6.

```
mcf7.FMFusionReport is the output of FusionMap and can be imported using   importFus
mcf7_sample_1.fq and mcf7_sample_2.fq are fastq files to be used for  coverage estim
using tophatRun on  the fusion SULF2_ARFGEF2.fa.
mcf7_trs_accepted_hits.bam, mcf7_trs_accepted_hits.bam_sorted.bam and
mcf7_trs_accepted_hits.bam_sorted.bam.bai are the output of tophatRun
and they can be used to generate the GAlignments object to be loaded in the fSet obj
using addGA function.
```

8 R-Bioconductor information

```
> sessionInfo()
```

```
R version 3.6.1 (2019-07-05)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.04.3 LTS

Matrix products: default
BLAS:   /home/biocbuild/bbs-3.10-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.10-bioc/R/lib/libRlapack.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=en_US.UTF-8
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [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

attached base packages:
[1] stats4    parallel  stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
 [1] xtable_1.8-4
 [2] Rsubread_2.0.0
 [3] chimera_1.28.0
 [4] Homo.sapiens_1.3.1
 [5] org.Hs.eg.db_3.10.0
 [6] GO.db_3.10.0
 [7] OrganismDbi_1.28.0
 [8] TxDb.Hsapiens.UCSC.hg19.knownGene_3.2.2
 [9] GenomicFeatures_1.38.0
[10] BSgenome.Hsapiens.UCSC.hg19_1.4.0
[11] BSgenome_1.54.0
[12] rtracklayer_1.46.0
[13] AnnotationDbi_1.48.0
[14] GenomicAlignments_1.22.0
[15] SummarizedExperiment_1.16.0
[16] DelayedArray_0.12.0
[17] BiocParallel_1.20.0
[18] matrixStats_0.55.0
[19] Rsamtools_2.2.0
[20] Biostrings_2.54.0
[21] XVector_0.26.0
```

```
[22] GenomicRanges_1.38.0
[23] GenomeInfoDb_1.22.0
[24] IRanges_2.20.0
[25] S4Vectors_0.24.0
[26] Biobase_2.46.0
[27] BiocGenerics_0.32.0

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.2             lattice_0.20-38        prettyunits_1.0.2
 [4] assertthat_0.2.1       zeallot_0.1.0          digest_0.6.22
 [7] BiocFileCache_1.10.0   R6_2.4.0               backports_1.1.5
[10] RSQLite_2.1.2          httr_1.4.1             pillar_1.4.2
[13] zlibbioc_1.32.0        rlang_0.4.1            progress_1.2.2
[16] curl_4.2               blob_1.2.0             Matrix_1.2-17
[19] stringr_1.4.0          RCurl_1.95-4.12        bit_1.1-14
[22] biomaRt_2.42.0         compiler_3.6.1         pkgconfig_2.0.3
[25] askpass_1.1            openssl_1.4.1          tidyselect_0.2.5
[28] tibble_2.1.3           GenomeInfoDbData_1.2.2 XML_3.98-1.20
[31] crayon_1.3.4           dplyr_0.8.3            dbplyr_1.4.2
[34] bitops_1.0-6           rappdirs_0.3.1         grid_3.6.1
[37] RBGL_1.62.0            DBI_1.0.0              magrittr_1.5
[40] graph_1.64.0           stringi_1.4.3          vctrs_0.2.0
[43] tools_3.6.1            bit64_0.9-7            glue_1.3.1
[46] purrr_0.3.3            hms_0.5.1              BiocManager_1.30.9
[49] memoise_1.1.0
```