

## Homework 11 - BINF Programming I

Now that you have a working class hierarchy of objects representing spliced transcripts, you'll get some experience using the classes you've written to investigate biological data. In this homework, you'll write a program that finds NMD targets (see description from Project One). If you find any, you'll then investigate their expression using an RNA-Seq data set as part of Project Two.

In this homework, you'll write a program called `findNmdTargets.py` that identifies gene models whose pattern of introns and exons make them potential targets for nonsense-mediated decay.

I will add new testing code to `testProject1.py` which you should use to validate your code. Also, fix any errors you haven't already corrected from your project one code.

To start, do this:

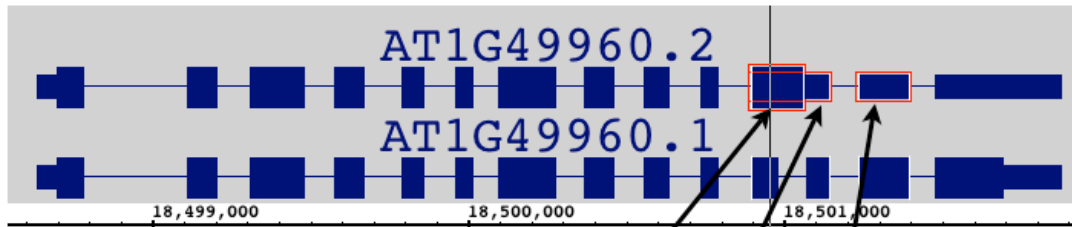
1. Create a file `findNmdTargets.py` and use the Unix `chmod` command to make it executable.
2. Add it to the repository in your `project1` directory.

Add the following methods:

3. `measureEejDistance` - accepts one argument, a `GeneModel`, and returns the number of bases in the spliced transcript separating the translation stop and the furthest (most distal) exon-exon junction (EEJ) in the downstream, 3-prime UTR. (See diagram below.) Returns `None` if there is no downstream EEJ, if the `GeneModel` does not have a translation stop or if translation start, or if the translation start and stop are the same. (Note that in TAIR9.bed, gene models that do not encode proteins have identical translation start and stop values.)
4. `reportEejDistances` - reports the number of bases in the spliced transcript separating the translation stop and the furthest (most distal) exon-exon junction (EEJ) in the downstream, 3-prime UTR (output from `measureEejDistance`) for each gene model in a given BED file or streamed from `stdin`. Accepts the following named arguments:
  - a. `bed_file` - a string, the name of a BED format file, default value `None`
  - b. `out_file` - a string, the name of a file to write, default value `None`
  - c. `distance` - an int, default value is 0. If non-zero, only print results for gene models with EEJ distance  $\geq$  `distance`. If 0, then print EEJ distance for all the gene models, or the string "NA" if a gene model has no 3-prime UTR EEJs.
  - d. **Output Format:** For each gene model, print the name of the gene model, followed by a tab character, and then the EEJ distance. Print to `out_file` or to `stdout` if `out_file` is `None`.
5. Write a "main" method that uses these methods (plus any others you may like to add) that allows me to run your program from the Unix command line, as follows:

```
cat TAIR.bed | findNmdTargets.py -N 55 > results.txt
cat TAIR.bed | findNmdTargets.py > results.txt
findNmdTargets.py -N 50 TAIR9.bed > results.txt
findNmdTargets.py TAIR9.bed > results.txt
findNmdTargets.py -N 50 TAIR9.bed results.txt
findNmdTargets.py TAIR9.bed results.txt
```

and so on... see `optdemo.py` for help in using `getopt` module to implement argument and option parsing.



**A**

translated part of  
the last coding  
region exon  
(including stop  
codon)

**B**

last coding region  
exon

**C**

second to last  
UTR exon

property	no ID	Selection Info	Search	Slic
name		AT1G49960.2	AT	60.2
id		AT1G4	AT	60.2
chromosome	chr1	chr1	chr1	60.2
start	18,500,896	18,500,896	18,501,234	
end	18,501,062	18,501,147	18,501,395	
length	166	251	161	
type		TAIR9 ALL	TAIR9 ALL	
cds min		18498699	18498699	
cds max		18501062	18501062	
forward		true	true	
score		0.0	0.0	

$$\text{EEJ distance} = 251 - 166 + 161$$

**B**

**A**

**C**