M. D. ANDERSON CANCER CENTER

The pyPRADA manual

Version 1.2

PRADA development team
Contact: rverhaak@mdanderson.org

2/15/2014

Contents

OVERVIEW

1.1 Background

Massively parallel mRNA sequencing (RNASeq) provides accurate estimate of the quantity and composition of transcriptome. To facilitate the analysis of paired-end RNAseq data, we developed the **P**ipeline for **R**n**A**seq **D**ata **A**nalysis (PRADA). The main focus of PRADA is to identify gene fusions and estimate gene expression. The BAM files generated by the pipeline are readily compatible with different tools for further downstream analysis.

This manual sought to describe available functionality of each module in PRADA.

1.2 Summary of available modules

PRADA currently supports 6 modules to process and identify abnormalities from RNAseq data:

preprocess	:	Generates aligned and recalibrated BAM files.
fusion	:	Identifies candidate gene fusions.
guess-ft	:	Supervised search for fusion transcripts.
guess-ig	:	Supervised search for intragenic rearrangements.
homology	:	Calculates homology between given two genes.
frame	:	Predicts functional consequence of fusion transcript

1.3 Terminology

Throughout this manual we refer to various terminologies which are commonly used in Next Generation Sequencing (NGS) and gene fusion analysis. The basic input to PRADA is a BAM file which is the binary form of Sequence alignment/map format file.

Template/Insert A sequence part of RNA/DNA which is sequenced on a sequencing machine.

Read A raw sequence with quality scores coming off a sequencing machine

Paired End/ Mate Pair A read pair that is sequenced from each end of the fragmented cDNA.

Discordant Read Pair A read pair where each end maps to distinct genes.

Junction Spanning Read A chimeric read that span a putative junction.

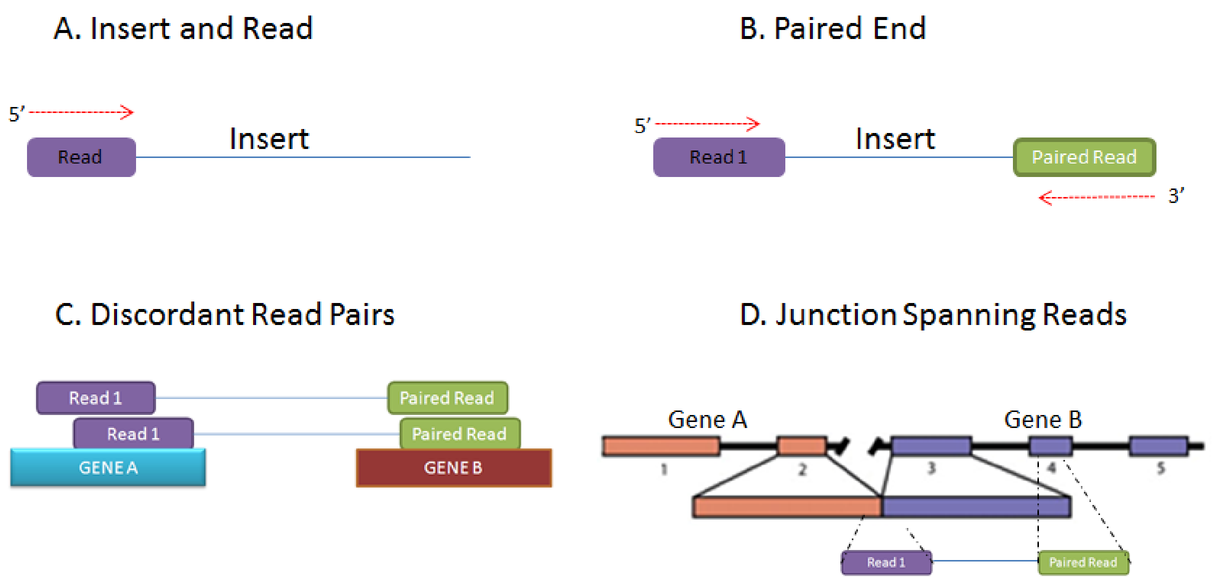


Figure 1: Description of the Terminology

1.5 Implementation and Computing Requirements

pyPRADA was implemented in Python and wraps up following existing tools within the package:

- Samtools
- GATK
- Picard
- BWA
- BLAST+

For further information about the installation refer section 2.

In current release, the "preprocess" module runs on a Portable Batch System (PBS). The current version of PRADA has been extensively tested on MD Anderson Cancer Center High Performance Cluster. Due to the involvement of several memory/time consuming steps, preprocess module requires 8G memory and at least 8 computing CPUs. Gene fusion/guess modules require less resource (4G memory).

INSTALL

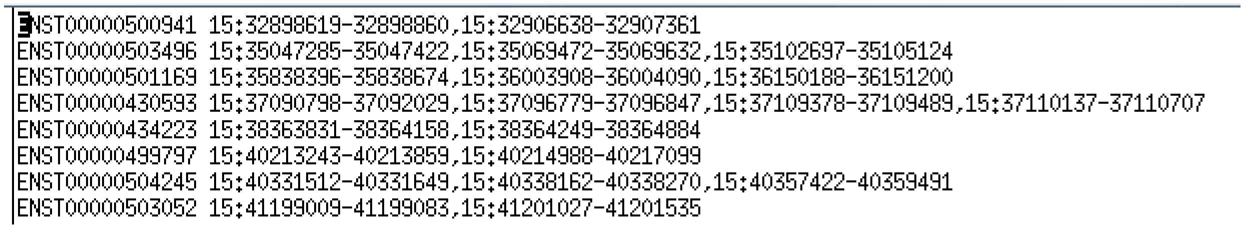
PRADA is written in python programing language and intended to run in a command line environment on UNIX or LINUX operating systems. To run pyPRADA, download the pre-compiled package and unzip to preferred installation location. Email to PRADA authors to request for the references files or download from https://bcbweb.mdanderson.edu/main/PRADA:Overview. If you are running PRADA within MD Anderson Cancer Center, the references files are available in HPCC and DQS extraspace.

2.1 Setting up the configuration file (conf.txt)

Once the pyPRADA package is downloaded and extracted, update the conf.txt file based on where the reference files are stored. Users may download reference files at http://bioinformatics.mdanderson.org/Software/PRADA/. The configure file also includes parameters for BWA alignments, recognizing the possible customization required by users. However, we note that the current pyPRADA setting has been extensively tested. Unless end users have a clear understanding of the parameters, we do not recommend changes of these parameters. Users may also update parameters for PBS scripts. To guarantee extensibility, pyPRADA directly incorporates all inputs from this block ("**—PBS**—") into the resultant PBS files. The minimum requirement is ***node*** and ***ppn***. The number of ppn should be equal to ***BWA aln –t*** to ensure compatible and maximal performance. The parameters have to be set accordingly as each run of the processing module requires different memory and runtime based on the size of the sample. According to our experiences the default setting of 12 CPUs and 5 days is acceptable to the majority of cases.

Update the reference to the files and check if all the files are available in the reference folder. Please make sure not to modify the key for different reference files defined in conf.txt. Please do not modify the title of each block such as "–PBS—" and "—REF—".

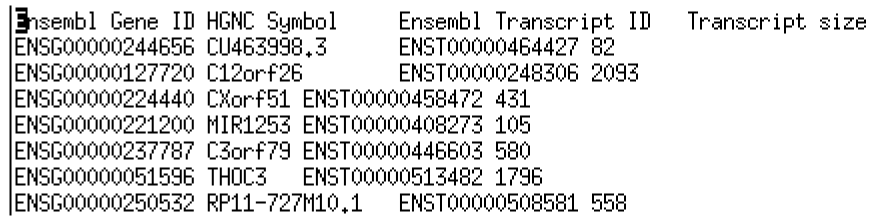
compdb_fasta	=	FASTA file with both genome and transcriptome.
compdb_fai	=	Index to the FASTA file generated using samtools.
compdb_map	=	Tab-delimited file with chromosome boundaries along with transcripts and their exon boundaries information .*



```
ENST00000500941 15:32898619-32898860,15:32906638-32907361
ENST00000503496 15:35047285-35047422,15:35069472-35069632,15:35102697-35105124
ENST00000501169 15:35838396-35838674,15:36003908-36004090,15:36150188-36151200
ENST00000430593 15:37090798-37092029,15:37096779-37096847,15:37109378-37109489,15:37110137-37110707
ENST00000434223 15:38363831-38364158,15:38364249-38364884
ENST00000499797 15:40213243-40213859,15:40214988-40217099
ENST00000504245 15:40331512-40331649,15:40338162-40338270,15:40357422-40359491
ENST00000503052 15:41199009-41199083,15:41201027-41201535
```

Figure 2: Sample MAP file

genome_fasta	=	FASTA file of the whole genome.
genome_gtf	=	GTF with gene annotation information.
dbsnp_vcf	=	DB-SNP know SNPs in VCF format.
select_tx	=	List of all the genes and their longest transcript.*



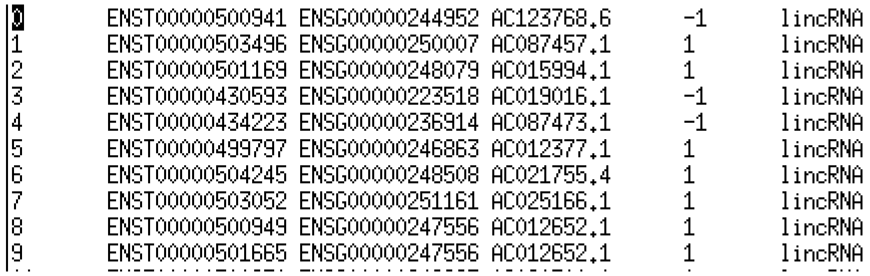
```
Ensembl Gene ID HGNC Symbol     Ensembl Transcript ID   Transcript size
ENSG00000244656 CU463998.3      ENST00000464427 82
ENSG00000127720 C12orf26        ENST00000248306 2093
ENSG00000224440 CXorf51 ENST00000458472 431
ENSG00000221200 MIR1253 ENST00000408273 105
ENSG00000237787 C3orf79 ENST00000446603 580
ENSG00000051596 THOC3   ENST00000513482 1796
ENSG00000250532 RP11-727M10.1   ENST00000508581 558
```

Figure 3: Sample Longest Transcript file

feature_file = Transcript information based on genomic location.

tx_seq_file = FASTA file of the whole transcriptome.

ref_anno = Annotation of transcripts with custom unique id.*

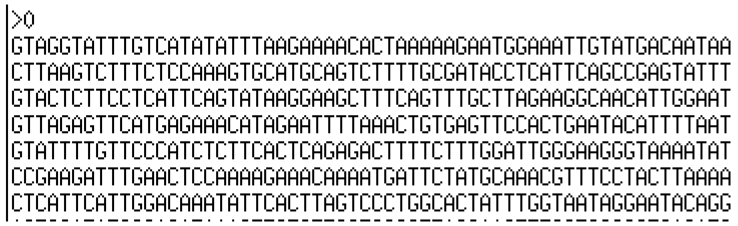


```
0       ENST00000500941 ENSG00000244952 AC123768.6      -1      lincRNA
1       ENST00000503496 ENSG00000250007 AC087457.1      1       lincRNA
2       ENST00000501169 ENSG00000248079 AC015994.1      1       lincRNA
3       ENST00000430593 ENSG00000223518 AC019016.1      -1      lincRNA
4       ENST00000434223 ENSG00000236914 AC087473.1      -1      lincRNA
5       ENST00000499797 ENSG00000246863 AC012377.1      1       lincRNA
6       ENST00000504245 ENSG00000248508 AC021755.4      1       lincRNA
7       ENST00000503052 ENSG00000251161 AC025166.1      1       lincRNA
8       ENST00000500949 ENSG00000247556 AC012652.1      1       lincRNA
9       ENST00000501665 ENSG00000247556 AC012652.1      1       lincRNA
```

Figure 4: Sample annotation file.

ref_map = Tab-delimited file with transcripts and their exon boundaries information .*

ref_fasta = FASTA file of transcriptome with unique id from ref_anno as FASTA headers.*



```
>0
GTAGGTATTTGTCATATATTTAAGAAAACACTAAAAAGAATGGAAATTGTATGACAATAA
CTTAAGTCTTTCTCCAAAGTGCATGCAGTCTTTTGCGATACCTCATTCAGCCGAGTATTT
GTACTCTTCCTCATTCAGTATAAGGAAGCTTTCAGTTTGCTTAGAAGGCAACATTGGAAT
GTTAGAGTTCATGAGAAACATAGAATTTTAAACTGTGAGTTCCACTGAATACATTTTAAT
GTATTTTGTTCCCATCTCTTCACTCAGAGACTTTTCTTTGGATTGGGAAGGGTAAAATAT
CCGAAGATTTGAACTCCAAAAGAAACAAAATGATTCTATGCAAACGTTTCCTACTTAAAA
CTCATTCATTGGACAAATATTCACTTAGTCCCTGGCACTATTTGGTAATAGGAATACAGG
```

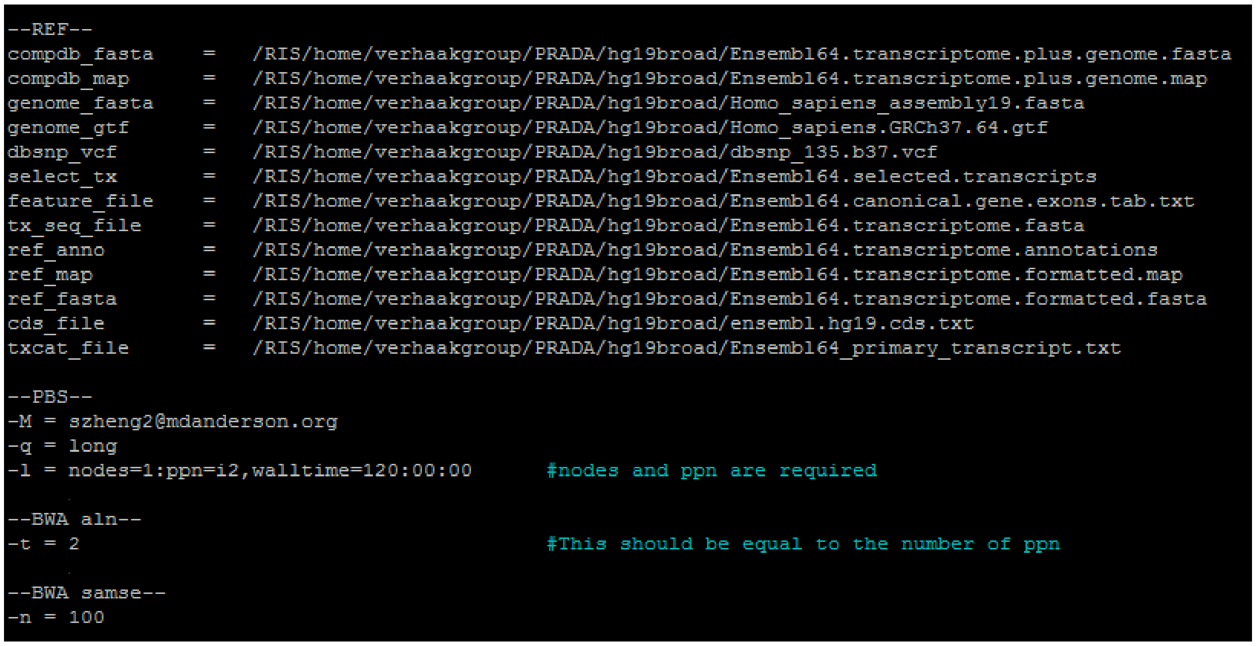
Figure 5: Sample custom FASTA file with modified header.

cds_file = Coding region start and end from ENSEMBL.

txcat_file = Primary transcript for each gene from ENSEMBL.

* Indicates that the files are custom generated and cannot be downloaded from any of the databases.

Once the configuration file is updated it should look like the figure below:



```
--REF--
compdb_fasta    =   /RIS/home/verhaakgroup/PRADA/hg19broad/Ensembl64.transcriptome.plus.genome.fasta
compdb_map      =   /RIS/home/verhaakgroup/PRADA/hg19broad/Ensembl64.transcriptome.plus.genome.map
genome_fasta    =   /RIS/home/verhaakgroup/PRADA/hg19broad/Homo_sapiens_assembly19.fasta
genome_gtf      =   /RIS/home/verhaakgroup/PRADA/hg19broad/Homo_sapiens.GRCh37.64.gtf
dbsnp_vcf       =   /RIS/home/verhaakgroup/PRADA/hg19broad/dbsnp_135.b37.vcf
select_tx       =   /RIS/home/verhaakgroup/PRADA/hg19broad/Ensembl64.selected.transcripts
feature_file    =   /RIS/home/verhaakgroup/PRADA/hg19broad/Ensembl64.canonical.gene.exons.tab.txt
tx_seq_file     =   /RIS/home/verhaakgroup/PRADA/hg19broad/Ensembl64.transcriptome.fasta
ref_anno        =   /RIS/home/verhaakgroup/PRADA/hg19broad/Ensembl64.transcriptome.annotations
ref_map         =   /RIS/home/verhaakgroup/PRADA/hg19broad/Ensembl64.transcriptome.formatted.map
ref_fasta       =   /RIS/home/verhaakgroup/PRADA/hg19broad/Ensembl64.transcriptome.formatted.fasta
cds_file        =   /RIS/home/verhaakgroup/PRADA/hg19broad/ensembl.hg19.cds.txt
txcat_file      =   /RIS/home/verhaakgroup/PRADA/hg19broad/Ensembl64_primary_transcript.txt

--PBS--
-M = szheng2@mdanderson.org
-q = long
-l = nodes=1:ppn=i2,walltime=120:00:00      #nodes and ppn are required

--BWA aln--
-t = 2                                      #This should be equal to the number of ppn

--BWA samse--
-n = 100
```

Figure 6: Sample configuration file

The PRADA development team is aware of redundant reference files, as many pieces of information could be extracted from the GTF file and composite FASTA file. We are working on restructuring the io module so that it requires a minimum set of files. We expect to have a much shorter list in the next version.

2.2 Installation of required Software and Packages

It is assumed that C, Perl v5.8.8 and Java v1.7.0 are installed, along with a PBS job scheduler with sufficient amount of memory and nodes are available. We note that the PBS script generated by pyPRADA is also a shell script that can be deployed in both T shell and B shell regardless of the presence of a PBS scheduler. pyPRADA requires installation of Python and several third party packages to run the pipeline.

2.2.1 Install Python

Enter the following commands to download and extract Python 2.7 to your hosting account.

wget http://www.python.org/ftp/python/2.7.2/Python-2.7.2.tgz

tar zxfv Python-2.7.2.tgz

cd Python-2.7.2

./configure --prefix=$HOME/python

make

```
make install
```

To make python2.7 as your default python, modify the *.bashrc* file (in B shell) in home directory.

```
export PATH=$HOME/python/Python-2.7.2/bin/:$PATH
```

Save .bashrc and source the file or re-login to make the changes effective.

```
source ~/.bashrc
```

In C-shell system, modify *.cshrc* file as following

```
setenv PATH $HOME/python/Python-2.7.2/bin:$PATH
```

Save and source .cshrc file.

2.2.2 Installation of Python packages Pysam and Biopython

Enter the following commands to download and extract Pysam-0.6 [5].

```
wget https://pysam.googlecode.com/files/pysam-0.6.tar.gz

tar zxfv pysam-0.6.tar.gz

cd pysam-0.6


python setup.py install

wget http://biopython.org/DIST/biopython-1.60.tar.gz

tar zxfv biopython-1.60.tar.gz

cd biopython-1.60

python setup.py install
```

2.2.3 set pyPRADA to system path

You may set pyPRADA to system path. To do so, modify .bashrc file in B-shell or .cshrc in C-shell. See 2.2.1.

USAGE

Below are examples of typical PRADA usage. Using the “-h” option with PRADA modules will report a list of all command line options.

3.1 preprocess

As TCGA RNA-seq data is provided as BAM files, PRADA is by default set up to start from BAMs. However, PRADA also works with FASTQ files – please see below.

START FROM BAM: PRADA provides two versions of the preprocess module, determined by two centers in TCGA that produce slightly different RNA-seq BAM files: UNC (prada-preprocess-unc) and the Broad Institute (prada-pre-process-bi). Both the modules are identical except for the first step where we convert BAM to FASTQ. In the bi version we use Picard SamToFastq function and in the unc version we use the ubu sam2fastq function. UNC generated BAMs have compatibility issues with standard BAM due to the MapSlice alignment strategy. For this reason Picard SamtoFastq does not work for them. More details about the incompatibility can be found at https://webshare.bioinf.unc.edu/public/mRNAseq_TCGA/UNC_mRNAseq_summary.pdf. We recommend testing both versions, to determine which works best for your BAM.

START FROM FASTQ: To start PRADA from FASTQ files, paired end FASTQ files should be named to XX.end1.fastq and XX.end2.fastq. PRADA requires strict naming convention otherwise the package will report error. Usage is similar to running the prepocess module with BAM files, only starting from step 2_e1_1.

Examples for both options are included in the package under folder name *testdata*.

Running the preprocess module: The usage is identical in both unc/bi versions:

prada-preprocess-unc -h

```
Pipeline for RNAseq Data Analaysis - preprocessing pipeline (PRADA).

    **Command**prada-preprocess -bam XX. -conf xx.txt -inputdir ..  -tag TCGA-XX -platform illumina -
step 1_1 -intermediate no --pbs xxx -outdir ... -submit no

    **Parameters**:

    -h              print help message

    -step_info      print complete steps and output files of each step.

    -bam            input BAM file, must has a .bam suffix.

     -ref           config file for references and parameters. Default is conf.txt in py-PRADA installation
                    folder. For reference files, each line is separated. The first field is keyword and should
                    not be changed. The second field refers to annotation files.  PBS parameters are for
                    only required in process module.

     -tag           a tag to describe the sample, likely sample ID, such as TCGA-LGG-01; no default. Tag is
                    used to define READ GROUP in the pipeline.

    -platform       only illumina at present (default).

     -step          values: 1_1/2,2_e1/2_1/2/3/4,3_e1/2_1/2,4_1/2,5,6_1/2,7,8; example 2_e1_1; no
                    default. See –step_info for details.

    -outdir         output dir. Default is the directory where the input bam is.

    -pbs            name for output pbs file and log file. Default (time-stamp) is used if no input.

    -intermediate   values:yes/no; if intermediate files should be kept. Default is not.

    -submit         If submit the job to HPC, default is no.

     -v             print version information.
```

Example:

We assume the current working folder is the PRADA directory. A pbs file will be generated and automatically submitted for each operation. If no submission is desired, change “-submit yes” to “no”. A message will be printed to stdout detailing the locations of the input BAM file, PBS file and resulting LOG file.

1. Run PRADA from BAM

./prada-preprocess-bi -inputdir testdata -sample U87_chr17p13.2 -tag U87 -step 1_1 -pbs frombam -outdir testdata/test1 -submit yes

2. Run PRADA from FASTQ

./prada-preprocess-bi -inputdir testdata -sample U87_chr17p13.2 -tag U87 -step 2_e1_1 -pbs fromfq -outdir testdata/test2 -submit yes

IMPORTANT: the job file can be run as a standalone shell script (using sh <name of script> or ./<name of script>.

3.2 fusion module

Running pyPRADA gene-fusion module:

The command line parameters required to run prada-fusion are described in the -help option.

$ prada-fusion -h

```
Pipeline for RNAseq Data Analaysis - fusion detection (PRADA).
    **Command**:
    prada-fusion -bam XX.bam -conf xx.txt -tag XX -mm 1 -junL XX -minmapq 30 -outdir ./
    **Parameters**:
    -h         print help message
    -bam       input BAM file, must has a .bam suffix. BAM is the output from PRADA preprocess
               module. CAUTION: a non-PRADA BAM may give erroneous results.
    -conf      config file for references and parameters. Use conf.txt in py-PRADA installation folder if
               none specified.
    -tag       a tag to describe the sample, used to name output files. Default is ''.
    -mm        number of mismatches allowed in discordant pairs and fusion spanning reads. Default is 1.
    -junL      length of sequences taken from EACH side of exons when making hypothetical junctions.
               No default. See details below.
    -minmapq   minimum read mapping quality to be considered as fusion evidences. Default is 30.
    -outdir    output directory.
    -v         print version
```

NOTE: -junL parameter defines how the junctions are generated. For a pair of exons, the hypothetical junction is generated by adopting the 3'end junL bases from exon 1 and 5' end junL bases from exon 2. There is no default value for -junL, it should be set based on the read length and quality of the bases at each end of the read. The number must be less than the standard read length. For example if the read lengths of each end in a high throughput experiment is 50bp. Then the -junL can be set anywhere from 35 to 49. Based on this number the exon-junction database is created, which is used to find junction spanning reads which imply the existence of gene fusions. If you assign the junL value as 45, then PRADA generates a database of all possible exon junctions for genes within gene-pair and the length of each sequence constituting the exon junction will be 90 bases. The sequence includes last 45 bases of an exon from 5' gene (geneA) and first 45 bases from the beginning of an exon in the 3' gene (geneB). We recommend setting junL as the 80% of read length. For instance, if read length is 75, use junL=60. Unmapped reads with mate pair mapping to the genes in the recurrent gene-pairs list are aligned to the junction database to identify the junction spanning reads. If the -junL is set more than the read length then the reads would not map the junction.
Example:

```
[MDAXX]$ prada-fusion -bam TCGA-A3-xxxx-01A-xxR-xxxx-xx.withRG.GATKRecalibrated.flagged.bam –conf conf.txt
-tag TCGA-A3-xxxx -mm 1 -junL 40 -minmapq 37 -oudir ./fusion
```

3.3 guess-ft

Guess fusion transcript is to search for evidences of fusions between GeneA and GeneB. The same biological logic is applied as prada-fusion, but instead in a supervised fashion.

```
python prada-guess-ft -h

  Usage: prada-guess-ft GeneA GeneB -conf xx.txt -inputbam X -mm 1 -minmapq 30 -junL X -outdir X -
         unmap X

  **Parameters**:
     -conf          the configure file. see prada-fusion -ref for details
     -inputbam      the input bam file
     -mm            number of mismatch allowed. Default is 1.
     -minmapq       mininum mapping quality for reads to be used in fusion finding
     -junL          length of exons to be used for junctions. see prada-fusion
     -outdir        output directory
     -unmap         the numapped reads. useful if user need to run guess-ft multiple times
```

Example:

Run guess to search for FGFR3-TACC3 fusion:

[MDAXX]$ prada-guess-ft FGFR3 TACC3 -conf conf.txt -inputbam TCGA-A3-xxxx-01A-xxR-xxxx-xx.withRG.GATKRecalibrated.flagged.bam -mm 1 -junL 40 -minmapq 30 -outdir ./guess

Run guess to search for EGFR-TACC3 fusion by setting unmapped.bam from earlier run:

[MDAXX]$ prada-guess-ft EGFR TACC3 -inputbam TCGA-A3-xxxx-01A-xxR-xxxx-xx.withRG.GATKRecalibrated.flagged.bam -mm 1 -junL 40 -minmapq 30 -outdir ./guess –unmap unmapped.bam

3.4 guess-if

The guess intragenic fusion usage is quite similar as guess-ft. Only the gene in which the intragenic deletion has to be deleted must be mentioned here.

$ prada-guess-if

```
Usage: prada-guess-if Gene -conf xx.txt -inputbam X -mm 1 -minmapq 30 -junL X -outdir ./ -unmap X

  **Parameters**:

     -conf          the configure file. see prada-fusion -ref for details

     -inputbam      the input bam file

     -mm            number of mismatch allowed

     -minmapq       mininum mapping quality for reads to be used in fusion finding

     -junL          length of exons to be used for junctions. see prada-fusion

     -outdir        output directory
```

Example:

Run guess to search for EGFR-TACC3 fusion:

[MDAXX]$ prada-guess-if EGFR -conf conf.txt -inputbam TCGA-A3-xxxx-01A-xxR-xxxx-xx.withRG.GATKRecalibrated.flagged.bam -mm 1 -junL 40 -minmapq 30 -outdir ./guess

Run guess to search for EGFR-TACC3 fusion by setting unmapped.bam from earlier run:

[MDAXX]$ prada-guess-if EGFR -inputbam TCGA-A3-xxxx-01A-xxR-xxxx-xx.withRG.GATKRecalibrated.flagged.bam -mm 1 -junL 40 -minmapq 30 -outdir ./guess –unmap unmapped.bam

3.5 frame

Used to classify list of gene fusions and their junction sites to in-frame or out-frame. The input file is a tab delaminated file with gene names and the junction boundary location.

$ prada-frame -h

```
Usage: prada-frame -conf xx.txt -i inputfile -docstr XX -outdir ./

**parameters**

   -conf      see prada-fusion -conf

    -i        input file, a tab/space delimited four-column file, each row like "GeneA GeneA_break GeneB
              GeneB_break"  Example:FGFR3 1808661 TACC3 1739325

   -docstr   outfile prefix. output to two files: XX.frame.brief.txt, XX.frame.detail.txt

   -outdir    output directory, default is ./
```

Example:

Run frame to classify fusion:

[MDAXX]$ prada-frame -conf conf.txt -i input.txt –docstr output -outdir ./guess

3.6 homology

Used to obtain homology score between the two genes in the fusion.

$ prada-homology

```
Usage: prada-homology -conf xx.txt -i inputfile -o outputfile -tmpdir XXX

**parameters**

   -conf       see prada-fusion -conf for details

   -i          a tab/space delimited two-column file --- gene1 gene2

   -o          output file

   -tmpdir     temporary directory that saves fasta files
```

Example:

$ prada-homology -conf conf.txt -i homology_test.txt -o homology_scores -tmpdir ./GUESS_ft

3.7 Quick examples

In this release, we provide a test data set in the package under the folder "*testdata*". This small data set was excerpted from U87 RNAseq (SRA accession number: SRX070907) but only contains 262,009 read pairs mapping to cytoband chr17p13.2. A gene fusion SPAG7-CAMTA2 was found in this region by PRADA, defuse and TopHat-Fusion.

In the folder we also included *frame_test.txt* for testing frame prediction module, *homology_text.txt* for testing homology prediction module. Example command lines were included in the readme.txt file. The preprocessing takes 20-30 minutes to finish.

DESCRIPTION

4.1 PROCESSING

The purpose of the processing module is to realign short reads to both transcriptome and genome. Although time consuming, this step is essential because a given input BAM file may be generated by very different sequence alignment tools and thus vary substantially in read alignment structure. This module first converts the BAM file into a FASTQ file and aligns them to ref genome and transcriptome using BWA, and remaps the resulting BAM file to convert the transcriptome locations to genome locations. Then, it merges the two ends into a paired BAM file and recalibrates the quality scores for the BAM file. Finally it marks the duplicate reads and generates index file for further downstream analysis. The two step alignment strategy is essential in RNAseq processing. For details, see M. Berger et al, Genome Res, 2010.

These steps are performed using multiple tools like BWA, GATK, Picard and samtools. The pyPRADA module generates a PBS file which calls the steps one after the other in a sequential format. The tools are wrapped along with the pyPRADA package. For system without PBS or similar scheduler installed, users the PBS file can be used simply as a *shell* script.

pyPRADA can be re-started at any step by passing parameter step (e.g. -step 1_1/2,2_1/2/3_e1/e2,3_1/2_e1/e2,4_1/2,5,6_1/2/3,7_1/2,8_1). To start pyPRADA from an intermediate step, it requires the output files generated in the previous step. If the files are deleted then start it from the first step.

Note: After each step the files generated in the previous step are deleted. Please set the parameter -intermediate as YES if you want to keep the intermediate files. This will help the user debug errors in the pipeline.

STEP 1

Based on how the input BAM file was generated, pyPRADA provides two versions of the processing step which differ in the conversion of BAM to FASTQ step, one uses Picard SAMtoFASTQ and other uses ubu-1.2-jar-with-dependencies.jar which was referred by UNC for samples processed by them (MAPSPLICE aligner). UBU requires the initial BAM file to be sorted by read name so in the initial step 1_1 it sorts the BAM file using samtools and in step 1_2 it converts the BAM to FASTQ file. Whereas, if you use the Picard version of the pyPRADA you will have just one step 1_1 which converts BAM to FASTQ file.

STEP 2

BWA (Burrows-Wheeler Alignment) is used in the second step of pyPRADA to align the reads to the reference genome and transcriptome. The publicly available version of BWA usually reports one alignment per read. Nonetheless to benefit from the PRADA alignment strategy of mapping to both genomic and transcriptomic references, BWA was customized to print all the alignments a read contains. This will allow conserving the alignment information from both genome and transcriptome. Custom BWA is used to align the FASTQ files to both genome and transcriptome at once. All the required reference files are defined in the config file. The reference file contains both the genome and transcriptome sequence information in a concatenated form to perform alignment at once. If errors occur in this step please re-run the same step to ensure it is not an error from the Linux server but from the tool. If the error is from the BWA tool then search for the encountered error in the BWA mailing list and debug the error accordingly. If an error occurs while processing the step 2, pyPRADA can be re-submitted from the current step by setting the parameter -step="2_e1_1" or -step="2_e2_1". However, this requires the existence of both end [1-2] FASTQ files generated in the first step. If those files are deleted then the pipeline needs to be run from the beginning.

The step 2_2 aims to generate SAM (Sequence Alignment/Map) files from its .sai index and reference sequence files using a customized version of BWA. SAM is the generic format for storing large nucleotide sequence alignments data. The parameter '-s' in the customized BWA allows the reporting of multiple alignments per read which enables the integration of genome and transcriptome mapping. Same as step 2_1, the generation of SAM file includes two steps 2_e1_2 and 2_e2_2, these steps are called sequentially to process end1 and end2 SAI files respectively. If errors occur resubmit the step to ensure errors are not related to Linux server or job scheduler. If errors persist in this step proceed to search in the BWA mailing list for a fix. Some errors might be related to the customization done to BWA. Please email to PRADA authors in regards with specific errors related to the customization done to BWA for debugging help.

In step 2_3 aims to generate sorted BAM file from the SAM file. BAM is the binary version of the SAM file and generated using the samtools view command. Samtools view function is used to convert the SAM to BAM in step 2_e1_3 and 2_e2_3.

In step 2_4 the resulting BAM file is sorted by read name using samtools sort function. Samtools perform merge sort which generates number of temporary files which are automatically deleted at the end of sorting step. The sorting step is time consuming and performing the step twice for each end is one of the bottle necks of processing steps. If there are errors resubmit the step to ensure no errors were caused from Linux server or job scheduler by passing parameter -step 2_e1_4 or 2_e2_4 based on which step the error occurred. If the input SAM files are missing then you will have to re-start from step 2_3 or step 1. If the errors persist, search for a fix in samtools mailing list.

Once the SAM file is generated, all the intermediate ".sai", ".fastq" and ".sam" files will be deleted. The parameter -intermediate should be set to YES to keeping the intermediate files.

STEP 3

Earlier, the reads are aligned to both the genome and transcriptome. In this step the remapTranscriptome tool from GATK is used to map transcript placements to genomic coordinates, preserving intronic information and filtering reads with ambiguous placements. This tool requires the transcriptome to genome mapping information and the genome reference file to perform remapping. The resulting remapped BAM file is named as ".remapped.bam". For more information on how this tool works please refer to Michael F. Berger et.al. 2010 [2]. This jar executable is customized and is not currently available in the public release of GATK 1.5-32 or earlier versions. No help information is found in the mailing list or FAQ of GATK since this is an unpublished custom utility. Re-running the step by using -step 3_e1_1 or 3_e2_1 is encouraged if errors are encountered. If the sorted bam files are missing than re-start the pipeline from step 1 or 2 based on the availability of input files. Otherwise try contacting the authors of PRADA if errors persist.

The step 3_2 is identical to the 2_4 step; the remapped BAM file is sorted by read name using samtools sort function. Similarly, this step also has 3_e1_2 and 3_e2_2 to sort end1 and end2 remapped bam files respectively and the sorted files have an extension "sorted.remapped.bam". If there are errors in this step, re-run the current step by passing the parameter -step 3_e1_2 or 3_e2_2 based on the step in which the error occurred. If the input files are missing then restart from step 3_1 or earlier steps based on the availability of required files. Once the sorted remapped bam files are generated, the intermediate sorted.bam and remapped.bam files are deleted.

STEP 4

Up to this step, the RNAseq data is analyzed separately for end1 and end2. In this step the processed BAM ("sorted.remapped.bam") files from both ends are combined into one bam file with name extension "paired.bam". The FLAGS for each mapped reads are updated by 0x40 and 0x80 which indicates the first and last segment in the template. The tool pairMaker is a customized jar executable which is not currently available in the public release of GATK 1.5.32 or earlier versions. If errors are encountered at this step resubmit by assigning the parameter -step 4_1. If the "sorted.remapped.bam" files are missing than re-start the pipeline from step 3 or 1 based on the availability of input files. Since this is a customized tool, no documentation can be found through the GATK website. If you need further information to debug errors or understand the process please contact the authors of PRADA.

The step 4_2 is similar to the sorting done in step 2 and 3. Nonetheless, the sortSam tool from Picard is used to sort the paired bam file by genomic coordinates (i.e. coordinate). The resulting sorted paired BAM file is named as ".paired.sorted.bam". If errors are found re-run the current step by passing the parameter -step 4_2. If the input files are missing, please restart from step 3 or earlier steps based on the availability of intermediate files.

STEP 5

In this step the read group information is added to the sorted paired data. Read group information is contained within the header of a SAM file in the form of a line. In the SAM header the line starting with @RG represents the read group. The read group information includes: (a) platform used, (b) DNA library description, (c) unique id for the experiment, and (d) the lane in which the data was sequenced. For more information on the read group can be found in the SAM manual [1] . This particular line is required in the header file for GATK to read the BAM or SAM files. Hence, read group is added before the step 6 which uses GATK functions for recalibration. The required parameters for read group are passed through required input -platform and -tag. The resulting bam file with read group is named as ".withRG.paired.sorted.bam". If errors are found in this step, restart this step by assigning the parameter -step 5_1. If the input files are missing than re-start processing module from step 4 or 1 based on the availability of input files. The customized GATK tools used in step 3 and 4 does not require @RG info.

STEP 6

In this step the sorted paired data is subjected to base quality recalibration. The whole process is defined in two steps, the first step is performed in the step 6_2, and the tool CountCovariates from GATK is used to generate a summary table constituting all possible combination of bases at every position across the read length. In the 6_3 step the tool TableRecalibration is called from GATK 1.5 version to generate the recalibrated BAM files.

In step 6_1 PRADA indexes the "withRG.paired.sorted.bam" as the index is a required input for CountCovariates in the next step. The indexing is done using samtools index function which generates a .BAI file. The generated index file will have the same file name as the bam file but the file extension is .bai instead of .bam. The pipeline then proceeds to call the CountCovariates from GATK version 1.5. The tool CountCovariates is used to generate a comma delimited table with information related to recalibrated quality score which are stored into a file with name extension ".orig.csv". These tables are used as input to generate the recalibrated files later. Visit the GATK website to gather information on how the recalibrations are performed.

In step 6_2 it calls the TableRecalibration from GATK version 1.5. The tool TableRecalibration uses the tables from 6_2 to perform recalibration and print the BAM file. The input file includes the BAM file, reference fastq files and the GATK report ".grp" file as input along with the table. It generates a BAM file named as ".GATKRecalibrated.bam". Once the recalibrated BAM files are generated without any errors the sorted paired bam files is deleted. If there are errors then restart the pipeline from step 6_2. In case that the input files are missing then call the pipeline from step 6_1 or early steps based on the availability of the intermediate files.

NOTE: The TableRecalibration tool was replaced by PrintReads and CountCovariates tool was renamed as BaseRecalibrator in GATK 2.0 and both the earlier and new version of the tools have similar functionality.

STEP 7

In this step all the duplicate reads are flagged in the recalibrated bam file. The duplicate reads are only flagged by assigning them a flag 0x400 and they are not deleted. In this level Picard tool MarkDuplicates is used to flag the duplicate reads from the withRG.GATKRecalibrated.BAM file and it generates a withRG.GATKRecalibrated.flagged.bam file. Once the flagged BAM file is generated without any errors, the input recalibrated BAM file is deleted. If there are errors then restart the pipeline from step 7_1. In case that the input files are missing then call the pipeline from step 6 or early steps based on the availability of the intermediate files. This is the final output file for pyPRADA data processing module together with its index produced by this step.

STEP 8

In the 8th step which was earlier called as expression module in Perl version of PRADA is used to generate RPKM and quality control statistics. These metrics are generated by RNASeQC jar executable [3]. This tool requires the input files to be indexed, sorted by location, flagged for duplicates and includes read group information. These are performed in the PRADA's data processing module. To generate RPKM values for files that are processed by different aligners then make sure the pre-processing steps are performed as mentioned in the RNASeQC documentation and then follow the naming constraints (i.e. .withRG.GATKRecalibrated.flagged.bam) and call step 8_1. More information can be obtained at RNASeQC documentation site [4].

4.2 GENE FUSION

Provided the PRADA processed BAM files, the gene fusion module extracts a list of discordant read pairs. From the list of discordant pairs it will generate a list of gene pairs, these discordant read pairs map to and filters recurrent gene pairs. For these recurrent genes the gene fusion module will identify the junction spanning reads. Finally, summarizes the results of all the fusions into a text file by printing out read ids of the discordant pairs and spanning reads. As, a last step it will also perform a pairwise alignment of the two genes within a candidate gene-pair to provide homology scores for the end user for further filtration.

The fusion module requires number of python modules most of which are available by default in Python. Pysam [5] and Biopython [6] are two modules which are required to download from the web. Since pyPRADA fusion module usually takes less than 2 hours, we do not provide the functionality of resuming analysis at a middle point.

FUSION-STEPS

STEP 0: Loading of gene annotations.

In this step pyPRADA loads gene, transcript and exon information from annotations files defined in the config file. Each entity is modeled by a virtual class. Transcript object is comprised of exon objects, and subsequently gene object is comprised of transcript objects. These classes are used internally by pyPRADA.

STEP 1: Finding Discordant Pairs.

Discordant read pairs are paired read ends that are mapping uniquely (i.e. mapping quality equal to 37) to different protein-coding genes with orientation consistent to sense-sense chimera. Mitochondrial genes, clone IDs and possible read-through transcripts, which are subsequent gene pairs within 1Mb of each other and same transcription direction, are ignored. If a read maps to overlapping genes, these genes are split up as two different instances.
In this step the recalibrated sorted indexed BAM file is loaded to extract mate pairs mapping to distinct genes. First, for each gene it will extract all the reads mapping to it and then filters for unique reads with high mapping quality and checks if those reads mate pair map to different genes. Later, it will check if orientation of the discordant read pairs is consistent with the strand information of the genes they are mapping to and assigned an orientation 'R' if the read maps to the reverse strand of the residing gene, inversely ‘F’. The discordant read pairs with F and R orientation at either ends are printed into the output file.

STEP 2: Finding Recurrent Pairs
Gene pair represents the genes that a discordant read pair maps to and only gene pairs with at least two discordant read pairs are considered as recurrent gene pairs for further analysis. In this step gene pairs which occur once per sample are filtered and remaining is checked for read through events. This step improves the sensitivity and computational efficiency of the approach as the number of bona-fide gene pairs to be further examined in step 3 is reduced.

STEP 3: Finding Potential Junction Spanning reads.
A list of all the genes in the recurrent gene-pairs is obtained. Next, all the reads mapping the exonic regions of these genes are fetched to extract the reads with unmapped mate pairs. The unmapped end of the mate pair is considered as potential junction spanning read, so the unmapped end of the read is printed into the FASTQ file. The header of each read in the FASTQ file also has information of the gene to which the mapped end of the mate pair maps to and its orientation (F and R from step 1).

STEP 4: Building Junction Database.
All hypothetical exon junctions for a gene pair are created using Ensembl transcriptome sequences as a customized fasta file for all of the recurrent gene pairs. Each fasta sequence has a fixed length and the length is defined from (-junL). The sequence contains the last junL bases in exon from 5' gene and the first junL bases of the exon from 3' gene in the gene pair.

STEP 5: Aligning Potential Junction reads to junction database

Using customized BWA from PRADA processing step to align the unmapped reads from step 3 to the junction database in step 4. From the resulting BAM file uniquely mapped reads are parsed and checked if the genes constituting the mapped junctions are consistent with the genes to which the mate pair are mapped too.

STEP 6: Summarizing Fusion Evidence

In this step the discordant reads from step 2 and the junction spanning reads from step 5 are summarized into the output file. The mismatch filter is also applied in this step and prints candidates and evidences files.

STEP 7: Generating Fusion List

Filters the gene pairs listed in candidates file based on homology and, presence of at least two discordant pairs and one perfect junction spanning read.

OUTPUT:

Discordant represents the discordant reads mapping to the gene pair. Junction Spanning Reads (JSR) are reads which maps to the gene-fusion exon junction and the mate end maps to one of the genes in the gene-pair. Perfect JSR are reads with 0 mis-matches. Junction represents the unique exon junctions that were found in the JSR. Position Consistency indicates if the mapping location of the discordant reads is consistent with the location of the spanning reads.

XXXX.fus.summary.txt

```
Gene_A   Gene_B   A_chr    B_chr    A_strand B_strand Discordant_n      JSR_n    perfectJSR_n      Junc_n
         Position_Consist  Junction Identity Align_LenEvalue  BitScore

GNAS     TSPAN7   20       X        1        1        2        8        6        1        YES
         GNAS:20:57486241_TSPAN7:X:38505515,8       100.00   11       3.1      21.1

CLU      TRIM4    8        7        -1       -1       2        1        1        1        PARTIALLY
         CLU:8:27455977_TRIM4:7:99500938,1 100.00   12       1.3      22.9

CLASP2   B2M      3        15       -1       1        2        1        1        1        PARTIALLY
         CLASP2:3:33661095_B2M:15:45009805,1        100.00   11       3.2      21.1
```

XXXX.fus.candidates.txt

```
Gene_A   Gene_B   A_chr    B_chr    A_strand B_strand Discordant_n      JSR_n    perfectJSR_n      Junc_n
         Position_Consist  Junction
GPX4     GNS      chr19    chr12    1        -1       2        1        0        1        YES
         GPX4:chr19:1056808_GNS:chr12:63417154,1
LILRB4   LILRB1   chr19    chr19    1        1        2        3        2        1        NO
         LILRB4:chr19:59868112_LILRB1:chr19:59837237,3
```

XXXX.fus.evidences.txt

```
@@       LILRB4,chr19,1    LILRB1,chr19,1
```

```
>discordant
UNC9-SN296_172:5:1104:11009:8514#0          F          LILRB4.59869683.mm0
UNC9-SN296_172:5:1104:11009:8514#0          R          LILRB1.59839840.mm0
UNC9-SN296_172:5:2102:8847:42293#0          F          LILRB4.59869504.mm0
UNC9-SN296_172:5:2102:8847:42293#0          R          LILRB1.59839839.mm0

>spanning
UNC9-SN296_172:5:2102:18901:91570#0_prada_LILRB4_prada_F    LILRB4:chr19:59868112_LILRB1:chr19:59837237.mm1
UNC9-SN296_172:5:1202:19802:43551#0_prada_LILRB4_prada_F    LILRB4:chr19:59868112_LILRB1:chr19:59837237.mm0
UNC9-SN296_172:5:1206:14313:83821#0_prada_LILRB4_prada_F    LILRB4:chr19:59868112_LILRB1:chr19:59837237.mm0

>junction
LILRB4:chr19:59868112_LILRB1:chr19:59837237 3

>summary
Number of Discordant Pairs = 2
Number of Fusion Reads = 3
Number of Perfect Fusion Reads = 2
Number of Distinct Junctions = 1
Position Consistency = NO
```

4.3 GUESS-ft:

General UsEr defined Supervised Search for fusion transcript. This program is to perform targeted search of gene fusions from RNAseq data. It uses samtools, pysam package. It requires a bam file and other reference files which are included in PRADA package.

Note: It is a less accurate tool than prada-fusion.

DESCRIPTION

The gene symbols of the 5' and 3' genes in the fusion are assigned to GeneA and GeneB respectively. The BAM file to search for the gene-fusion is passed through -inputbam.

The first step in guess-ft is populating the gene, transcript and exon objects defined in Bioclass. This step is identical to the step 0 in the genefusion module of pyPRADA.

Next, all the unmapped reads (flag 0x4) in the inputbam file are extracted using samtools view functions. This is one of the most time consuming step in GUESS module. To speed up the process the -unmapped parameter is provided where the unmapped.bam file generated from earlier run of GUESS for the SAME sample could be set. This will speed up multiple runs of GUESS on same sample.

Later, the junction database is created which has all possible exon junctions between GeneA and GeneB. The length of the sequence is passed as parameter. Using pysam fetch functionality the reads mapped to GeneA and GeneB are extracted from the input BAM file. From these extracted reads, it will check if there are discordant read pairs which are mapping to GeneA and GeneB. Also, it looks for the potential junction spanning read where one end of the read pair maps to the genes and the other end is unmapped. These unmapped reads sequences are extracted from the unmap.bam file generated earlier and saved as FASTQ file.

Finally, the unmapped reads obtained in the previous step are aligned to the junction database using BWA. The resulting SAM file is parsed for reads mapping to junctions and this information along with the discordant read pairs are printed into the output summary file. The output is similar to the gene fusion fus.summary output.

4.4 GUESS-if

General UsEr defined Supervised Search for intragenic fusion. It is an extension of GUESS-ft. This program is to perform targeted search of abnormal multiple exon junctions in a gene, such as EGFRvIII where exon 1 is abnormally connected to exon 8 caused by exons 2-7 deletion. It uses samtools, pysam package. It requires a bam file and other reference files which are included in PRADA package.

The gene symbols of the gene to search for intragenic deletion is passed a parameter. The BAM file to search for the gene-fusion is passed through -inputbam.

The first step in guess-if is populating the gene, transcript and exon objects defined in Bioclass. This step is identical to the step 0 in the genefusion module of pyPRADA.

Next, all the unmapped reads (flag 0x4) in the inputbam file are extracted using samtools view functions. This is one of the most time consuming step in GUESS module. To speed up the process the -unmapped parameter is provided where the unmapped.bam file generated from earlier run of GUESS for the SAME sample could be set. This will speed up if you want to rerun GUESS on same sample.

Later, the junction database is created which has all possible novel exon junctions within the Gene. Novel exon-junctions are those junctions which are not found in any of the transcripts of the gene. The length of the sequence is passed as parameter. Using pysam fetch functionality the reads mapped to Gene are extracted from the input BAM file. From these extracted reads, it will check if there are reads where one end of the read pair maps to the genes and the other end are unmapped. These unmapped reads sequences are extracted from the unmap.bam file generated earlier and saved as FASTQ file.

Finally, the unmapped reads obtained in the previous step are aligned to the junction database using BWA. The resulting SAM file is parsed for reads mapping to junctions and this information is printed into the output summary file.

OUTPUT:

The output is similar to the fusion detail output file. It has

```
EGFR

>spanning

>junction

>summary
Number of Fusion Reads = 0
Number of Distinct Junctions = 0
```

4.5 Frame

The frame module is used to predict the functional consequences of gene fusions. Possible predictions include classifying the fusion transcript as in-frame, out-of-frame, UTR-UTR, UTR-CDS, Unknown etc. It relies on the definition of transcript CDS/UTR boundaries from ENSEMBL, and gives a prediction for every pair of transcripts coded by the fusion genes. The whole point is to see if the fusion lead to reading shift of the CDS sequences. The result is purely predicted, no guarantee is assumed.

Based on the genes in the fusion pair and the boundaries of the fusion junction provided in the input file, the module identifies all the transcripts of these genes constituting the fusion boundaries. Based on the placement of the junction on the transcripts we tag them as coding (CDS) if they lie within the coding boundaries and as 5' un-translated region (5UTR) or 3' un-translated region (3UTR) if they lie outside the coding boundaries. If the junction lies in the coding site of both the 5' gene and 3' gene in the fusion gene pair, then it identifies the length of the sequence from coding start site to the junction of the transcripts for both the genes. From the length of the open reading frame it derives the position of break points at codon level (position 1-3 in codon) and checks if both the genes have same break points. Finally based on this information it classifies the fusion transcript as in-frame if sequences break point lies in same codon position in both the gene transcripts otherwise defined as out of frame. If the fusion junction lies in coding region for one gene but in 3UTR or 5UTR in other gene then they are classified accordingly. Also, if junction point lies in un-translated regions for both the transcripts then we classify them as 5UTR-3UTR or vice-versa.

This module provides two output files, a brief and detail version. In the brief version it prints the frame classification of only the primary transcripts pair from the two genes in the fusion. The detailed version includes all possible combinations of the transcripts within a fusion.

4.6 Homology

The homology module is used to calculate the sequence similarity of genes. It is used to filter out potential false positive fusions that were incorrectly found due to homology caused misalignments. This module aligns the longest transcripts of the gene pair in a fusion using blastn and provides with e-value and alignment length. The output file is a text file with the gene names, length of the longest transcripts, identity, e-value bit score and alignment length. Homology is also run by default in the gene-fusion module.

OUTPUT:

```
#Gene1   Gene2    Transcript1_Len    Transcript2_Len    Identity   Align_LenEvalue    BitScore

FGFR3    TACC3    4223      2788     100.00    14        0.12       26.5
```

REFERENCE:

1) SAM Manual http://samtools.sourceforge.net/SAM1.pdf
2) Berger MF, Levin JZ, Vijayendran K, Sivachenko A, Adiconis X, Maguire J, Johnson LA, Robinson J, Verhaak RG, Sougnez C, Onofrio RC, Ziaugra L, Cibulskis K, Laine E, Barretina J, Winckler W, Fisher DE, Getz G, Meyerson M, Jaffe DB et al (2010) Integrative analysis of the melanoma transcriptome. Genome Res 20: 413–427
3) Deluca DS, Levin JZ, Sivachenko A, Fennell T, Nazaire MD, Williams C, Reich M, Winckler W, Getz G. (2012) RNA-SeQC: RNA-seq metrics for quality control and process optimization. Bioinformatics
4) RNA-SeQC http://www.broadinstitute.org/cancer/cga/rna-seqc
5) Pysam http://www.cgat.org/~andreas/documentation/pysam/contents.html
6) Cock PJ, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, Friedberg I, Hamelryck T, Kauff F, Wilczynski B, and de Hoon MJ. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics 2009 Jun 1; 25(11) 1422-3. doi:10.1093/bioinformatics/btp163 pmid:19304878.

STEP	INPUT	OUTPUT	COMMAND
1_1	sampleid.bam	sampleid.sorted.bam	/pyPRADA_v1.0beta/tools/samtools-0.1.16/samtools sort -n -m 1000000000 sampleid.bam sampleid.sorted
1_2	sampleid.sorted.bam	sampleid.end[1,2].fastq	java -Xmx8g -jar /pyPRADA_v1.0beta/tools/ubu-1.2-jar-with-dependencies.jar sam2fastq --in sampleid.sorted.bam --fastq1 sampleid.end1.fastq --fastq2 sampleid.end2.fastq --end1 /1 --end2 /2
			rm -f sampleid.sorted.bam
2_e1_1	sampleid.end1.fastq	sampleid.end1.sai	/pyPRADA_v1.0beta/tools/bwa-0.5.7-mh/bwa aln -t 12 $refdir/Ensembl64.transcriptome.plus.genome.fasta sampleid.end1.fastq > sampleid.end1.sai
2_e1_2	sampleid.end1.sai & sampleid.end1.fastq	sampleid.end1.sam	/pyPRADA_v1.0beta/tools/bwa-0.5.7-mh/bwa samse -s -n 100 $redir/Ensembl64.transcriptome.plus.genome.fasta sampleid.end1.sai sampleid.end1.fastq > sampleid.end1.sam
			rm -f sampleid.end1.sai
			rm -f sampleid.end1.fastq
2 e1 3	sampleid.end1.sam	sampleid.end1.bam	/pyPRADA_v1.0beta/tools/samtools-0.1.16/samtools view -bS -o sampleid.end1.bam sampleid.end1.sam
			rm -f sampleid.end1.sam
2_e1_4	sampleid.end1.bam	sampleid.end1.sorted.bam	/pyPRADA_v1.0beta/tools/samtools-0.1.16/samtools sort -n -m 1000000000 sampleid.end1.bam sampleid.end1.sorted
			rm -f sampleid.end1.bam
2 e2 1	sampelid.end2.fastq	sampleid.end2.sai	/pyPRADA_v1.0beta/tools/bwa-0.5.7-mh/bwa aln -t 12 $refdir/Ensembl64.transcriptome.plus.genome.fasta sampleid.end2.fastq > sampleid.end2.sai
2_e2_2	sampleid.end2.sai & sampleid.end2.fastq	sampleid.end2.sam	/pyPRADA_v1.0beta/tools/bwa-0.5.7-mh/bwa samse -s -n 100 $redir/Ensembl64.transcriptome.plus.genome.fasta sampleid.end2.sai sampleid.end2.fastq > sampleid.end2.sam
			rm -f sampleid.end2.sai
			rm -f sampleid.end2.fastq
2 e2 3	sampleid.end2.sam	sampleid.end2.bam	/pyPRADA_v1.0beta/tools/samtools-0.1.16/samtools view -bS -o sampleid.end2.bam sampleid.end2.sam
			rm -f sampleid.end2.sam
2_e2_4	sampleid.end2.bam	sampleid.end2.sorted.bam	/pyPRADA_v1.0beta/tools/samtools-0.1.16/samtools sort -n -m 1000000000 sampleid.end2.bam sampleid.end2.sorted
			rm -f sampleid.end2.bam
3_e1_1	sampleid.end1.sorted.bam	sampleid.end1.remapped.bam	java -Djava.io.tmpdir=tmp/ -cp /pyPRADA_v1.0beta/tools/GATK//RemapAlignments.jar -Xmx8g org.broadinstitute.cga.tools.gatk.rna.RemapAlignments M=$refdir/Ensembl64.transcriptome.plus.genome.map IN=sampleid.end1.sorted.bam OUT=sampleid.end1.remapped.bam R=$refdir/Homo_sapiens_assembly19.fasta REDUCE=TRUE
			rm -f sampleid.end1.sorted.bam
			rm -f sampleid.end1.remapped.bam
3 e1 2	sampleid.end1.remapped.bam	sampleid.end1.remapped.sorted.bam	/pyPRADA_v1.0beta/tools/samtools-0.1.16/samtools sort -n -m 1000000000 sampleid.end1.remapped.bam sampleid.end1.remapped.sorted
3_e2_1	sampleid.end2.sorted.bam	sampleid.end2.remapped.bam	java -Djava.io.tmpdir=tmp/ -cp /pyPRADA_v1.0beta/tools/GATK//RemapAlignments.jar -Xmx8g org.broadinstitute.cga.tools.gatk.rna.RemapAlignments M=$refdir/Ensembl64.transcriptome.plus.genome.map IN=sampleid.end2.sorted.bam OUT=sampleid.end2.remapped.bam R=$refdir/Homo_sapiens_assembly19.fasta REDUCE=TRUE
3 e2 2	sampleid.end2.remapped.bam	sampleid.end2.remapped.sorted.bam	/pyPRADA_v1.0beta/tools/samtools-0.1.16/samtools sort -n -m 1000000000 sampleid.end2.remapped.bam sampleid.end2.remapped.sorted
			rm -f sampleid.end2.sorted.bam
			rm -f sampleid.end2.remapped.bam
4 1	sampleid.end1.remapped.sorted.bam & sampleid.end2.remapped.sorted.bam	sampleid.paired.bam	java -Djava.io.tmpdir=tmp/ -Xmx8g -jar /pyPRADA_v1.0beta/tools/GATK//PairMaker.jar IN1=sampleid.end1.remapped.sorted.bam IN2=sampleid.end2.remapped.sorted.bam OUTPUT=sampleid.paired.bam TMP DIR=tmp/
			rm -f sampleid.end1.remapped.sorted.bam
			rm -f sampleid.end2.remapped.sorted.bam
4_2	sampleid.paired.bam	sampleid.paired.sorted.bam	/pyPRADA_v1.0beta/tools/samtools-0.1.16/samtools sort -m 1000000000 sampleid.paired.bam sampleid.paired.sorted
			rm -f sampleid.paired.bam
5	sampleid.paired.sorted.bam	sampleid.withRG.paired.sorted.bam	java -Xmx8g -jar /pyPRADA_v1.0beta/tools/Picard//AddOrReplaceReadGroups.jar I=sampleid.paired.sorted.bam O=sampleid.withRG.paired.sorted.bam RGLB=sampleid RGPL=illumina RGPU=sampleid RGSM=sampleid
			rm -f sampleid.paired.sorted.bam
6_1	sampleid.withRG.paired.sorted.bam	sampleid.withRG.paired.sorted.bam.bai	/pyPRADA_v1.0beta/tools/samtools-0.1.16/samtools index sampleid.withRG.paired.sorted.bam
	sampleid.withRG.paired.sorted.bam	sampleid.orig.csv	java -Xmx8g -jar /pyPRADA_v1.0beta/tools/GATK//GenomeAnalysisTK.jar -l INFO -R $refdir/Homo_sapiens_assembly19.fasta --default_platform illumina --knownSites $refdir/dbsnp_135.b37.vcf -I sampleid.withRG.paired.sorted.bam --downsample_to_coverage 10000 -T CountCovariates -cov ReadGroupCovariate -cov QualityScoreCovariate -cov CycleCovariate -cov DinucCovariate -nt 12 -recalFile sampleid.orig.csv
6 2	sampleid.withRG.paired.sorted.bam & sampleid.orig.csv	sampleid.withRG.GATKRecalibrated.bam	java -Xmx8g -jar /pyPRADA_v1.0beta/tools/GATK//GenomeAnalysisTK.jar -l INFO -R $refdir/Homo_sapiens_assembly19.fasta --default_platform illumina -I sampleid.withRG.paired.sorted.bam -T TableRecalibration --out sampleid.withRG.GATKRecalibrated.bam -recalFile sampleid.orig.csv
			rm -f sampleid.withRG.paired.sorted.bam
7	sampleid.withRG.GATKRecalibrated.bam	sampleid.withRG.GATKRecalibrated.flagged.bam	java -Xmx8g -jar /pyPRADA_v1.0beta/tools/Picard//MarkDuplicates.jar I=sampleid.withRG.GATKRecalibrated.bam O=sampleid.withRG.GATKRecalibrated.flagged.bam METRICS_FILE=sampleid.Duplicates_metrics.txt VALIDATION_STRINGENCY=SILENT TMP DIR=tmp/
	sampleid.withRG.GATKRecalibrated.flagged.bam	sampleid.withRG.GATKRecalibrated.flagged.bam.bai	/pyPRADA_v1.0beta/tools/samtools-0.1.16/samtools index sampleid.withRG.GATKRecalibrated.flagged.bam
			rm -f sampleid.withRG.GATKRecalibrated.bam
8	sampleid.withRG.GATKRecalibrated.flagged.bam	sampleid/	java -Xmx8g -jar /pyPRADA_v1.0beta/tools/RNA-SeQC_v1.1.7.jar -ttype 2 -t $refdir/Homo_sapiens.GRCh37.64.gtf -r $refdir/Homo_sapiens_assembly19.fasta -s 'sampleid|sampleid.withRG.GATKRecalibrated.flagged.bam|Disc' -o sampleid/