Consiglio Nazionale delle Ricerche

ISTITUTO ITALIANO DI TECNOLOGIA

elixir ITALY

NextGenIT
Consolidation of the Italian Infrastructure for Omics Data and Bioinformatics

MUR
Centro Nazionale di Ricerca Sviluppo di terapia genica e farmaci con tecnologia a RNA

ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

TRAINING COURSE IN Computational Methods for Epitranscriptomics

Bari, 26th-28th April 2023

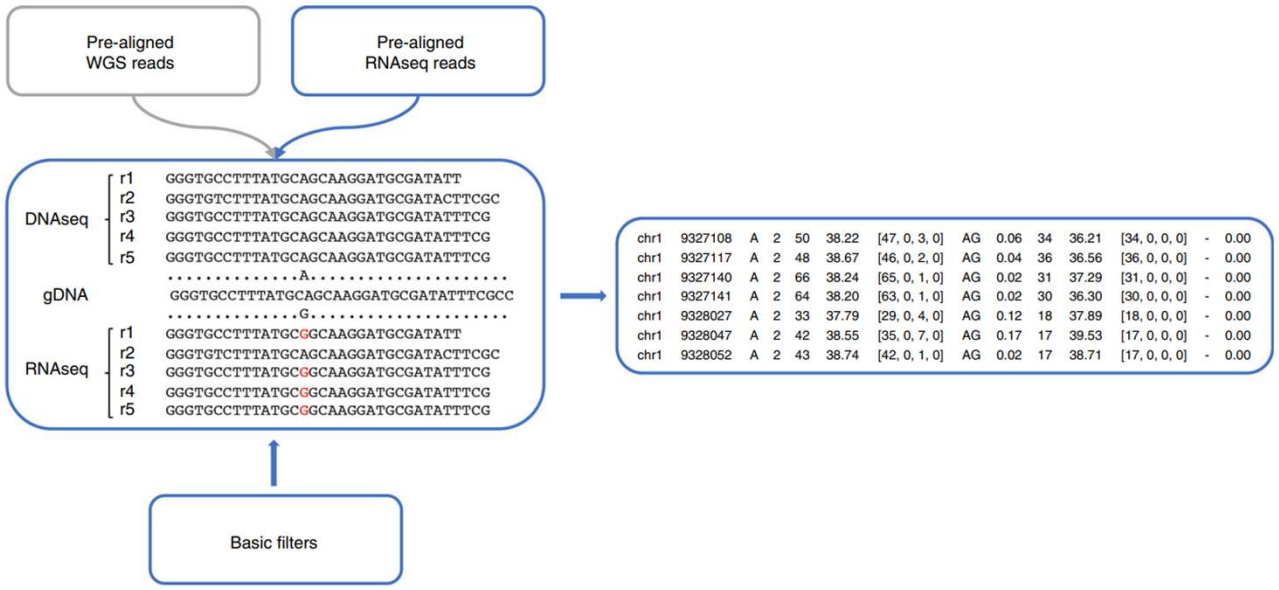
RNA editing detection by REDItools: Theoretical and practical Introduction to the REDItools package

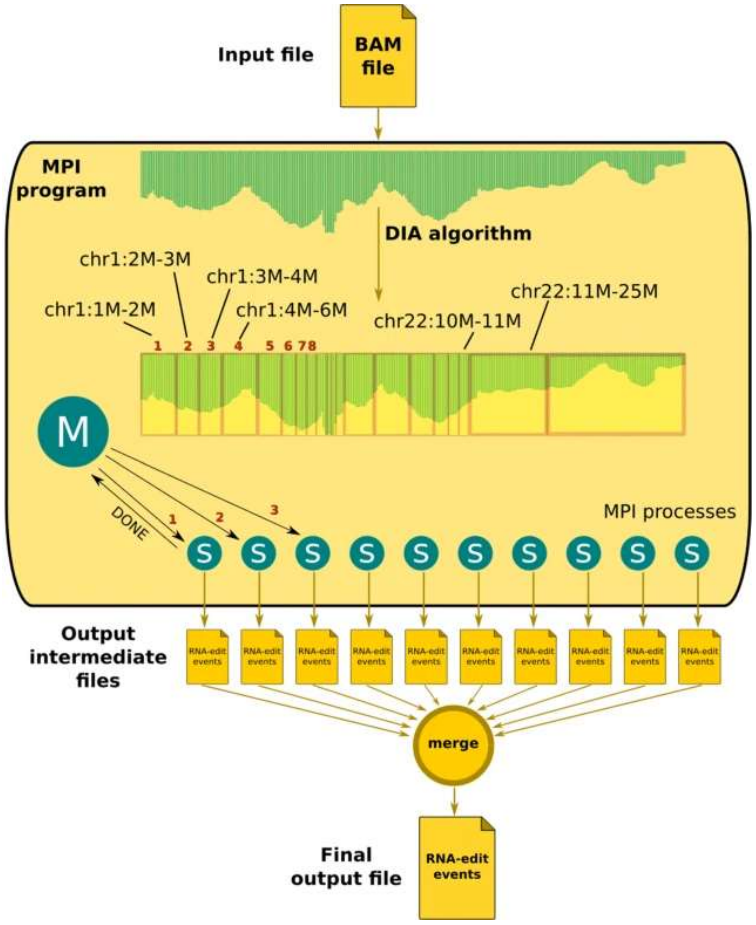
Thursday 27th April - 17.00-18.00 (day 2) Bari

Adriano **Fonzino**, Ph.D.

REDItools v.1 and REDItools v.2

REDItools are python scripts developed with the aim to study RNA editing at genomic scale by next generation sequencing data. These are conceived to facilitate the investigation of RNA editing at large-scale. They can handle reads from whatever platform in the standard **BAM** format and implement a variety of filters starting from **RNAseq** and **WGS/WES** experiments. To date, there are two versions of **REDItools**, the **v1.x** which can be used on a standard local machine and the **v2.x** which implements an HPC-aware version of the algorithm speeding up the computation speed.





Picardi E., Pesole G., 2013; Flati T. et al., 2020, Lo Giudice C. et al, 2022

REDItools v.1

As we said, **REDItools** have several scripts useful for the facilitation of RNA editing detection. They are divided into two principal folders, the **"main"** and the **"accessory"**. The former contains three **"main"** scripts that are the pivotal ones, used for the most common use cases, while the latter contains several **auxiliary** python scripts for the **downstream analysis** and for **pre-** and **post-processing** of needed data.

The three main scripts can be accessed from the folder `/path/to/REDItools/main` and these are:

- **`main/REDItoolDnaRna.py:`** the main script devoted to the identification of RNA editing events considering the combined information from RNA-Seq and DNA-Seq (WGS or WES) data in BAM format. To look at potential RNA editing candidates, RNA-Seq data alone can also be used (like Denovo script).

- **`main/REDItoolDenovo.py:`** it been conceived to predict potential RNA editing events using RNA-Seq data alone and without any a priori knowledge about genome information and biological properties of RNA editing phenomenon.

- **`main/REDItoolKnown.py:`** it has been developed to explore the RNA editing potential of RNA-Seq data sets using known editing events. Such events can be downloaded from REDIportal database or generated from supplementary materials of a variety of publications. Known RNA editing events have to be stored in TAB files.

REDItools v.1 accessory scripts

The **REDItools** package comprises **auxiliary Python scripts** to facilitate the manipulation of output and input tables.

These can be found in /path/to/REDItools/accessory and are:

- **`AnnotateTable.py:`** annotates positions of REDItool output tables;
  Input: REDItool table and indexed annotations in gtf format;
  Output: REDItool table with extra columns including annotations;

- **`FilterTable.py:`** filters in or out positions from a REDItool output table using tabular file of given positions;
  Input: REDItool table and a list of positions to filter in gtf format;
  Output: filtered REDItool table;

- **`selectPositions.py:`** filters out positions from REDItool tables basing on given criteria;
  Input: REDItool table and specific parameters to filter sites;
  Output: filtered REDItool table;

Create Working Environment

Launch the following commands:

```
conda create -n reditools_school python=2.7
conda activate reditools_school
conda install -c bioconda fisher        # (v0.1.4) required for REDItoolDenovo.py script
conda install -c bioconda pysam         # (v0.20)
conda install -c bioconda samtools      # (v1.6)
conda install -c bioconda tabix         # required if not installed with samtools
pip install scipy                       # (v1.2.3)
pip install pandas                      # (0.24.2)
```

Download from github Reditools v.1:

```
git clone https://github.com/BioinfoUNIBA/REDItools
```

Let's try whether the environment is working with the following commands:

```
python –c "import pysam, pandas, scipy, fisher"
tabix
bgzip
```

Training Dataset

The **training dataset** can be accessed to path:

/data/data_reditools/Epitranscriptome_course_2023

We have a total of 6 starting **BAM files** that we will use in these sessions. These were extracted from originals starting BAM files aligned from GTEx reads of 3 **brain** and 3 **artery** samples.
These sorted and index **BAM files** can be found at the following path:

```
/data/data_reditools/Epitranscriptome_course_2023/artery
     |--SRR1083076_chrs_4gria2_14_19.bam
     |--SRR1091254_chrs_4gria2_14_19.bam
     |--SRR1368668_chrs_4gria2_14_19.bam

/data/data_reditools/Epitranscriptome_course_2023/brain
     |--SRR1086680_chrs_4gria2_14_19.bam
     |--SRR1311771_chrs_4gria2_14_19.bam
     |--SRR1319672_chrs_4gria2_14_19.bam
```

The **REDItools package** is already installed and can be accessed to path:

```
/data/data_reditools/Epitranscriptome_course_2023/src/REDItools
```

Reference files and **annotations** are accessible in the directory:

```
/data/data_reditools/Epitranscriptome_course_2023/refs
          |--GRCh37.primary_assembly.genome_chrs_4_14_19.fa
          |--GRCh37.primary_assembly.genome_chrs_4_14_19.fa.fai
          |--refGene_sorted.gtf.gz
          |--snp151Common.sorted.gtf.gz
          |--rmsk_sorted.gtf.gz
          |--hg19_RefSeq.bed (bed format is used for the infer_experiments.py script)
```

REDItools DNA-RNA script: strandness

First, we must know the strandness of our RNAseq experiments: if we don't possess this information, we can use **infer_experiments.py** from the **RSeQC** package to infer the strandness starting only from the BAM file.
In general, there are three types of library preps (two directional and one unstranded):

- **unstranded** = the information related the origin of the read is lost.
- **second-strand** = directional, where the first read of the read pair (or in case of single end reads, the only read) is from the transcript strand
- **first-strand** = directional, where the first read (or the only read in case of SE) is from the opposite strand.

Unstranded in our case (GTEx samples). For further information please see https://chipster.csc.fi/manual/library-type-summary.html

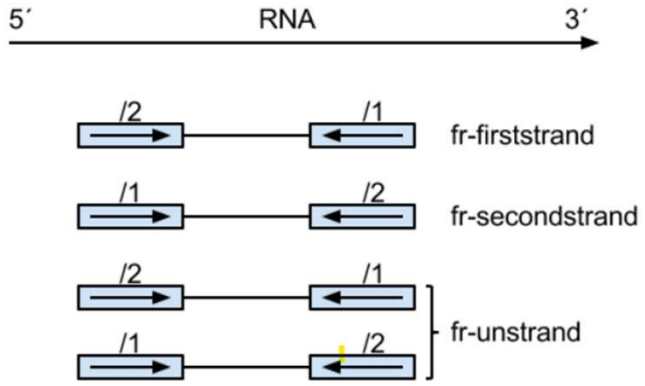
```
#!/bin/bash

RNA=$1
TISSUE=$2

BASEDIR=/home/instructor_1/data_reditools/Epitranscriptome_course_2023

BAM=$BASEDIR/$TISSUE/${RNA}_chrs_4gria2_14_19.bam
REFSEQ=/home/instructor_1/data_reditools/Epitranscriptome_course_2023/refs/hg19_RefSeq.bed

# launch command
infer_experiment.py -i $BAM -r $REFSEQ
```




```
This is PairEnd Data
Fraction of reads failed to determine: 0.0210
Fraction of reads explained by "1++,1--,2+-,2-+": 0.4849
Fraction of reads explained by "1+-,1-+,2++,2--": 0.4942
```

REDItools DNA-RNA script: Available Options (1)

REDItoolDnaRna.py

Options: (these are case-sensitive)

- **`-i`** RNA-Seq BAM file
- **`-j`** DNA-Seq BAM files separated by comma or folder containing BAM files. Note that each chromosome/region must be present in a single BAM file only.
- **`-I`** Sort input RNA-Seq BAM file
- **`-J`** Sort input DNA-Seq BAM file
- **`-f`** Reference file in fasta format. Note that chromosome/region names in the reference must match chromosome/region names in BAMs files.
- **`-C`** Base interval to explore [100000]. It indicates how many bases have to be loaded during the run.
- **`-k`** List of chromosomes to skip separated by comma or file (each line must contain a chromosome/region name).
- **`-t`** Number of threads [1]. It indicates how many processes should be launched. Each process will work on an individual chromosome/region.
- **`-o`** Output folder [rediFolder_XXXX] in which all results will be stored. XXXX is a random number generated at each run.
- **`-F`** Internal folder name [null] is the main folder containing output tables.

REDItools DNA-RNA script: Available Options (2)

REDItoolDnaRna.py

Options:

- **-c** Minimum read coverage (dna,rna) [10,10]
- **-q** Minimum quality score (dna,rna) [25,25]
- **-m** Minimum mapping quality score (dna,rna) [25,25]
- **-O** Minimum homoplymeric length (dna,rna) [5,5]
- **-s** Infer strand (for strand oriented reads) [1]. It indicates which read is in line with RNA. Available values are: 1:read1 as RNA,read2 not as RNA; 2:read1 not as RNA,read2 as RNA; 12:read1 as RNA,read2 as RNA; 0:read1 not as RNA,read2 not as RNA.
- **-g** Strand inference type 1:maxValue 2:useConfidence [1]; maxValue: the most prominent strand count will be used; useConfidence: strand is assigned if over a prefixed frequency confidence (-x option)
- **-x** Strand confidence [0.70]
- **-S** Strand correction. Once the strand has been inferred, only bases according to this strand will be selected.
- **-G** Infer strand by GFF annotation (must be GFF and sorted, otherwise use -X). Sorting requires grep and sort unix executables.
- **-K** GFF File with positions to exclude (must be GFF and sorted, otherwise use -X). Sorting requires grep and sort unix executables.

REDItools DNA-RNA script: Available Options (3)

REDItoolDnaRna.py

Options:

- **-T** Work only on given GFF positions (must be GFF and sorted, otherwise use -X). Sorting requires grep and sort unix executables.
- **-X** Sort annotation files. It requires grep and sort unix executables.
- **-e** Exclude multi hits in RNA-Seq
- **-E** Exclude multi hits in DNA-Seq
- **-d** Exclude duplicates in RNA-Seq
- **-D** Exclude duplicates in DNA-Seq
- **-p** Use paired concordant reads only in RNA-Seq
- **-P** Use paired concordant reads only in DNA-Seq
- **-u** Consider mapping quality in RNA-Seq
- **-U** Consider mapping quality in DNA-Seq
- **-a** Trim x bases up and y bases down per read [0-0] in RNA-Seq
- **-A** Trim x bases up and y bases down per read [0-0] in DNA-Seq

REDItools DNA-RNA script: Available Options (4)

REDItoolDnaRna.py

Options:

-l	Remove substitutions in homopolymeric regions in RNA-Seq
-L	Remove substitutions in homopolymeric regions in DNA-Seq
-v	Minimum number of reads supporting the variation [3] for RNA-Seq
-n	Minimum editing frequency [0.1] for RNA-Seq
-N	Minimum variation frequency [0.1] for DNA-Seq
-z	Exclude positions with multiple changes in RNA-Seq
-Z	Exclude positions with multiple changes in DNA-Seq
-W	Select RNA-Seq positions with defined changes (separated by comma ex: AG,TC) [default all]
-R	Exclude invariant RNA-Seq positions
-V	Exclude sites not supported by DNA-Seq
-w	File containing splice sites annotations
-r	Num. of bases near splice sites to explore [4]
--gzip	Gzip output files
-h	Print the help

For more information, please see the official github repository of REDItools v.1 at https://github.com/BioinfoUNIBA/REDItools/blob/master/README_1.md

1) Now launch **REDItoolsDnaRna.py** on the sample **SRR1319672** (no WGS for time constraints) redirecting the **output folder** to your area and using 4 thread per/sample. The software must select:
   - a minimum coverage of 1 read for both DNA and RNA,
   - a minimum mapping quality of 30 and 255 (suggested for STAR aligner) for DNA and RNA respectively,
   - select only sites with variations supported by at least 1 substitution,
   - consider only reads with a minimum quality score of 30 for both DNA and RNA,
   - try to exclude multimapping reads,
   - don't use any threshold for the selection of sites basing on variation frequency, for both DNA and RNA,
   - try to remove substitutions in homopolymer regions in RNA-Seq,
   - consider only properly paired reads.

2) Check the output…

3) Launch a second time but retrieve back directly only sites with variations in agree with filters applied before (exclude invariants ones).

4) Compare the output…

REDItools DNA-RNA script

Practical Session

```
#!/bin/bash
# define inputs
RNA=$1
TISSUE=$2

# genome from gencode
GENOME=/home/instructor_1/data_reditools/Epitranscriptome_course_2023/refs/GRCh37.primary_assembly.genome_chrs_4_14_19.fa
# define outputs
BASEDIR=/home/instructor_1/data_reditools/Epitranscriptome_course_2023

INPUTDIR=$BASEDIR/$TISSUE
OUTPUTDIR=$BASEDIR/$TISSUE/${RNA}_filt_inv   # change to redirect output to your area
BAM=$INPUTDIR/${RNA}_chrs_4gria2_14_19.bam

echo ""
echo "#############################################################################################################"
echo Running REDItoolDnaRna.py on $RNA from tissue $TISSUE against genome $GENOME

# -j for DNA seq data or other experiments
python /home/instructor_1/data_reditools/Epitranscriptome_course_2023/src/REDItools/main/REDItoolDnaRna.py \
    -o $OUTPUTDIR \
    -i $BAM \
    -f $GENOME \
    -t 4 \
    -c 1,1 \
    -m 30,255 \
    -v 1 \
    -q 30,30 \
    -e \
    -n 0.0 \
    -N 0.0 \
    -u \
    -l \
    -p \
    -R
```

REDItools DNA-RNA script

Example output table with RNA/WGS-WES

Reference				RNAseq					WGS/WES				
chr1	13312	C	2	1	30.00	[0, 1, 0, 0]	-	0.00	48	44.90	[0, 48, 0, 0]	-	0.00
chr1	13313	T	2	1	31.00	[0, 0, 0, 1]	-	0.00	48	43.83	[0, 0, 0, 48]	-	0.00
chr1	13314	G	2	1	31.00	[0, 0, 1, 0]	-	0.00	45	45.00	[0, 0, 45, 0]	-	0.00
chr1	13315	G	2	1	33.00	[0, 0, 1, 0]	-	0.00	46	44.07	[0, 0, 46, 0]	-	0.00
chr1	13317	T	2	1	35.00	[0, 0, 0, 1]	-	0.00	45	44.91	[0, 0, 0, 45]	-	0.00
chr1	13318	C	2	1	30.00	[0, 1, 0, 0]	-	0.00	42	44.76	[0, 42, 0, 0]	-	0.00
chr1	13319	T	2	1	33.00	[0, 0, 0, 1]	-	0.00	44	45.20	[0, 0, 0, 44]	-	0.00
chr1	13320	G	2	1	30.00	[0, 0, 1, 0]	-	0.00	44	43.48	[0, 0, 44, 0]	-	0.00
chr1	13321	A	2	1	30.00	[1, 0, 0, 0]	-	0.00	41	45.10	[41, 0, 0, 0]	-	0.00
chr1	13322	G	2	1	34.00	[0, 0, 1, 0]	-	0.00	43	43.58	[0, 0, 43, 0]	-	0.00

Column names

Reference: Region; Position; Reference; Strand (0 → -; 1 → +; 2 → unknown);
RNAseq: Coverage-q30; MeanQ; BaseCount[A,C,G,T]; AllSubs; Frequency;
WGS/WES: gCoverage-q30; gMeanQ; gBaseCount[A,C,G,T]; gAllSubs; gFrequency.

Prerequisites for the use of AnnotateTable.py accessory script (GTF files)

```
chr1    hg19_refseq     exon    11869   12227   .       +       .       gene_id "DDX11L17"; transcript_id "NR_148357"; exon_number "1";
exon_id "NR_148357.1"; gene_name "DDX11L17";
chr1    hg19_refseq     transcript      11869   14362   .       +       .       gene_id "DDX11L17"; transcript_id "NR_148357";  gene_name
"DDX11L17";
chr1    hg19_refseq     exon    11874   12227   .       +       .       gene_id "DDX11L1"; transcript_id "NR_046018"; exon_number "1";
exon_id "NR_046018.1"; gene_name "DDX11L1";
chr1    hg19_refseq     transcript      11874   14409   .       +       .       gene_id "DDX11L1"; transcript_id "NR_046018";  gene_name
"DDX11L1";
chr1    hg19_refseq     exon    12613   12721   .       +       .       gene_id "DDX11L17"; transcript_id "NR_148357"; exon_number "2";
exon_id "NR_148357.2"; gene_name "DDX11L17";
chr1    hg19_refseq     exon    12613   12721   .       +       .       gene_id "DDX11L1"; transcript_id "NR_046018"; exon_number "2";
exon_id "NR_046018.2"; gene_name "DDX11L1";
chr1    hg19_refseq     exon    13221   14362   .       +       .       gene_id "DDX11L17"; transcript_id "NR_148357"; exon_number "3";
exon_id "NR_148357.3"; gene_name "DDX11L17";
chr1    hg19_refseq     exon    13221   14409   .       +       .       gene_id "DDX11L1"; transcript_id "NR_046018"; exon_number "3";
exon_id "NR_046018.3"; gene_name "DDX11L1";
chr1    hg19_refseq     exon    14362   14829   .       -       .       gene_id "WASH7P"; transcript_id "NR_024540"; exon_number "11";
exon_id "NR_024540.11"; gene_name "WASH7P";
chr1    hg19_refseq     transcript      14362   29370   .       -       .       gene_id "WASH7P"; transcript_id "NR_024540";  gene_name
"WASH7P";
```

Fields: seqname, source, feature, start (1-based), end (1-based), score, strand, frame, attribute

Prerequisites for the use of AnnotateTable.py accessory script

Practical Session

```
# please execute in your area
# Download required annotations tables from UCSC (RepeatMask, Refseq, SNPs are already available into the refs folder of your environment)
curl http://hgdownload.cse.ucsc.edu/goldenpath/hg19/database/rmsk.txt.gz > rmsk.txt.gz
curl http://hgdownload.cse.ucsc.edu/goldenpath/hg19/database/refGene.txt.gz > refGene.txt.gz

# Decompress gz tables
gunzip rmsk.txt.gz
gunzip refGene.txt.gz

# Covert to GTF files RepeatMask via awk
gawk 'OFS="\t"{print $6,"rmsk_hg19",$12,$7+1,$8,".",$10,".","gene_id \""$11"\"; transcript_id \""$13"\";"}' rmsk.txt > rmsk.gtf

# For Refseq genes annotation table there's the need for the UCSC genePredToGtf utility and convert it to GTF
curl http://hgdownload.soe.ucsc.edu/admin/exe/linux.x86_64/genePredToGtf > genePredToGtf
chmod u+x $BASEDIR/genePredToGtf
cut -f 2- refGene.txt | $BASEDIR/genePredToGtf -utr -source=hg19_refseq file stdin refGene.gtf

# Sorting all GTF converted annotation tables
sort -k1,1 -k4,4n rmsk.gtf > rmsk_sorted.gtf
sort -k1,1 -k4,4n refGene.gtf > refGene_sorted.gtf

# Compress with bgzip sorted GTF files and index via Tabix software (used internally by accessory scripts to access positions via pysam.Tabix Python wrapper)
bgzip rmsk_sorted.gtf
bgzip refGene_sorted.gtf
tabix -p gff rmsk_sorted.gtf.gz
tabix -p gff refGene_sorted.gtf.gz

# Proceed with the AnnotateTable.py accessory script for rmsk and then for Refgene GTF tables.
```

Annotate a table with AnnotateTable.py accessory script

Practical Session

Annotate the **SRR1319672 output table** produced **without invariant positions** using AnnotateTable.py accessory script:

1) Annotate using **rmsk** sorted an indexed **GTF annotations** (ALU, SINE, LINE, ect.) and save it into your personal area within the same folder of the input table but as outTable_***_*rmsk*

2) Annotate using **Refgene/refseq** sorted and indexed **GTF annotations** (only gene name) but starting from the ***_rmsk annotated table to add the only the gene name column. Save it in the same folder as outTable_***_*rmsk_RefGene.*

```
python AnnotateTable.py -h
USAGE: python AnnotateTable.py [options]
Options:
     -a              Sorted Annotation file
     -i              Annotate a file of positions [column1=region, column2=coordinate (1 based)]
                     or a single position [region:coordinate (1 based)]
     -k              skip lines starting with: #
     -r              Add a prefix to chromosome name [] (chr when the name is a number)
     -s              Strand column in annotation file [4]
     -u              Not use table strand info (fix it to 2)
     -c              Add columns separated by comma (feature:1, gene_id:2, transcript_id:3) [1,2]
     -n              Column name [Col]
     -S              Correct strand by annotation
     -C              Columns with base distribution [7,12] (in combination with -S)
     -o              Save lines to a file
     -h              Print this help
```

Annotate a table with AnnotateTable.py accessory script

```
#!/bin/bash
BASEDIR=/home/instructor_1/data_reditools/Epitranscriptome_course_2023

# reditable with SelPos in our case
reditable=$1
RepeatMask=$BASEDIR/refs/rmsk_sorted.gtf.gz
RefGene=$BASEDIR/refs/refGene_sorted.gtf.gz

# launch AnnotateTable.py for RepeatMask
# it will automatically produce a tabix gz version of the starting GTF annotation file with index file
echo "Launching AnnotateTable.py for RepeatMask annotations"
python /home/instructor_1/data_reditools/Epitranscriptome_course_2023/src/REDItools/accessory/AnnotateTable.py \
    -i $reditable \
    -a $RepeatMask \
    -u \
    -c1,2,3 \
    -n rmsk \
    -o ${reditable}_rmsk

# launch AnnotateTable.py for Refseq gene annotations
# it will automatically produce a tabix gz version of the starting GTF annotation file with index file
# instead of -u use -S if you want to correct strand by annotation
echo "Launching AnnotateTable.py for Refgene annotations"
python /home/instructor_1/data_reditools/Epitranscriptome_course_2023/src/REDItools/accessory/AnnotateTable.py \
    -i ${reditable}_rmsk \
    -a $RefGene \
    -u \
    -c 2 \
    -n RefGene \
    -o ${reditable}_rmsk_Refgene

echo Computations finished.
```

Annotate a table with AnnotateTable.py accessory script: Output

chr4	158272377	A	2	1	40.00	[0, 0, 1, 0]	AG	1.00	-	-	-	-	-	LINE	L3	CR1-LINE	GRIA2
chr4	158274707	A	2	1	39.00	[0, 0, 1, 0]	AG	1.00	-	-	-	-	-	SINE	AluY	Alu-SINE	GRIA2
chr4	158275437	A	2	1	41.00	[0, 0, 1, 0]	AG	1.00	-	-	-	-	-	-	-	-	GRIA2
chr4	158281158	A	2	121	39.68	[119, 0, 1, 1]	AT AG	0.01	-	-	-	-	-	-	-	-	GRIA2
chr4	158281293	A	2	84	39.08	[78, 0, 6, 0]	AG	0.07	-	-	-	-	-	-	-	-	GRIA2
chr4	158281294	A	2	77	39.04	[16, 1, 60, 0]	AG AC	0.79	-	-	-	-	-	-	-	-	GRIA2

Launching de-novo search for putative editing sites candidates

ReditoolsDenovo Practical Session

REDItoolDenovo.py has been conceived to **predict potential RNA editing events** using RNA-Seq data alone and without any a priori knowledge about genome information and biological properties of RNA editing phenomenon.

REDItoolDenovo.py includes one additional column concerning the **reliability** of editing prediction:
Pvalue: is the **pvalue** per site calculated according to Fisher exact test. It indicates how much the observed base distribution for a change is different from the expected, calculated by the empirical base substitution for the entire RNA-Seq experiment.
See github repo for further information.

Exercise:

Run on the Brain sample SRR1319672 and retrieve only sites with pvalues < 0.001. Is GRIA2 recoding site at coordinates chr4:158281294 still there?

Launching de-novo search for putative editing sites candidates

ReditoolsDenovo

```
#!/bin/bash
# define inputs
# genome from gencode
GENOME=/home/instructor_1/data_reditools/Epitranscriptome_course_2023/refs/GRCh37.primary_assembly.genome_chrs_4_14_19.fa
# define outputs
BASEDIR=/home/instructor_1/data_reditools/Epitranscriptome_course_2023
RNA=$1
TISSUE=$2
INPUTDIR=$BASEDIR/$TISSUE
OUTPUTDIR=$BASEDIR/$TISSUE/${RNA}_Denovo   # change to redirect output to your working area
BAM=$INPUTDIR/${RNA}_chrs_4gria2_14_19.bam
echo ""
echo "#############################################################################################################"
echo Running REDItoolDenovo.py on:
echo Sample: $RNA
echo Tissue: $TISSUE
echo Genome: $GENOME
echo Bam file: $BAM

# launch REDItoolDenovo
python /home/instructor_1/data_reditools/Epitranscriptome_course_2023/src/REDItools/main/REDItoolDenovo.py \
    -i $BAM \
    -o $OUTPUTDIR \
    -f $GENOME \
    -t 3 \
    -c 5 \
    -m 255 \
    -v 1 \
    -q 30 \
    -e \
    -n 0.0 \
    -u
```

Launch REDItools on well known editing positions

REDItoolKnown.py Practical Session

1) Retrieve TAB file from rediportal of gria2 exonic editing sites

2) Compress and index it via Tabix

3) Run REDItoolKnown.py on [SRR1319672 brain] and [SRR1091254 artery] to speed up search limiting REDItool search only on well defined known positions.

Launch REDItools on well known editing positions

REDItoolKnown.py

```
#!/bin/bash
# define inputs
# genome from gencode
GENOME=/home/instructor_1/data_reditools/Epitranscriptome_course_2023/refs/GRCh37.primary_assembly.genome_chrs_4_14_19.fa
# define outputs
BASEDIR=/home/instructor_1/data_reditools/Epitranscriptome_course_2023
RNA=$1
TISSUE=$2
INPUTDIR=$BASEDIR/$TISSUE
OUTPUTDIR=$BASEDIR/$TISSUE/${RNA}_known   # change to redirect output to your working area
BAM=$INPUTDIR/${RNA}_chrs_4gria2_14_19.bam
TAB=/home/instructor_1/data_reditools/Epitranscriptome_course_2023/src_reditoolsKnown/AtlasTable_gria2_exonic.txt.gz
echo ""
echo "#############################################################################################################"
echo Running REDItoolKnown-py on:
echo Sample: $RNA
echo Tissue: $TISSUE
echo Genome: $GENOME
echo Tab file: $TAB
echo Bam file: $BAM

# launch REDItoolKnown
python /home/instructor_1/data_reditools/Epitranscriptome_course_2023/src/REDItools/main/REDItoolKnown.py \
    -i $BAM \
    -o $OUTPUTDIR \
    -f $GENOME \
    -l $TAB \
    -t 3 \
    -c 5 \
    -m 255 \
    -v 1 \
    -q 30 \
    -e \
    -n 0.1 \
    -u
```

Filter an output REDItools table according to different criteria

Output tables are very huge with the major part made by invariant positions → Proceed with the accessory script **selectPositions.py** to retain only AG,TC

```
#!/bin/bash

reditable=$1

echo Launching selectPosition.py script on REDItools output table: $reditable
# launch selectPositions.py to decrease the number of sites to annotate and to eliminate invariant
sites
python
/home/instructor_1/data_reditools/Epitranscriptome_course_2023/src/REDItools/accessory/selectPositions
.py \
    -i $reditable \
    -d -1 \
    -c 3 \
    -v 1 \
    -s AG,TC \
    -r \
    -o ${reditable}_SelPos

echo Computations finished.
echo ""
```

Annotate Table script

```
#!/bin/bash
BASEDIR=/home/instructor_1/data_reditools/Epitranscriptome_course_2023
# reditable with SelPos in our case
reditable=$1
RepeatMask=$BASEDIR/refs/rmsk_sorted.gtf.gz
RefGene=$BASEDIR/refs/refGene_sorted.gtf.gz

# launch AnnotateTable.py for RepeatMask
# it will automatically produce a tabix gz version of the starting GTF annotation file with index file
echo "Launching AnnotateTable.py for RepeatMask annotations"
python /home/instructor_1/data_reditools/Epitranscriptome_course_2023/src/REDItools/accessory/AnnotateTable.py \
    -i $reditable \
    -a $RepeatMask \
    -u \
    -c1,2,3 \
    -n rmsk \
    -o ${reditable}_rmsk

# launch AnnotateTable.py for Refseq gene annotations
# it will automatically produce a tabix gz version of the starting GTF annotation file with index file
echo "Launching AnnotateTable.py for Refgene annotations"
python /home/instructor_1/data_reditools/Epitranscriptome_course_2023/src/REDItools/accessory/AnnotateTable.py \
    -i ${reditable}_rmsk \
    -a $RefGene \
    -c 2 \
    -n RefGene \
    -S \
    -o ${reditable}_rmsk_Refgene

echo Computations finished.
```

Preliminary Filter table steps

Download and prepare dbSNP annotations for REDItools

snp151Common **GTF table** creation for **REDItools**:

```
# download from UCSC dbSNP151Common
curl http://hgdownload.cse.ucsc.edu/goldenpath/hg19/database/snp151Common.txt.gz > snp151Common.txt.gz

# decompress it
gunzip snp151Common.txt.gz

# convert to gtf via simple awk command
awk 'OFS="\t"{if ($11=="genomic" && $12=="single") print
$2,"ucsc_snp151_hg19","snp",$4,$4,".",$7,".","gene_id \""$5"\"; transcript_id \""$5"\";"}'
snp151Common.txt > snp151Common.gtf

# sort for position
sort -k1,1 -k4,4n snp151Common.gtf > snp151Common.sorted.gtf

# compress and index via tabix
bgzip snp151Common.sorted.gtf
tabix -p gff snp151Common.sorted.gtf.gz
```

TO NOTE: you don’t need here during the course to perform these passages since you have already been provided with a preprocessed **snp151Common.sorted.gtf.gz** table in order to spare time.

Filter Table from known common SNPs

Filtering in and out

```
#!/bin/bash
BASEDIR=/home/instructor_1/data_reditools/Epitranscriptome_course_2023

reditable=$1
SNP=/home/instructor_1/data_reditools/Epitranscriptome_course_2023/refs/snp151Common.sorted.gtf.gz
noSNPoutput=${reditable}_noSNPs
SNPoutput=${reditable}_SNPs

echo Performing FiterTable.py accessory script to filter known common SNPs from dbSNP v151 on REDItable $reditable
echo Retrieving noSNP filtered table from REDItools output annotated table...
# launch FilterTable.py script to retrieve noSNPs sites
python /home/instructor_1/data_reditools/Epitranscriptome_course_2023/src/REDItools/accessory/FilterTable.py \
    -i $reditable \
    -s $SNP \
    -S snp \
    -o $noSNPoutput \
    -E \
    -p

echo Retrieving only SNPs table from REDItools output annotated table...
# launch FilterTable.py script to retrieve SNPs sites (-f instead of -s)
python /home/instructor_1/data_reditools/Epitranscriptome_course_2023/src/REDItools/accessory/FilterTable.py \
    -i $reditable \
    -f $SNP \
    -F snp \
    -o $SNPoutput \
    -E \
    -p

echo Computations finished.
```