Integrating the output of two genefinders, tRNAscan-SE and Aragorn

Input: Output of tRNAscan and Aragorn on each genome saved as separate files.

TSE outputs are read in two different formats: “*.tse.out” and “*.SS.tse.out”

ARA outputs are read in one format “.ara.out”.

Script has four functions:

1. Integrate_Tse_Ara() 2. making_ara_df() 3. making_tse_df() 4. integrate() 5. formatoutput()

For each genome the fallowing pipeline will be run to make the integrated gene file.

1. making_ara_df :

This function will extract genes’ information found by aragorn as a dataframe format.

The output will be in a table with columns:

aragenename , arasourceOrg ,araidentity , aradirection ,arabegin ,araend , araac, arasourceSO, arasourceseq ,arascore ,arageneseq , arageness, araintronbegin, araintronend, aranote (will be pseudo if gene is marked as pseudo)

2. making_tse_df:

This function will extract the genes’ information found by tRNAscan in a dataframe format.

It reads each gene along with its secondary structure from two files: “*.tse.out” and “*.SS.tse.out”.

The output will be in a table with columns:

tsegenename, tsesourceOrg, tseidentity, tsedirection, tsebegin, tseend, tseac, tsesourceseq, tsescore, tsegeneseq, tsegeness, tseintronbegin, tseintronend, tseacloc, note

NOTE2: We had few anticodons for genes found by ara with four letters like “(gtag)”

The last column, **note**, will take the value pseudo if the gene is known as Pseudo-gene by tse. It can take truncated,pseudo if the gene is both pseudo and truncated. It can also be just truncated.

If gene is not found by tse, but it is found by aragorn, it will take the value “notfound” and vice versa. If the gene is not pseudo or truncated the column note will take the value “notfound”

3. integrate:

This function will take the tables made by previous two functions as input. It will switch the start and end positions found by tse, if they are on reverse strand. Using the functions “Granges” and “findOverlaps” which are implemented based on the idea of interval tree to find overlapped genes using their coordinates (sourceseq,position,strand), we found genes that are found by both gene finders (they overlap). These genes are saved in a dataframe called “overlapdf”. Using the function “setdiff” we found genes that are found by only one of the genefinders. Finally, we combine both data as one dataframe called “integrated_Tse_Ara”. At this step another column called “foundby” is added to the table which shows by which genefinder the gene is found. The value for this column can be “both”, “tse”, or “ara”.

Geneid at this step is made by SourceOrg + sourceseq + index of gene within that organism(not sourceseq!) seprated by “_”.

4. formatoutput:

This function will format the **integrated_Tse_Ara** dataframe as a fixed width format to print out. The table is printed as four files: **SecondaryS.txt, Identity.txt, Coordinate.txt, Intron.txt.** This is the summery of what each file contains:

1. Intron.txt :

"GeneId", "TseIntronBegin", "TseIntronEnd", "AraIntronLocStart","AraIntronLocEnd"

2. Coordinate.txt:

"GeneId", "SourceOrg", "SourceSeq", "SourceSO", "Direction","TseBegin","TseEnd", "AraBegin", "AraEnd","Foundby"

3. Identity.txt:

"GeneId", "TseIdentity", "AraIdentity", "TseAc", "AraAc", "TseAcLoc", "AraScore","TseScore", "TseNote", "AraNote"

4. SecondaryS.txt

geneid, tseseq, tsess, araseq, arass

At the end of this script there is a function called **summery_table().** This function is for making a dataframe table with the four rows for each geneset of:

tse2 , ARA, Union(tse2,ARA), Intersection(tse2,ARA)

It had 35 columns for T (#trnas), N (#nucleotides), N/T, length range, %G, %C, %T, %A, %intron-containing, and the 23 class frequencies including IUPAC aa codes for the 20 elongators(A, C, E, ... , Y), X for initiators, Z for selenocysteine, $ for pseudogenes, ? for sup, # for stop and O for pyl