AlternateScript\_UsingTransform

This is an alternate script to retrieve the same information in a different way

* Script Execution
* Introduction and purpose of my script
* Concept Implemented
* Validations and Exceptions
* List of Objects

Script Execution:

This is a normal simple perl script which can be executed in the command line with the below parameters,

***perl xxxxx.pl arg1 arg2 arg3***

arg1->(1-22,x,y)

arg2->(Starting\_Coordinate)

arg3->(Ending\_Coordinate)

*Example : perl xxxxx.pl 12 63840524 80294266*

Introduction and purpose of my script:

The purpose of the script is to convert or map the GRch38 co-ordinates to the older assembly (GRch37) to its corresponding region. The perl script receives the co-ordinates in the above format and finally returns the mapped co-ordinates which refers to the older assembly (GRch37)

Concept Implemented:

The script works on the base line of the transform() function which are used to convert features like Genes, Transcripts and Exons between coordinate systems In our case both the coordinate systems are same but different versions.

At first the script validates the input and places the region covered by the coordinates in a Slice. Then the Slice is used to get all genes present in a slice with a loop and then transform it to GRch37 assembly. After the transformation is done we get seq\_region\_start and seq\_region\_end of the gene, which is stored in array variables.

The same procedure is done for transcripts and exons as well and the sequence start and end values for all the genes, transcripts and exons are entirely stored in array variables. Finally the minimum value of the seq\_region\_start and the maximum value of the seq\_region\_end gives the mapped coordinates or corresponding region in GRch37.

Validations and Exceptions:

*Input Validation* : The input is validated and throws a alert message for invalid input

*Array Validation* : The arrays which store seq\_region values are also validated by if statement. If the array is empty it throws message as “Region does not contain any Gene/Transcript/Exon in GRch37 assembly”

*Exceptions* : We might get some exceptions when running the loops in GRch37 assembly. This is because the entire object itself (Gene/Transcript/exon) might be missing in the target assembly. We catch such exceptions using eval for further analysis and proceed up with the loop.

List of used objects :

*Adaptors* : Slice

*Methods* : transform(), fetch\_by\_region(), get\_all\_genes(), get\_all\_transcripts(), get\_all\_exons(), seq\_region\_start(), seq\_region\_end(), min(), max()

*Array Variables* :

@tgt\_start – Stores the seq\_region\_start() for all features (Gene/Transcripts/Exons) in GRch37

@tgt\_end - Stores the seq\_region\_end() for all features (Gene/Transcripts/Exons) in GRch37

Points Analysed:

* The same script works fine with project() as well
* If the region is really big it takes some time for processing, at these times the transcripts and exon part may be commented
* If the project() or transform() function fails in getting the seq\_region, we can replace existing functions as get\_all\_genes\_by\_souce(‘ensembl’), get\_all\_transcripts\_source(‘ensembl’), get\_all\_exons(‘ensembl’) and can give a try again