Extended examples of the functionalities of the *Chimera* package

Introduction

This document provides a description of the main functionalities of *Chimera* package applied to the analysis of the output of *ChimeraScan*, *FusionMap* and *deFuse* on the dataset published by Edgren and Kangaspeska (Edgren, et al., 2011; Kangaspeska, et al., 2012). These papers provide a set of PCR validated fusion (TPs) events (Table 1). The output of *ChimeraScan*, *FusionMap* and *deFuse* have been generated using the default settings suggested by the tools' developers. These data are available in the example folder of chimera version 1.6.6 (or higher). The same folder contains the list of Edgren validated fusions (file *edgren.stat.detection.txt*) detectable with the tools supported by Chimera.

Table 1: Fusions, validated by PCR in Edgren and Kangaspeska's papers

Sample	donorEnd	acceptorStart	Fusions
BT-474	ACACA	STAC2	ACACA:STAC2[§#cs]
BT-474p	AHCTF1	NAAA	AHCTF1:NAAA[§#]
SK-BR-3	ANKHD1	PCDH1	ANKHD1:PCDH1
MCF-7	ARFGEF2	SULF2	ARFGEF2:SULF2[§cs]
MCF-7	BCAS4	BCAS3	BCAS4:BCAS3[§cs]
KPL-4	BSG	NFIX	BSG:NFIX[§cs]
SK-BR-3	CCDC85C	SETD3	CCDC85C:SETD3[§cs]
BT-474	CPNE1	PI3	CPNE1:PI3
SK-BR-3	CSE1L	KCNB1	CSE1L:KCNB1[§]
SK-BR-3	CYTH1	EIF3H	CYTH1:EIF3H[§#cs]
SK-BR-3	DHX35	ITCH	DHX35:ITCH[§cs]
BT-474	DIDO1	TTI1	DIDO1:TTI1[§cs]
BT-474	GLB1	CMTM7	GLB1:CMTM7[§cs]
MCF-7*	GCN1L1	MSI1	GCN1L1:MSI1[§cs]
MCF-7*	GDPD1	TMEM49/VMP1	GDPD1:VMP1
BT-474	LAMP1	MCF2L	LAMP1:MCF2L[§cs]
BT-474p	MED1	STXBP4	MED1:STXBP4[§cs]
BT-474p	MED1	ACSF2	MED1:ACSF2[§cs]
BT-474p	MED13	BCAS3	MED13:BCAS3[§#cs]
SK-BR-3	NFS1	PREX1	NFS1:PREX1
KPL-4	NOTCH1	NUP214	NOTCH1:NUP214[§#cs]
KPL-4	PPP1R12A	"SEPT10"	PPP1R12A:SEPT10[§cs]
BT-474p	PIP4K2B	RAD51C	PIP4K2B:RAD51C[§cs]
BT-474	RAB22A	MYO9B	RAB22A:MYO9B[§cs]
SK-BR-3	RARA	PKIA	RARA:PKIA[§]
BT-474	RPS6KB1	SNF8	RPS6KB1:SNF8[§cs]
MCF-7	RPS6KB1	VMP1	RPS6KB1: VMP1
MCF-7*	SMARCA4	CARM1	SMARCA4:CARM1[§#cs]
BT-474	SKA2	MYO19	SKA2:MYO19[§]
BT-474	STARD3	STARD3	STARD3:STARD3
BT-474p	STX16	RAE1	STX16:RAE1[§]
SK-BR-3	SUMF1	LRRFIP2	SUMF1:LRRFIP2[§cs]

SK-BR-3	TATDN1	GSDMB	TATDN1:GSDMB[§cs]
BT-474p	THRA	AC090627/SKAP1	THRA:SKAP1
BT-474p	TOB1	SYNRG	TOB1:SYNRG[§cs]
BT-474p	TRPC4AP	MRPL45	TRPC4AP:MRPL45[§]
BT-474	VAPB	IKZF3	VAPB:IKZF3[§cs]
BT-474p	USP32	MED1	USP32:MED1
SK-BR-3	WDR67	ZNF704	WDR67:ZNF704
BT-474	ZMYND8	CEP250	ZMYND8:CEP250[§cs]

***Fusions retrieved from (Kangaspeska, et al., 2012); [§]fusions detected combining chimeraScan and deFuse outputs. [cs]Fusions detected by chimeraScan. [#]In frame fused peptides.**

Fusion detection algorithms characteristics

According with the classification of fusion-finder algorithms proposed by Beccuti and co-workers (Beccuti M, 2013), the alignment strategies of *deFuse*, *ChimeraScan* and *FusionMap* can be classified as: whole paired-end (Fig. 1A), paired-end + fragmentation (Fig. 1B) and direct fragmentation (Fig. 1C).

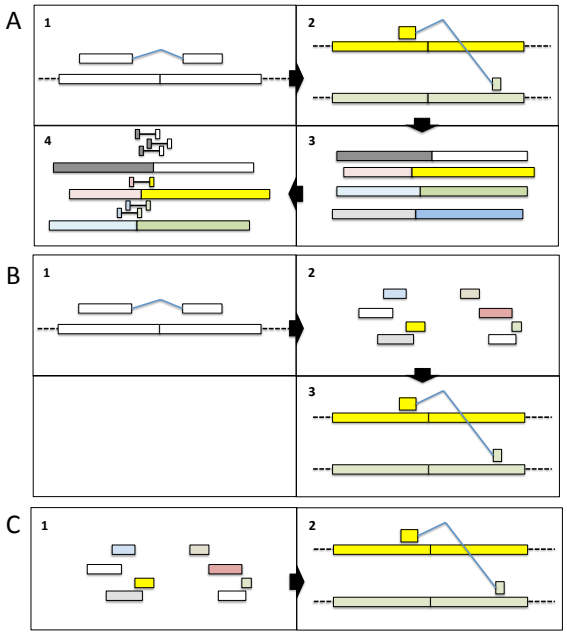

Fig. 1: A) Whole paired-end approach. 1. Mapping reads on reference genome. 2. Search for discordant alignments. 3. Reconstruct putative chimeric products. 4. Align unmapped reads on the chimeric products. B) Paired-end + fragmentation approach. 1. As A1. 2. Not aligned reads are fragmented. 3. Reads fragments are aligned over genomes and used to search for discordant alignments. C) Direct fragmentation. 1. As B2. 2. As B3.

The three tools also implement different sets of filters for fusion detection refinements. *deFuse* uses only paired-end information. *ChimeraScan* uses paired-end and anchor-length information. *FusionMap* uses black-lists, junction-spanning reads, quality score, read-through transcripts, and PCR artifacts detection. Furthermore, in (Carrara, et al., 2013) *ChimeraScan* resulted to be the most sensitive tool and *deFuse*, together with *TopHat-fusion*, ranked just behind chimeraScan. Taken together the above information

suggests that the integration of the results produced by these three tools might result more informative then the use of a single tool.

Data upload in Chimera.

importFusionData: Function *importFusionData* creates a list of fSet objects (see appendix 1), containing the output of one of the following fusion detection tools:

- bellerophontes
- deFuse
- FusionFinder
- FusionHunter
- mapSplice,
- tophat-fusion
- FusionMap
- chimeraScan
- STAR
- Rsubread
- fusionCatcher

As stated in the introduction, we analyze the output of only three of them, since the objective of the analysis is simply to provide an overview of *Chimera* functionalities to highlight the efficacy of the instruments available in *Chimera*.

```
#importing data
> library(chimera)
> df.e <- importFusionData("defuse", paste(find.package(package="chimera"),
"/examples/Edgren_df.tsv", sep=""))
> cs.e <- importFusionData("chimerascan", paste(find.package(package="chimera"),
"/examples/Edgren_cs.txt", sep=""), org="hs")
> fm.e <- importFusionData("fusionmap", paste(find.package(package="chimera"),
"/examples/Edgreen_fm.txt", sep=""), org="hs")
```

On the given dataset, *deFuse* detects 915 fusions, *ChimeraScan* 13346 and *FusionMap* 69. Fusion names can be extracted with the function *fusionName*:

```
#extracting fusion names
> fm.n.e <- fusionName(fm.e)
> cs.n.e <- fusionName(cs.e)
> df.n.e <- fusionName(df.e)
```

The fusions detected by the different algorithms show a very limited overlap (Figure 2A), as does the set of PCR validated fusions (TPs) (Figure 2B).
FusionMap is the tool that implements the largest number of filters but it is also the one that detects the lowest number of TPs, which are totally included in the set of fusions found by *deFuse* and *ChimeraScan* (Fig. 2B). *ChimeraScan* detects 25 PCR validated fusions and 8 of them are *ChimeraScan* specific, whereas *deFuse* detects a total of 23 PCR validated fusion and 6 of them are *deFuse* specific (Fig. 2B).

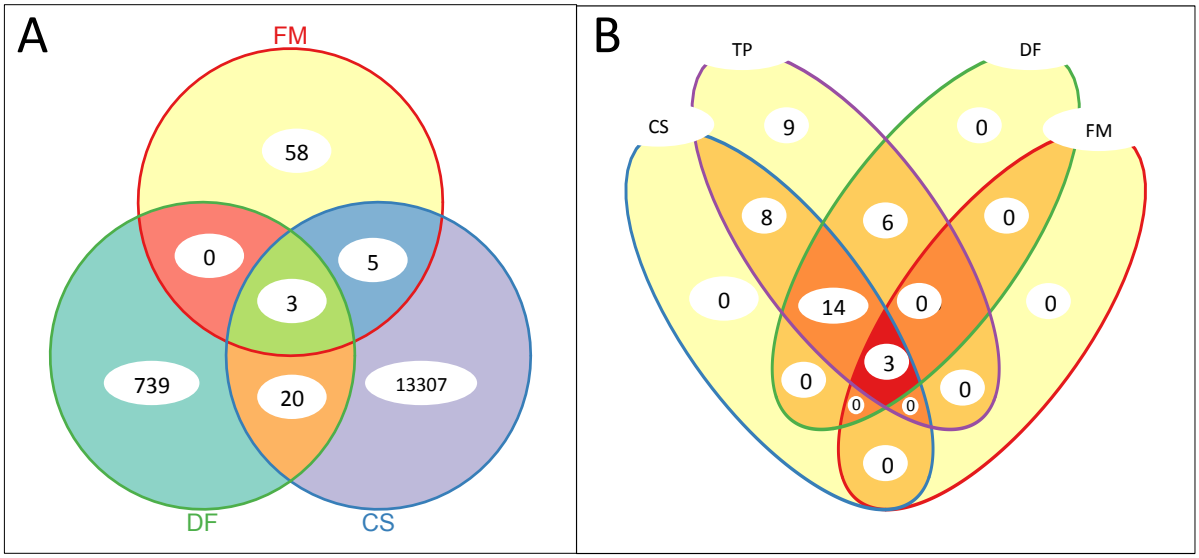

Fig. 2: Venn diagram representing the overlaps between fusions detected by deFuse (DF), ChimeraScan (CS) and FusionMap (FM). A) All set of detected fusions, B) PCR validated fusions (Table 1). TP indicate the set of fusions confirmed by PCR in the Edgren and Kangaspeska papers.

supportingReads: Function *supportingReads* allows the extraction of the reads associated to the detected fusions. Specifically, the function provides two options: "all" and "spanning". The first retrieves all the reads supporting the fusion event and the latter the reads mapping over the break-point (in case the fusion detection tool provides them).

```
#extracting the number of spanning reads supporting the fusions
> supporting.reads.fm <- supportingReads(fm.e, fusion.reads="spanning")
> supporting.reads.cs <- supportingReads(cs.e, fusion.reads="spanning")
> supporting.reads.df <- supportingReads(df.e, fusion.reads="spanning")
> names(supporting.reads.fm) <- fm.n.e
> names(supporting.reads.cs) <- cs.n.e
> names(supporting.reads.df) <- df.n.e
```

It is notable that *deFuse* detects a higher number of break-point mapping reads with respect to *ChimeraScan* and *FusionMap* (Fig. 3).

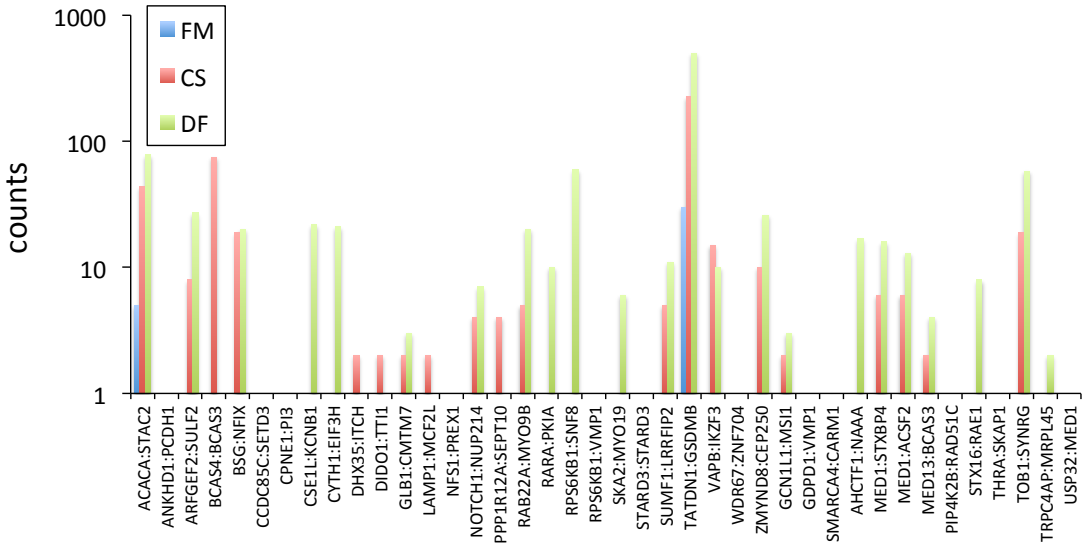

Fig. 3: Spanning reads supporting the PCR validated fusions.

The three fusion detection tools are based on different combinations of alignment approaches and filtering and it is clear, from the above results, that each tool is able to detect only a subset of TPs. The intersection of the fusions detected by *ChimeraScan, deFuse* and *FusionMap* contains only 3 fusions and all of them belong to the set of the TPs, but, since the number of TPs is 40, intersection of the tool outputs substantially reduces the detection sensitivity. The union of the fusions detected by the three tools amounts to 14261 putative fusions: despite this large number, the union contains only 31 of the 40 TPs present in the dataset. Therefore, reducing the number of non-informative fusions is mandatory, and for this reason *Chimera* provides a set of filtering and annotating procedures that facilitate fusions prioritization.

Filtering procedures

filterList: Function *filterList* allows to: i) remove, from the imported fusions list, fusions supported by a number of reads lower than a user defined threshold, e.g. at least 1 read spanning over the break-point, ii) remove fusions including introns, iii) remove fusions involving genomic regions that are not annotated known genes, iv) remove read-through events, v) filter fusions on the basis of Oncofuse (Shugay, et al., 2013) annotation (see next paragraph).

Spanning reads filter (Fig. 4) has a notable effect even with a low spanning reads threshold, i.e. a threshold of 1 spanning read reduces the total detected fusions from 14261 to 1596 (Fig. 4A) and it has minimal effects on the number of TPs, since only PIP4K2B:RAD51C and CCDC85C:SETD3 are lost (Fig. 4B). This dramatic effect is linked to the default setting of *ChimeraScan*, which does not include spanning reads threshold to the detected putative fusions.

```
#Combining chimeraScan and deFuse results
>csdf.e <- c(cs.e, df.e)
>tmp2 <- filterList(csdf.e, type="supporting.reads", query=1)
#This step is very time consuming and it should be run as batch
>tmp3 <- filterList(csdf.e, type="intronic")
>tmp4 <- filterList(csdf.e, type="annotated.genes", parallel=TRUE)
>tmp5 <- filterList(csdf.e, type="read.through")
```

Note that only the output of chimeraScan and deFuse have been considered for Fig.4, since FusionMap detects a set of TPs which is included in the set detected by the other two tools.

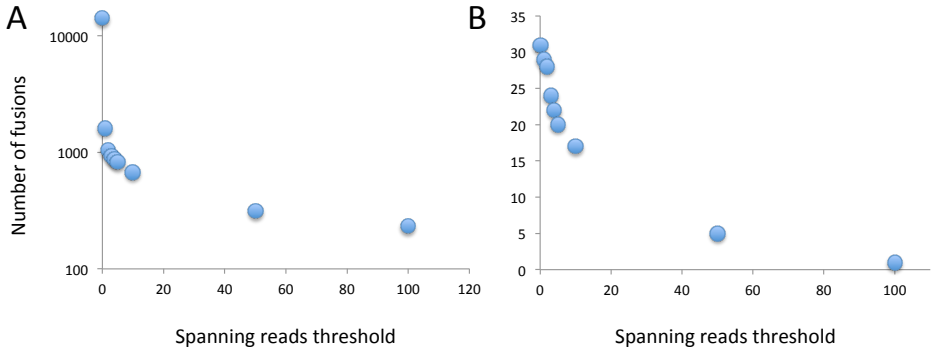

Fig. 4: Reduction of detected fusions using the filter based on the number of supporting spanning reads. A) Detected fusions. B) PCR validated fusions.

The other available filters have a much smaller effect (Table 2).

Table 2: effects of filters on total number of detected fusions and TPs

Filter	Total number of detected fusions (*ChimeraScan+deFuse*)	TPs
Unfiltered	14261	31
Intronic	10368	31
Annotated	10964	31
Read-through	14237	31

The greatest reduction, corresponding to 3893 discarded fusion events, is obtained by removing fusions in which a full intron, or part of it, is included in the mature transcript. The rationale of this filter is that, since a fusion has to produce a translatable protein, the presence of a long intron will produce, in the best scenario, a truncated protein. Therefore, this filter is useful if the target of the analysis are fusions in which the two partners produce a chimeric protein, but it can be dangerous and should not be applied if the interest is on truncated proteins. The other filter with a comparable reduction factor is the one based on annotated genes, which leads to 10964 retained fusions (3297 discarded). This filter keeps only fusions in which both genomic regions correspond to the location of annotated genes. The applicability conditions are the same as for the intronic filter. The read-through filter has a very limited effect. It is however notable that all above mentioned filters, in the biological framework of the Edgren/Kangaspeska studies, never discard a TP.
We have also quantified the effect of filtering procedures using precision (1) and recall (2) on the set of 60 validated fusions and 61 false fusions described in the deFuse paper (McPherson, et al., 2011).

$$precision = \frac{TP}{TP+FP}\ (1); recall = \frac{TP}{TP+FN}\ (2)$$

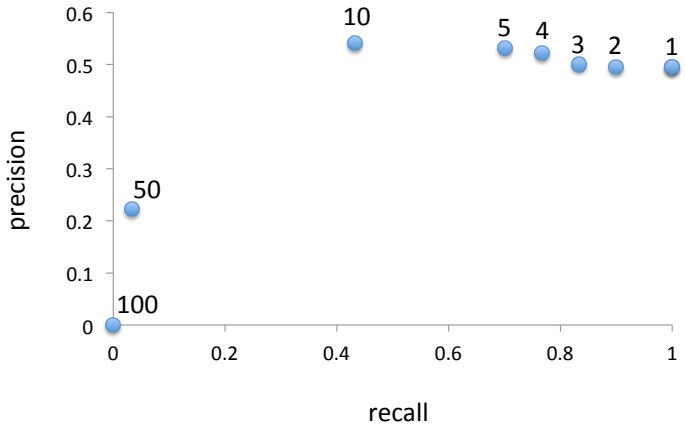

Fig. 5: Precision versus recall as a consequence of spanning reads filtering on the set of 60 validated fusions and 61 false fusions described in deFuse paper. The number placed over each dot refers to the used spanning read threshold.

The filter based on spanning reads threshold has a limited precision, i.e. ~0.5 up to a threshold of 10 spanning reads (Fig.5). However, the increase of the threshold value negatively affects the recall rate (Fig. 5). On the other end the "Intronic" filter performs much better showing a precision of 0.8 and a recall of 0.87. The efficacy of fusion filters "Annotated" and "Read-through", implemented in Chimera, could not be evaluated since both validated and false fusions, described in deFuse paper, are all associated to genomic regions encompassing known genes and both genes, involved in fusion, have different names.

Annotation procedures

oncofuseRun: Chimera embeds Oncofuse, a recently published naive Bayes Network Classifier for fusions events (Shugay, et al., 2013). Oncofuse can be downloaded using the function *oncofuseInstallation* and it can be run using the function *oncofuseRun,* which returns Oncofuse results organized in a data frame structure. When the parameter "plot" is set to TRUE, *oncofuseRun* provides also a plot (Fig. 6) of the expression gain scores, as functions of the Bayes probability that the fusion behaves as a passenger event for the tumor, i.e. a low p-value corresponds to a high probability for the fusion to be a driver tumor mutation (for more information see (Shugay, et al., 2013)).

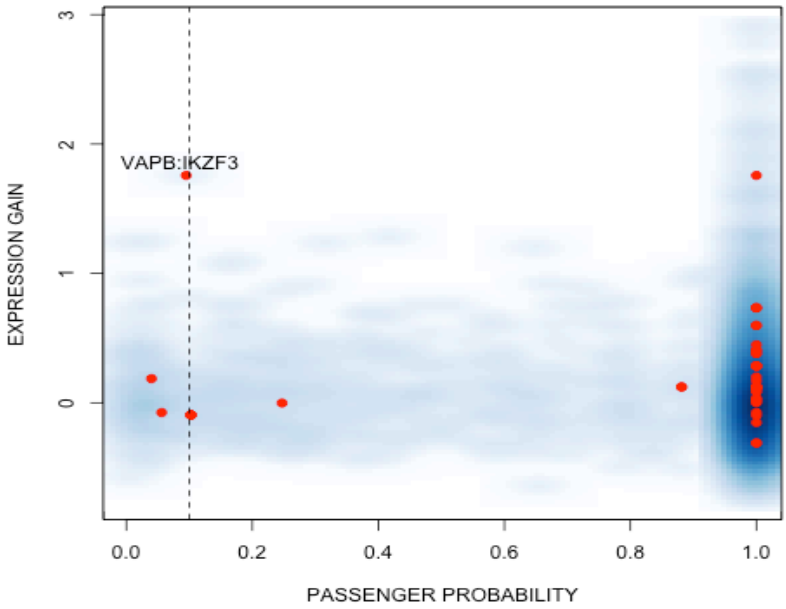

Fig. 6: Plot of the fusion expression gain as function of the probability of the fusion to behave as tumor driver event. The smoothed density representation of the scatterplot is shown in blue. The PCR validated fusions are shown as red dots. The dashed vertical line refers to p-value=0.1. So, for example, VAPB:IKZF3 is the fusion characterized by a probability to be a passenger mutation below 0.1 and an expression gain score greater than 1.

Furthermore, the output of Oncofuse is used by the function *listFilter* to filter fusions on the basis of Oncofuse annotation. The filtering procedure selects fusions located on exons or coding regions (CDS) and those being putative tumor driver mutations, i.e. characterized by low passenger probability, or being characterized by a specific expression gain score threshold.

```
#Installing Oncofuse
>installOncofuse()
#executing oncofuse using epithelial tissue model, since the fusions are related to breast cancer
>csdf.of <- oncofuseRun(csdf.e, tissue="EPI", plot=TRUE)
#extracting only fusions located in the CDS of both genes
>tmp6 <- filterList(csdf.e, oncofuse.output=csdf.of, type="oncofuse",
oncofuse.type="g5g3CDS", parallel=TRUE)
#extracting the fusions characterized by a probability to be a passenger mutation ≤ 0.1
>tmp7 <- filterList(csdf.e, oncofuse.output=csdf.of, type="oncofuse",
oncofuse.type="passenger.prob", query=0.1, parallel=TRUE)
```

chimeraSeqSet: It is also possible to reconstruct the sequence of the fused transcript using the function *chimeraSeqSet*. Fusion reconstruction is particularly useful to remap the reads over the putative transcripts of interest, specifically in cases in which this option is not provided by the fusion finder used for the mapping, as in the case of

```
#PCR validated fusions
>tp <- read.table(paste(find.package(package="chimera"),"examples",
"Edgren_true.positives.txt" , sep="/"),sep="\t",header=T)
#extracting fusion names for chimeraScan+deFuse results
csdf.e.n <- fusionName(csdf.e, parallel=T)
#detected fusions
>length(csdf.e.n[which(csdf.e.n%in%as.character(tp$fusions))])
[1] 51
>length(unique(csdf.e.n[which(csdf.e.n%in%as.character(tp$fusions))]))
[1] 29
>tmp.seq <- chimeraSeqSet(csdf.e[which(csdf.e.n%in%as.character(tp$fusions))],
parallel=FALSE)
#saving the transcripts fusions as fasta file.
>sapply(tmp.seq, function(x){writeXStringSet(x, "detected.fusions.fa", format="fasta",
append=TRUE)})
```

ChimeraScan.
The 29 TPs present in the union of chimeraScan and deFuse results are represented by 51 fusion events, since different break-points for the same fusion transcript have been detected. The sequence data generated by *chimeraSeqSet* can be imported as extended description of the fusion of interest with the function *addRNA*, see Appendix 1. The name of the fasta sequence encompassing the fusion transcript, generated by *chimeraSeqSet,* has the following structure: transcript1 name - breakpoint on transcript1: transcript2 name - breakpoint on transcript2, e.g. uc002loy.4-522:uc002mwd.3-5482.

fusionPeptides: To know more about the characteristics of a break-point, the reconstructed fusions can be used to extract the donor and acceptor peptides involved in the fusion. This operation can be done using function *fusionPeptides*, which provides a brief output describing the type of observed events (see green inset below).

```
#Extracting the peptide sequences involved in the fusion
>tmpx <- lapply(tmp.seq, fusionPeptides)
……..
fused proteins are not in frame
fused proteins are in frame
……..
>tmpx[c(1:3,50:51)]
```

The output produced by *fusionPeptides* is a list of objects encompassing i) the fusion event, if present, ii) the 5' end peptide, iii) the 3' end peptide, iv) the full 5' end protein and v) the full 3' end protein (Fig. 7).

```
$`uc002xvp.1-243:uc002iyu.4-1031`
  A AAStringSet instance of length 5
    width seq                                                                                                  names
[1]     0                                                                                                      fusion
[2]    60 MQRTGGGAPRPGRNHGLPGSLRQPDPVALLMLLVDADQPEPMRSGARELALFLTPEPGAE                                         p1pep
[3]   106 GTFDRSVTLLEVCGSWPEGFGLRHMSSMEHTEEGLRERLADAMAESPSRDVVGSGTELQREGSIETLSNSSGSTSGSIPRNFDGYRSPLPTNESQPLSLFPTGFP*  p2pep
[4]   120 MQRTGGGAPRPGRNHGLPGSLRQPDPVALLMLLVDADQPEPMRSGARELALFL...EGMLLRLEEFCSLADLIRSDTSQILEENIPVLKAKLTEMRGIYAKVDRLES* p1
[5]   914 MNEAMATDSPRRPSRCTGGVVVRPQAVTEQSYMESVVTFLQDVVPQAYSGTPL...GTELQREGSIETLSNSSGSTSGSIPRNFDGYRSPLPTNESQPLSLFPTGFP* p2

$`uc010cuy.3-107:uc002hrs.3-1862`
  A AAStringSet instance of length 5
    width seq                                                                                                  names
[1]    44 MFREFTQQNICVGVGRSKDADGFIRVSSGKKRGLVPVDALTEI*                                                         fusion
[2]     9 MFREFTQQN                                                                                            p1pep
[3]    35 ICVGVGRSKDADGFIRVSSGKKRGLVPVDALTEI*                                                                  p2pep
[4]   992 MFREFTQQNKATLVDHGIRRLTFLVAQKDFRKQVNYEVDRRFHREFPKFFTFR...SRDYVLKQIRSLVQANPEVAMDSIIHMTQHISPTQRAEVIRILSTMDSPST* p1
[5]   412 MTEMSEKENEPDDAATHSPPGTVSALQETKLQRFKRSLSLKTILRSKSLENFF...PFSGNKEQGYMSLKENQICVGVGRSKDADGFIRVSSGKKRGLVPVDALTEI* p2
```

Fig. 7: Output of *fusionPeptides* functions. uc002xvp.1-243:uc002iyu.4-1031 fusion involves coding regions for both genes but the fusion is not in frame. uc010cuy.3-107:uc002hrs.3-1862 fusion involves coding regions that produce a small in-frame fusion protein. uc010wdb.2-1:uc002hrs.3-2596

The output of *fusionPeptides* reveals a total of 6 in-frame fusions out of the 29 TPs (Table 1, fusions marked with #). It has to be noted that the 29 TPs used in this analysis have been validated in breast cancer cell lines by PCR (Edgren, et al., 2011; Kangaspeska, et al., 2012) but, to the best of our knowledge, there is no experimental evidence that all 29 fusion events are able to generate fusion proteins.

subreadRun: The reconstructed fusions generated by *chimeraSeqSet* function are also of potential interest as reference for remapping. This option could be of interest in case the fusion detection tool does not use, in the alignment procedure, reconstructed fusions, as in the case for ChimeraScan.
Chimera uses Rsubread (Liao, et al., 2013) for alignment and the wrapper function *subreadRun* generates a sorted and indexed bam file containing only the mapped reads.

```
>download.file("http://130.192.119.59/public/edgren_1.fastq.gz",
"edgren_1.fastq.gz", mode="wb")
>download.file("http://130.192.119.59/public/edgren_2.fastq.gz",
"edgren_2.fastq.gz", mode="wb")
>system("gzip –d *.gz")
>require(Rsubread)
#This step is significantly time consuming and it should be run as batch
> subreadRun(ebwt=paste(find.package(package="chimera"),
"/examples/uc002xtx.4-272_uc010zyd.2-988.fa", sep=""), input1= "edgren_1.fastq",
input2="edgren_2.fastq", outfile.prefix="accepted_hits", alignment="se", cores=48)
>dir()
[1] "accepted_hits_mapped.bam" "accepted_hits_mapped.bam.bai"
```

Mapped reads can be imported as part of the description of the fusion event, using the method *addGA* (see appendix 1).

Validation tool

Only the tools based on "whole paired-end alignment approach" generate putative fusions and use them to detect reads mapping in the break-point region. However, to the best of our knowledge (Beccuti M, 2013), there is no tool that assesses if the reads mapping on a putative fusion transcripts are able to reconstruct, by *de novo* assembly, the break-point region.

gapfillerRun: This function integrates in the package the GapFiller tool (Nadalin, et al., 2012), previously developed by two of the co-authors of *Chimera*. GapFiller is a seed-and-extend local assembler able to correctly fill the gap between paired reads, thus it generates accurate longer sequences with respect to input reads.

The rationale of the reconstruction approach of *gapfillerRun* is summarized in Fig. 8.

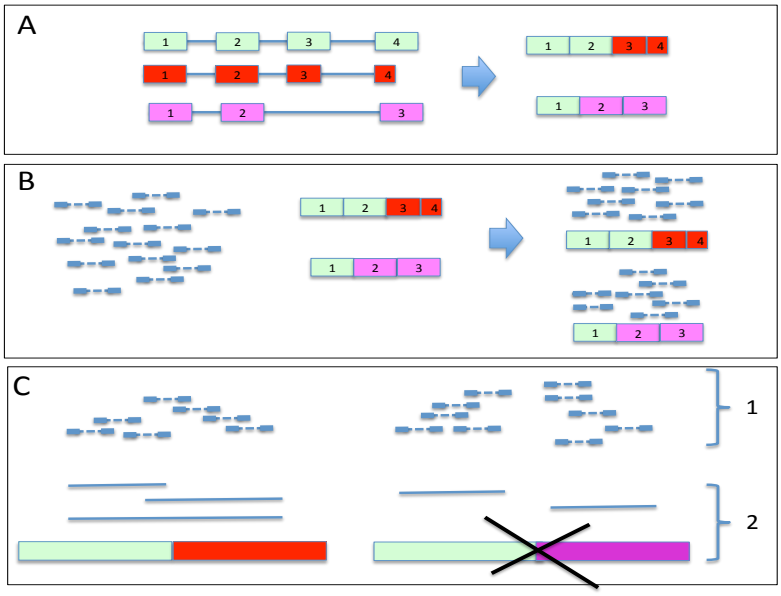

Fig. 8: *De novo* reconstruction of break-point in chimera. A) *chimeraSeqSet* is used to reconstruct putative fusions. B) subreadRun is used to remap the reads over the putative fusion transcripts. C) *De novo* reconstruction. 1. GapFiller is used to assemble the sequence between the two mates of the paired reads. 2. GapFiller contigs are aligned to the fusion transcript, and if at least one of them overlaps the break-point, the fusion is considered validated by *de novo* assembly.

It is important to remark that the alignment of the ungapped sequence lying between the two mates of a paired read (i.e. a GapFiller contig) provides a stronger evidence for the mapping to be correct, with respect to the alignment of paired reads alone. Indeed, having the contig instead of the paired read, means having the exact distance between the mates as well as the complete sequence, which can be mapped with less ambiguity against the fusion transcript.
To test the efficacy of the above-mentioned approach we have used fusions detected by ChimeraScan in (Edgren, et al., 2011) (Table 3).

Table 3: *De novo* reconstruction of the break-point region

Fusion	# reads at break-point in fusion transcript	GapFiller break-point reconstruction
RPS6KB1:SNF8	396	YES
BCAS4:BCAS3	216	YES
MED1:ACSF2	167	YES
LAMP1:MCF2L	122	YES
CCDC85C:SETD3	80	YES
PIP4K2B:RAD51C	80	YES
BSG:NFIX	73	YES
SMARCA4:CARM1	72	YES
DIDO1:TTI1	64	YES
MED1:STXBP4	61	YES
NOTCH1:NUP214	58	YES
TOB1:SYNRG	43	YES
VAPB:IKZF	42	YES
RAB22A:MYO9B	42	YES
GLB1:CMTM7	41	YES
MED13:BCAS3	35	NO
GCN1L1:MSI1	29	NO
DHX35:ITCH	7	NO
ARFGEF2:SULF2	4	YES
CYTH1:EIF3H	1	NO
SUMF1:LRRFIP2	1	NO
TATDN1:GSDMB	0	NO
ACACA:STAC2	0	NO
ZMYND8:CEP250	0	NO
PPP1R12A:SEPT10	0	NO

The remapping of Edgren data on the fusion transcripts shows that the number of reads spanning over the break-point ranges between 396 and 0. We have investigated the 9 cases in which we do not validate the fusions by de novo assembly (Table 3). In the case of the absence of reads covering the break point (TATDN1:GSDMB, ACACA:STAC2, ZMYND8:CEP250, PPP1R12A:SEPT10) *de novo* assembly fails because of the lack of overlapping reads that allows the joining of the contigs of the two transcripts. A similar situation also happens in the case of CYTH1:EIF3H, SUMF1:LRRFIP2, MED13:BCAS3 (Fig. 9A), where all spanning reads comes from one of the two transcripts over the break point. The possibility that fusion has a high probability of being an artifact, when the majority of the spanning reads come from only one of the two fused transcripts, was also highlighted in the Edgren paper (Edgren, et al., 2011). We have other two cases in which

de novo assembly of break points fails: GCN1L1:MSI1 (Fig. 9B) and DHX35:ITCH (Fig. 9C). In both cases the reads spanning the break point comes from both transcripts but their number is a very little fraction of the total reads mapping on the two transcripts. We are now working on a new version of GapFiller to overcome this issue.

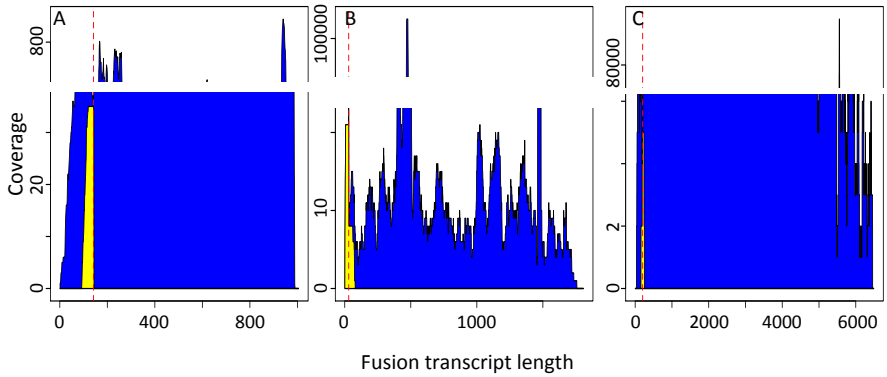

Fig. 9: Coverage of fusions not validated by *de novo* assembly. A) MED13:BCAS3, B) GCN1L1:MSI1, C) DHX35:ITCH. In blue full fusion transcript coverage. In yellow break point coverage. Break point is shown by a dashed read line.

An example of break point validation is provided in the green inset below.

```
#extracting ARFGEF2:SULF2 from ChimeraScan output
> my.fset <- cs.e[[which(cs.n.e=="ARFGEF2:SULF2")]]
#remapping reads on ARFGEF2:SULF2
> subreadRun(ebwt=paste(find.package(package="chimera"), "/examples/uc002xtx.4-
272_uc010zyd.2-988.fa", sep=""), input1=paste(find.package(package="chimera"),
"/examples/uc002xtx.4-272_uc010zyd.2-988_R1.fastq", sep=""),
input2=paste(find.package(package="chimera"), "/examples/uc002xtx.4-
272_uc010zyd.2-988_R2.fastq", sep=""), outfile.prefix="accepted_hits", alignment="se",
cores=4)
my.fset <- addGA(my.fset,"accepted_hits.bam")
my.fset <- addGA(my.fset,"accepted_hits.bam")
#evaluating the number of reads spanning on the break-point at nt 268
> tmp.ga <- fusionGA(my.fset)
> tmp.acc <- which(start(tmp.ga) < 268)
> tmp.don <- which(end(tmp.ga) > 268)
> length(tmp.ga[intersect(tmp.acc, tmp.don)])
[1] 4
# Installing GapFiller
> gapfillerInstallation(os="mac64")
#running gapfiller
> tmp <- gapfillerRun(fusion.fa= paste(find.package(package="chimera"),
"/examples/uc002xtx.4-272_uc010zyd.2-988.fa", sep=""),seed1=
paste(find.package(package="chimera"), "/examples/uc002xtx.4-272_uc010zyd.2-
988_R1.fastq", sep=""), seed2= paste(find.package(package="chimera"),
"/examples/uc002xtx.4-272_uc010zyd.2-988_R2.fastq", sep=""), slack=7, block.length=5,
overlap=20)
.....
de novo alignment has overlap over the fusion break point
>tmp
#The output of gapfillerRun is a list encompassing:
#i) the contigs generated by GapFiller,
#ii) the conting encompassing the break-point
#iii) the sequence of the fusion transcripts.
```

Visualization tool

prettyPrint: the function *prettyPrint* reorganizes a list of fSet objects, in a format suitable to be saved in a a tab delimited file (Fig. 10).

```
#Extracting the peptide sequences involved in the fusion
> tmp <- importFusionData("fusionmap",
paste(find.package(package="chimera"),"/examples/mcf7.FMFusionReport", sep=""),
org="hs")
> prettyPrint(tmp, "tmp.df.txt", fusion.reads="spanning")
```

gene1	chr.gene1	breakpoint.gene1	strand.gene1	transcripts.gene1	gene2	chr.gene2	breakpoint.gene2	strand.gene2	transcripts.gene2	fusion.breakpoint	supporting.reads
HMGN2	chr1	26799003	-	uc001bmp.4	ESYT1	chr12	56527458	-	uc001sjr.3,uc001sjq.3	CCTCTTCTTCACActgctcca	1
CC2D1B	chr1	52818776	+	uc001ctq.2,uc001ctr.3,uc001cts.3	DTYMK	chr2	242615169	+	10zpb.2,uc002wbz.2,uc002wc	ACATGCTTAACAGgtcgttt	1
NOS1AP	chr1	162336994	+	10pks.1,uc001gbw.2,uc001gbv.2,uc009w	C1orf226	chr1	162351681	+	uc010pkt.1,uc001gby.2	GCAGCCCCTTAGttgacca	1
GREB1	chr2	11682953	+	uc002rbm.3	GREB1	chr2	11696580	+	uc002rbk.1,uc002rbl.3	AAAGTGAGTTCAGtagctgc	1
RYBP	chr3	72495772	+	uc003dpe.3	YAF2	chr12	42631957	+	10skp.2,uc001rmv.3,uc001rm	TTGTCTTTTTGGCctggtgg	1
SLC30A5	chr5	68400246	+	uc011crc.2,uc003jvg.3	AZIN1	chr8	103842164	-	uc003yky.3,uc003ykx.3	CTAGGATTACAGtagaaaaa	1
TUBB	chr6	30691455	-	uc003nrl.3,uc011dmq.2	KRT80	chr12	52574593	+	uc001rzw.3	TATCATAGAGGGCtttgattg	1
EEF1A1	chr6	74228908	+	uc003phi.3,uc021zbs.1,uc003phj.3	GHITM	chr10	85912360	-	10qma.1,uc001kcs.1,uc010qm	ATACCAGCTTCAAattcacca	1
HNRNPK	chr9	86584135	-	04anh.4,uc011lsx.2,uc004anf.4,uc004an	AATF	chr17	35311150	+	uc002hni.3	CTGTTCACCAGATgttttccc	1
NDUFA1	chrX	119007356	+	uc004esc.4	SYNJ2BP-COX16	chr14	70834275	-	uc001xmc.4	CTATGTGTCAAAGacggagt	1
FAM208B	chr10	5727138	+	uc001iij.3	FAM208B	chr10	5751493	+	uc021pmm.1	ACGCCAAGGACGGcatttat	1
YLPM1	chr14	75294295	+	uc001xqo.1	ITPK1	chr14	93406390	-	01ybf.3,uc001ybh.3,uc001yb	CTGTATTGCTGCccggcttt	1
SULF2	chr20	46365686	+	002xto.3,uc002xtr.3,uc002xtq.3,uc010gh	ARFGEF2	chr20	47538547	-	uc002xtx.4	CCTGCATGGAACcgagcgc	2

Fig. 10: Example of the tabular output generated by *prettyPrint* function

breakpointOverlaps: this function uses the information stored in the fusionRNA and fusionGA slots of an fSet object to produce a plot of the coverage of the full fusion transcript (Fig. 11, blue) and of the region spanning over the break point (Fig. 11, yellow). A GAlignment object encompassing all the reads spanning over the break point is also produced.

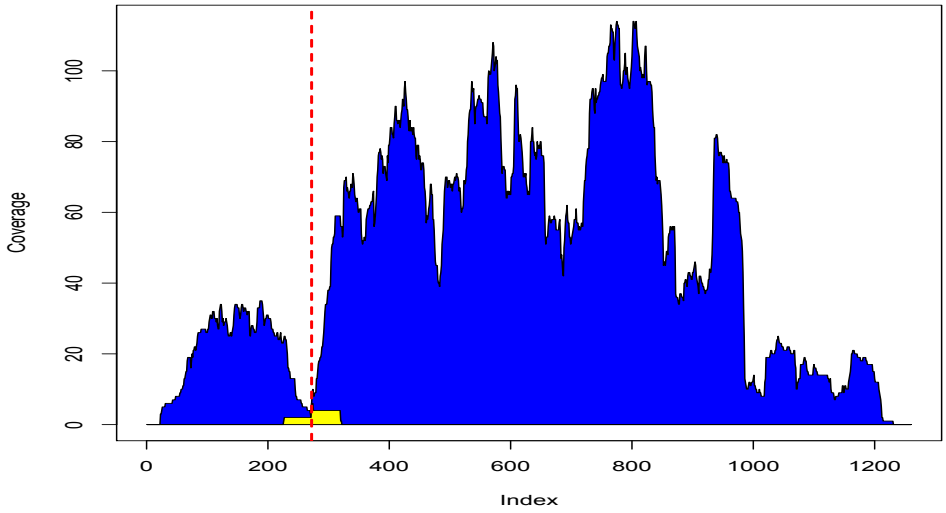

Fig. 11: Coverage plot of the full fusion transcript (blue) and of the reads spanning over the break point (yellow). The break point is marked by a dashed line (red)

```
#Inspecting break point spanning reads
> load(paste(find.package(package="chimera"), "/examples/fset_ARFGEF2-SULF2.rda",
sep=""))
> my.seq <- chimeraSeqs(my.fset)
> my.fset <- addRNA(my.fset, my.seq)
> tmp <- breakpointOverlaps(my.fset,plot=TRUE)
```

References

Beccuti M, C.M., Cordero F, Donatelli S, Calogero RA. The structure of state-of-art gene fusion-finder algorithms. *OA Bioinformatics* 2013;1(1):2.

Carrara, M.*, et al.* State-of-the-Art Fusion-Finder Algorithms Sensitivity and Specificity. *BioMed research international* 2013;2013:340620.

Edgren, H.*, et al.* Identification of fusion genes in breast cancer by paired-end RNA-sequencing. *Genome biology* 2011;12(1):R6.

Kangaspeska, S.*, et al.* Reanalysis of RNA-sequencing data reveals several additional fusion genes with multiple isoforms. *PloS one* 2012;7(10):e48745.

Liao, Y., Smyth, G.K. and Shi, W. The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. *Nucleic acids research* 2013;41(10):e108.

McPherson, A.*, et al.* deFuse: an algorithm for gene fusion discovery in tumor RNA-Seq data. *PLoS computational biology* 2011;7(5):e1001138.

Nadalin, F., Vezzi, F. and Policriti, A. GapFiller: a de novo assembly approach to fill the gap within paired reads. *BMC bioinformatics* 2012;13 Suppl 14:S8.

Shugay, M.*, et al.* Oncofuse: a computational framework for the prediction of the oncogenic potential of gene fusions. *Bioinformatics* 2013;29(20):2539-2546.

Appendix 1:

Data from each fusion are stored in a data structure called fSet encompassing the following information:

slots:

@ *fusionInfo*: a list encompassing the following fusion informations derived from the output of the fusion detection tool:

$ fusionTool: the tool that has generated the fusions

$ UniqueCuttingPositionCount: the number of unique cutting positions detected for the fusion.

$ SeedCount:

- the number of reads overlapping the break-point, i.e. spanning reads (FusionMap, FusionHunter, mapSplice, Tophat-fusion, ChimeraScan, STAR, Rsubread, FusionCatcher).
- Both spanning and encompassing reads (Bellerophontes, FusionFinder).
- Encompassing reads, i.e. one read of a pair on gene 1, and the other on gene2 (deFuse).

$ RescuedCount:

- the number of reads overlapping the break-point rescued after identification of the break point (FusionMap).
- Encompassing reads (Tophat-fusion, FusionCatcher).
- Both spanning and encompassing reads (ChimeraScan, STAR, Rsubread).

$ SplicePattern: the splice pattern for a fusion junction

$ FusionGene: the name of the fusion gene in the format gene1 -> gene2.

$ frameShift: frameshift at break-point

@ *fusionLoc*: A GRangesList encompassing genomic coordinates for gene 1 and 2.

@ *fusionRNA*: A DNAStringSet encompassing the fusion transcript sequence.

@ *fusionGA*: A GAlignments object encompassing positions for all reads mapping on the DNAStringSet representing the fusion.

Methods:

fusionData is an accessory function used to retrieve the list in @*fusionInfo* slot.

fusionGRL is an accessory function used to retrieve from @*fusionLoc* the GRangesList encompassing fusion locations for gene 1 and 2.

fusionRNA is an accessory function used to extract from @ *fusionRNA* the DNAStringSet encompassing the fusion sequence.

addRNA is an accessory function used to add a DNAStringSet encompassing the fusion sequence into the @*fusionRNA* slot.

fusionGA is an accessory function used to extract the GAlignments object encompassing positions for all reads mapping on the fusion from the @*fusionGA* slot.

addGA is an accessory function used to add the GAlignments object into the @*fusionGA* slot.