

Table of Contents

1 General Notes.....	1
1.1 “Work” Directory.....	1
1.2 Limitations.....	1
2 Process Flows.....	2
3 FastQ File Generator.....	3
3.1 Objective.....	3
3.2 Data Sources.....	3
3.3 DB Route of the Process Flow.....	4
3.4 Ensembl Route of the Process Flow.....	8
4 Bar Charts: Compare Transcripts from same Gene.....	11

1 General Notes

1.1 “Work” Directory

For ease of reference throughout this manual, it is assumed that all files, including:

- Python scripts
- Text files
- Data Base files

will be located in the same directory as each other, henceforth referred to as the “**Work**” directory.

1.2 Limitations

For both the “Fastq” file generation and the Bar Chart transcript comparison tools, this software is only compatible with “Positive” (Forward) Strand transcripts. The handling of “Negative” (Reverse) Strand transcripts will be included in a later version.

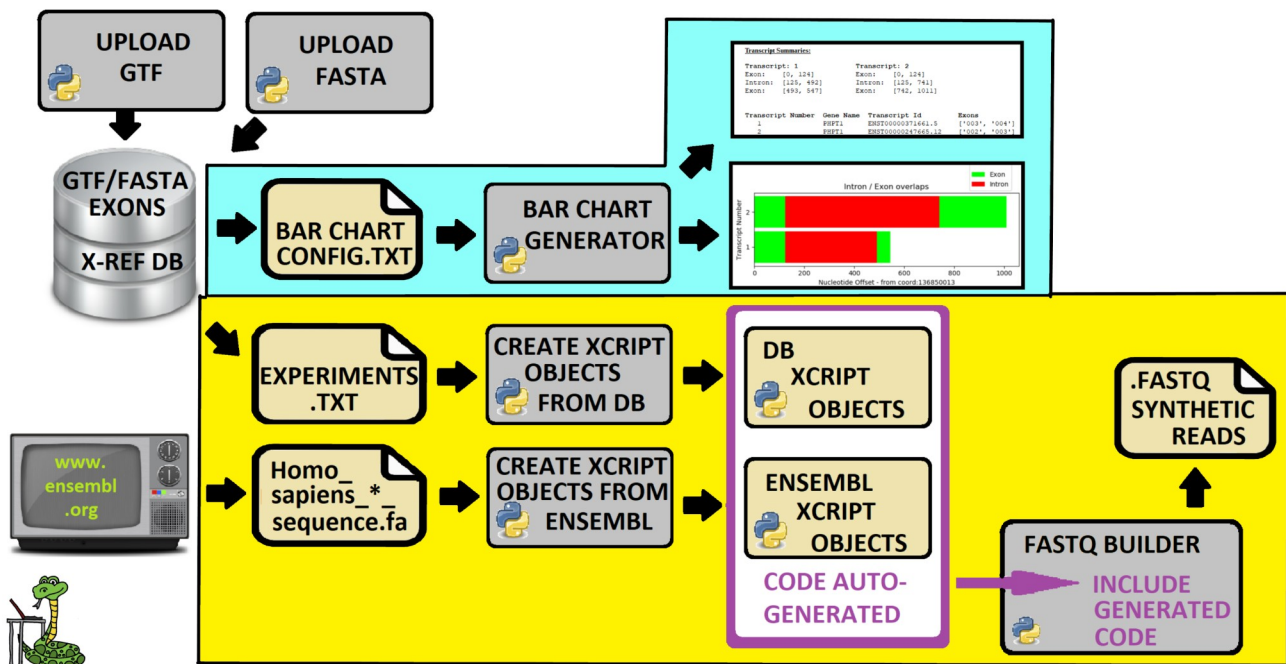
What this tool does

2 Process Flows

There are two distinct sections to the Process Flow:

- “Fastq” Synthetic RNA-Seq Reads Generator (highlighted in yellow below)
- Bar-Chart Generator (highlighted in cyan below) for comparing Gene Transcripts

Workflow of .Fastq Generator and Bar Chart Generator:

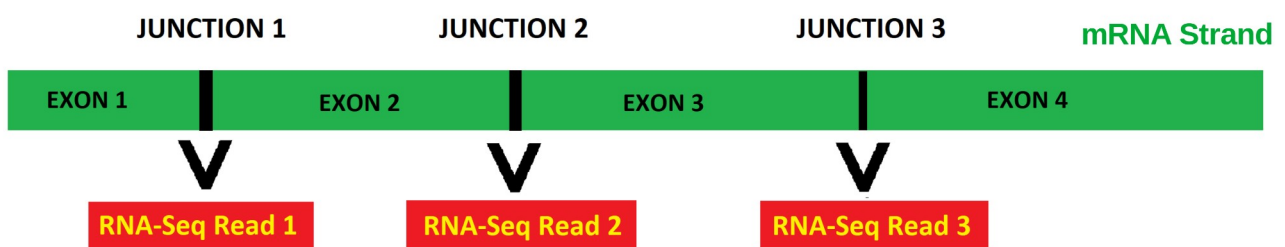


3 FastQ File Generator

3.1 Objective

The main objective of this tool is to generate .fastq files consisting of RNA-Seq/Ribo-Seq reads (henceforth referred to as “Reads”) which *span across exon-exon junctions*.

As a visual example, we might have the following (theoretical) transcript:



In the above scenario we have:

- **One** Transcript (mRNA Strand)
- **Four** Exons
- **Three** Exon-Exon Junctions

So in this case, **three** “Reads” will be generated for this transcript – one for each of the Exon-Exon Junctions.

3.2 Data Sources

The data can be sourced in two ways. The manner in which the data is sourced, informs the choice on which route the user must follow within the Process Flow.

1. **DB ROUTE:** Create a local Database from .GTF and .FASTA files (this route is recommended if testing a third party tool such as LeafCutter as it is more scalable once set up correctly)
 - Requires:

- .GTF and .FASTA file (download at: https://www.gencodegenes.org/human/release_35.html)
 - DB Browser for SQLite (download at <https://sqlitebrowser.org/>) or similar
 - Python 3 installation
 - The DB route of the Process Flow should be followed in this case (see 3.3).
2. **ENSEMBL ROUTE:** Navigate directly to the Ensembl website (easier method)
- Requires:
 - Internet access
 - Python 3 installation
 - The Ensembl route of the Process Flow should be followed in this case (see 3.4).

3.3 DB Route of the Process Flow

For this method, it is assumed that “DB Browser for SQLite” has been installed.

The following steps should be taken:

1. Download:
 - ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_35/GRCh38.primary_assembly.genome.fa.gz
 - ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_35/gencode.v35.primary_assembly.annotation.gtf.gz
 - from https://www.gencodegenes.org/human/release_35.html
2. In a Linux system (or other compatible) use the gunzip command to unzip these files
3. Place the unzipped files into the “Work” directory
4. Run “DB Browser for SQLite” and select the option “New Database”, and save it with this name *exactly*: **fasta_gtf_v35.db** – into the “Work” directory
5. Using “DB Browser for SQLite”, create the following two tables (and “commit”/write the changes as necessary):

```
CREATE TABLE "EXONS_GTF" (
  "GeneName"      TEXT NOT NULL,
  "XcriptID"      TEXT NOT NULL,
  "ExonNo"        TEXT NOT NULL,
  "CoordFrom"     TEXT NOT NULL,
  "CoordTo"       TEXT NOT NULL,
```

```

"XcriptType"      TEXT NOT NULL,
"GeneID"          TEXT NOT NULL,
"ExonID"          TEXT NOT NULL,
"XcriptName"      TEXT NOT NULL,
"Chrom"           TEXT NOT NULL,
"Strand"          TEXT NOT NULL,
PRIMARY KEY ("GeneName", "XcriptID", "ExonNo")
)

```

```

CREATE TABLE "FASTA_1000" (
  "Chrom"          TEXT NOT NULL,
  "UpperIndex"     TEXT NOT NULL,
  "Seq" TEXT NOT NULL,
  PRIMARY KEY ("Chrom", "UpperIndex")
)

```

6. Run the Python program [sql_upload_fasta.py](#)

- This program requires the file “**GRCh38.v35.primary_assembly.genome.fa**” (see the gunzip command above) to be in the “Work” directory

7. Run the Python program [sql_upload_gtf.py](#)

- This program requires the file “**gencode.v35.primary_assembly.annotation.gtf**” (see the gunzip command above) to be in the “Work” directory

8. We must now manually create a file called “**experiments.txt**”. This file will contain the transcripts which we wish to be able to use for generating “Reads” (in fastq format) and should be placed into the “Work” directory

9. In this demo, we will use the “**Protein Coding**” transcripts for the gene name “**PHPT1**”. Using the “DB Browser for SQLite” enter “PHPT1” in the “Gene Name” filter field, and “protein_coding” in the “XcriptType” field, and from the resulting displayed records, select and copy the cells as shown below (Note: CoordFrom & CoordTo are optional):

Database Structure Browse Data Edit Pragas Execute SQL											
Table: EXONS_GTF Filter in any column											
	GeneName	XcriptID	ExonNo	CoordFrom	CoordTo	XcriptType	GeneID	ExonID	XcriptName	Chrom	Strand
1	PHPT1	ENST00000247665.12	001	136849409	136849590	protein_coding	ENSG00000054148.18	ENSE0000188832...	PHPT1-201	9	+
2	PHPT1	ENST00000247665.12	002	136850013	136850137	protein_coding	ENSG00000054148.18	ENSE0000372475...	PHPT1-201	9	+
3	PHPT1	ENST00000247665.12	003	136850755	136851024	protein_coding	ENSG00000054148.18	ENSE0000361261...	PHPT1-201	9	+
4	PHPT1	ENST00000371661.5	001	136848724	136848780	protein_coding	ENSG00000054148.18	ENSE0000145578...	PHPT1-202	9	+
5	PHPT1	ENST00000371661.5	002	136849111	136849590	protein_coding	ENSG00000054148.18	ENSE0000145578...	PHPT1-202	9	+
6	PHPT1	ENST00000371661.5	003	136850013	136850137	protein_coding	ENSG00000054148.18	ENSE0000372475...	PHPT1-202	9	+
7	PHPT1	ENST00000371661.5	004	136850506	136850560	protein_coding	ENSG00000054148.18	ENSE0000364848...	PHPT1-202	9	+
8	PHPT1	ENST00000371661.5	005	136850755	136851022	protein_coding	ENSG00000054148.18	ENSE0000355316...	PHPT1-202	9	+

Database Structure Browse Data Edit Pragas Execute SQL											
Table: EXONS_GTF Filter in any column											
	GeneName	XcriptID	ExonNo	CoordFrom	CoordTo	XcriptType	GeneID	ExonID	XcriptName	Chrom	Strand
1	PHPT1	ENST00000247665.12	001	136849409	136849590	protein_coding	ENSG00000054148.18	ENSE00001888328.2	PHPT1-201	9	+
2	PHPT1	ENST00000247665.12	002	136850013	136850137	protein_coding	ENSG00000054148.18	ENSE00003724754.1	PHPT1-201	9	+
3	PHPT1	ENST00000247665.12	003	136850755	136851024	protein_coding	ENSG00000054148.18	ENSE00003612613.2	PHPT1-201	9	+
4	PHPT1	ENST00000371661.5	001	136848724	136848780	protein_coding	ENSG00000054148.18	ENSE00001455785.1	PHPT1-202	9	+
5	PHPT1	ENST00000371661.5	002	136849111	136849590	protein_coding	ENSG00000054148.18	ENSE00001455787.1	PHPT1-202	9	+
6	PHPT1	ENST00000371661.5	003	136850013	136850137	protein_coding	ENSG00000054148.18	ENSE00003724754.1	PHPT1-202	9	+
7	PHPT1	ENST00000371661.5	004	136850506	136850560	protein_coding	ENSG00000054148.18	ENSE00003648487.1	PHPT1-202	9	+
8	PHPT1	ENST00000371661.5	005	136850755	136851022	protein_coding	ENSG00000054148.18	ENSE00003553160.1	PHPT1-202	9	+

10. Paste this selection into a text editor (such as notepad), and add in a hash symbol **between each transcript** as shown. Note – *it is important to include a hash after the final transcript*, otherwise it will be skipped. Save the “**experiments.txt**” file

- Note that while CoordFrom and CoordTo have been included here, they are not required. Any field can be entered to the right of the Exon Number, as long as it is preceded by a tab. This will be for information purposes only
- Note also that the information for any number of transcripts (or parts of transcripts) can be included here. This file makes the data available to the next step in the process flow

```
PHPT1 ENST00000247665.12    001    136849409    136849590
PHPT1 ENST00000247665.12    002    136850013    136850137
PHPT1 ENST00000247665.12    003    136850755    136851024
#
PHPT1 ENST00000371661.5     001    136848724    136848780
PHPT1 ENST00000371661.5     002    136849111    136849590
PHPT1 ENST00000371661.5     003    136850013    136850137
PHPT1 ENST00000371661.5     004    136850506    136850560
PHPT1 ENST00000371661.5     005    136850755    136851022
#
```

11. Run the Python program “[create_transcript_objects_from_sql.py](#)”

12. A new Python file named “[transcript_vars_objects_from_sql.py](#)” will be **automatically generated** into the “Work” directory

- This file contains transcript data in the form of Python variables (strings) and Python object/instance definitions (the class for these is defined in another Python program). The snippet below shows the type of Python code which will be created in the auto-generated Python program.
- The line which is highlighted and printed in red ink will be used in the following steps of the process (this is an example of an Object Instance definition of the **Transcript** class from “[fastq_builder.py](#)”)

```
PHPT1_ENST00000371661_5_001_002_003_NAME = 'PHPT1_ENST00000371661.5_001_002_003'
PHPT1_ENST00000371661_5_001_002_003_FULL_SEQUENCE =
'GGCCGGGAACGCCCTGGACCGGAATAATTCCAGGGGGCAAGAGCTTTCGAACCAACCTCGCGGGCCGCTAACTGCCCCGTTCCAAGGGTGCCACCG
GAC
PHPT1_ENST00000371661_5_001_002_003 = Transcript(PHPT1_ENST00000371661_5_001_002_003_NAME,
PHPT1_ENST00000371661_5_001_002_003_FULL_SEQUENCE, [57,480,125])
```

13. Open the Python program named “[fastq_builder.py](#)” for editing. Scroll down to the region marked:

```
#####
# Configuration:
#####
```











14. In this section, **two groups** of “Reads” are declared in terms of transcripts (or regions thereof), volume of data (i.e. how many “Reads” will be produced per group), and the length of each “Read” (all “Reads” will have the same length for any given run of **“fastq_builder.py”**)
15. Following on from the example taken from the “DB Browser for SQLite” above, we can define a set of “Reads” to be produced as per the example here:
 - The yellow highlights are the object names (taken from **“transcript_vars_objects_from_sql.py”**)
 - The values in red are the amounts of “Reads” PER JUNCTION to be created for each transcript (so the top example has 4 junctions x 400 = 1600 “Reads” in total)
 - The replicates_per_group specifies how many copies of each “Reads” file will be created (each will have a different name, with a similar pattern)
 - The example below shows read_length = 50. This will mean that for each junction, there will be a 25nt region taken from each side of the junction, where possible. The program will compensate if there aren’t enough nucleotides on a given side of the junction (by taking extra nucleotides from the other side of the junction instead)
 - Extra transcripts can be added to each group by placing extra sub-lists (Python code format will require the addition of commas between them) into the parent “group” lists

```
group1 = [
    [PHPT1_ENST00000371661_5_001_002_003_004_005, 400]
]

group2 = [
    [PHPT1_ENST00000247665_12_001_002_003, 1000]
]

replicates_per_group = 5
read_length          = 50
```

16. Once configuration is complete, run the program **“fastq_builder.py”** – the files containing synthetic RNA-Seq “Reads” in .fastq format will be created in the “Work” directory:

 synth_data_grp_1_replicate_1.fastq	17/10/2020 18:50	FASTQ File
 synth_data_grp_1_replicate_2.fastq	17/10/2020 18:50	FASTQ File
 synth_data_grp_1_replicate_3.fastq	17/10/2020 18:50	FASTQ File
 synth_data_grp_1_replicate_4.fastq	17/10/2020 18:50	FASTQ File
 synth_data_grp_1_replicate_5.fastq	17/10/2020 18:50	FASTQ File
 synth_data_grp_2_replicate_1.fastq	17/10/2020 18:50	FASTQ File
 synth_data_grp_2_replicate_2.fastq	17/10/2020 18:50	FASTQ File
 synth_data_grp_2_replicate_3.fastq	17/10/2020 18:50	FASTQ File
 synth_data_grp_2_replicate_4.fastq	17/10/2020 18:50	FASTQ File
 synth_data_grp_2_replicate_5.fastq	17/10/2020 18:50	FASTQ File

17. An example of the contents of such a file are shown below

- Line 1 is the header. It can consist of almost anything, and the software here uses it to allow easy reference back to the criteria used to create the data
- Line 2 is the actual sequence of the “Read” (taken from the reference genome)
- Line 3 contains general information (and a conventional “+” symbol)
- Line 4 contains the PHRED scores (one “score” per nucleotide). “I” is the highest value of “certainty”, used here so that any alignment tool used on this data (to obtain a .bam or .sam file) will have no “doubt” about the accuracy of the “Reads”

```
@SYNTH_DATA length=50 Read No: 1 Junction: 1/2 Xcript Name: PHPT1_ENST00000247665_12_001_002_003 in Group: 2  
CCAGCCTCACATGTGCGGCTGGGAGATAACAGTGTGCACACCCAGCA  
+SYNTH_DATA 1 length=50 Phred Scores:  
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
```

18. This data can now be passed into an Alignment Tool for alignment to the Reference Genome

- The idea behind using Group1 and Group2 is to allow the easy creation of comparative data, generally used for Differential Splicing Analysis, i.e. the comparison of how a gene expresses one transcript more than another in a Control v Treatment scenario

3.4 Ensembl Route of the Process Flow

The following steps should be taken (the same gene and transcripts as in 3.3 will be used here):

1. Navigate to https://www.ensembl.org/Homo_sapiens/Gene/Summary?db=core;g=ENSG00000054148;r=9:136848724-136851027
2. Click on the (highlighted) Sequence option as shown below:

Location: 9:136,848,724-136,851,027
Gene: PHPT1

Gene-based displays

- Summary
- Splice variants
- Transcript comparison
- Gene alleles
- Sequence**
- Secondary Structure
- Comparative Genomics
- Genomic alignments
- Gene tree
- Gene gain/loss tree
- Orthologues
- Paralogues
- Ensembl protein families
- Ontologies
 - GO: Molecular function
 - GO: Biological process
 - GO: Cellular component
- Phenotypes
- Genetic Variation
 - Variant table
 - Variant image
 - Structural variants
- Gene expression
- Pathway
- Regulation

Gene: PHPT1 ENSG00000054148

Description phosphohistidine phosphatase 1 [Source:HGNC]

Gene Synonyms CGI-202, DKFZp564M173, HSPC141, PHP14,

Location [Chromosome 9: 136,848,724-136,851,027](#) forv GRCh38:CM000671.2

About this gene This gene has 6 transcripts ([splice variants](#)) and

Transcripts [Hide transcript table](#)

Show/hide columns (1 hidden)					
Name	Transcript ID	bp	Protein	Biotype	CDD
PHPT1-201	ENST00000247665.12	577	125aa	Protein coding	CCDS
PHPT1-202	ENST00000371661.5	985	124aa	Protein coding	CCDS4
PHPT1-205	ENST00000492540.5	1316	No protein	Processed transcript	.
PHPT1-204	ENST00000463215.1	627	No protein	Processed transcript	.
PHPT1-203	ENST00000462205.5	619	No protein	Processed transcript	.
PHPT1-206	ENST00000497413.1	618	No protein	Processed transcript	.

3. A “Download Sequence” button appears on the screen. Click on this, and from the popup window, select the options as shown, taking care to:
 - set the 5’ and 3’ flanking sequence options to 0
 - select **only** the Exons checkbox
 - leave File name and File format as they initially were

Download sequence

File name:

Homo_sapiens_PHPT1_sequence

File format:

FASTA

Preview

Download

Download Compressed

Settings

Sequences to export:

☐ Select/deselect all

☐ cDNA (transcripts)

☐ Coding sequences (CDS)

☐ Amino acid sequences

☐ 5' UTRs

☐ 3' UTRs

☒ Exons

☐ Introns

☐ Genomic sequence

5' Flanking sequence (upstream):

0

*(Maximum of 1000000)

3' Flanking sequence (downstream):

0

*(Maximum of 1000000)

4. Click the “Download” button – a new file will appear in the standard downloads folder. Move that file from there to the “Work” directory
 - As many such files as the user wishes to use can be placed into the “Work” directory at the same time. The following steps will process them all, provided they are named with the pattern “Homo_sapiens_*_sequence.fa”, where * is replaced by the gene name (the downloaded files from Ensembl do not need to be changed as they already follow this pattern)
5. Run the Python program “[create_transcript_objects_from_ensembl.py](#)”

6. A new **Python file** named “**transcript_vars_objects_from_ensembl.py**” will be **automatically generated** into the “Work” directory
 - This file will contain the transcript data extrapolated from the downloaded Ensembl files. The extrapolated data will be in the same format as shown in “point 12” of 3.3 above
 - Transcript Object Definitions can be copied and pasted to the Python program named “**fastq_builder.py**”, and from here on, the steps in 3.3 can be followed exactly (from “point 12” onwards)

4 Bar Charts: Compare Transcripts from same Gene

The secondary objective of this tool is to generate a Bar-Chart allowing an easy visual comparison of one or more Transcripts *from the same Gene*.

This is a useful feature when testing 3rd Party differential splicing applications, as it displays both a visual overview of the transcripts and a more precise “coordinate offset” shown in listing format.

The following steps should be taken:

1. If not already done, run steps 1-7 in 3.3 above (so that we now have a DB with the required tables and records)
2. Create a text file name “**bar_chart_config.txt**” in the “Work” directory
3. Using the same example gene (PHPT1) and protein coding transcripts as per the previous examples in this user manual, copy the fields from the “DB Browser for SQLite” below:

Database Structure Browse Data Edit Pragas Execute SQL											
Table: EXONS_GTF											
	GeneName	XcscriptID • 1	ExonNo	CoordFrom	CoordTo	XcscriptType	GeneID	ExonID	XcscriptName	Chrom	Strand
	PHPT1	ENST00000247665.12	001	136849409	136849590	protein_coding	ENSG00000054148.18	ENSE00001888328.2	PHPT1-201	9	+
1	PHPT1	ENST00000247665.12	002	136850013	136850137	protein_coding	ENSG00000054148.18	ENSE00003724754.1	PHPT1-201	9	+
2	PHPT1	ENST00000247665.12	003	136850755	136851024	protein_coding	ENSG00000054148.18	ENSE00003612613.2	PHPT1-201	9	+
3	PHPT1	ENST00000371661.5	001	136848724	136848780	protein_coding	ENSG00000054148.18	ENSE00001455785.1	PHPT1-202	9	+
4	PHPT1	ENST00000371661.5	002	136849111	136849590	protein_coding	ENSG00000054148.18	ENSE00001455787.1	PHPT1-202	9	+
5	PHPT1	ENST00000371661.5	003	136850013	136850137	protein_coding	ENSG00000054148.18	ENSE00003724754.1	PHPT1-202	9	+
6	PHPT1	ENST00000371661.5	004	136850506	136850560	protein_coding	ENSG00000054148.18	ENSE00003648487.1	PHPT1-202	9	+
7	PHPT1	ENST00000371661.5	005	136850755	136851022	protein_coding	ENSG00000054148.18	ENSE00003553160.1	PHPT1-202	9	+
8	PHPT1	ENST00000371661.5									

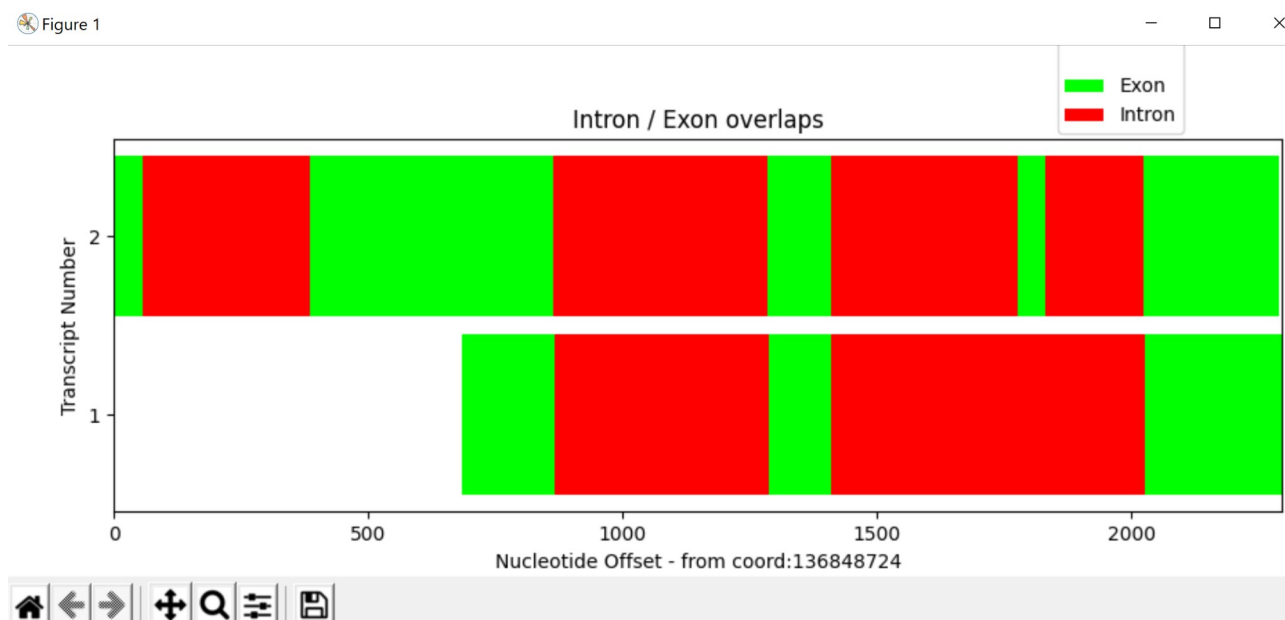
4. Paste the fields into “**bar_chart_config.txt**” and add a “#” symbol at the end of each transcript, as follows:

- Note that in this case, the CoordFrom and CoordTo are *required fields*.

```
PHPT1 ENST00000247665.12      001   136849409   136849590
PHPT1 ENST00000247665.12      002   136850013   136850137
PHPT1 ENST00000247665.12      003   136850755   136851024
#
PHPT1 ENST00000371661.5       001   136848724   136848780
PHPT1 ENST00000371661.5       002   136849111   136849590
PHPT1 ENST00000371661.5       003   136850013   136850137
PHPT1 ENST00000371661.5       004   136850506   136850560
PHPT1 ENST00000371661.5       005   136850755   136851022
#
```

5. Run “**bar_chart.py**”.

6. The visual bar chart will take on this appearance. Note that:
 - a) The transcript(s) which begin furthest to the left of the bar chart has/have an offset of 0. This means that this transcript begins at Coordinate 136848724 (see the footer of the chart) of the Chromosome in which the Transcript is located.
 - b) Such large numbers are cumbersome to read, and so the visual transcripts begin at 0, as do the listed transcript coordinates below



7. The listed transcript offsets will appear as follows:

Output Report:

Nucleotide Offset - From Coordinate: 136848724

Transcript: 1

Exon: [685, 866]
 Intron: [867, 1288]
 Exon: [1289, 1413]
 Intron: [1414, 2030]
 Exon: [2031, 2300]

Transcript: 2

Exon: [0, 56]
 Intron: [57, 386]
 Exon: [387, 866]
 Intron: [867, 1288]
 Exon: [1289, 1413]
 Intron: [1414, 1781]
 Exon: [1782, 1836]
 Intron: [1837, 2030]
 Exon: [2031, 2298]

Transcript Number	Gene Name	Transcript Id	Exons
1	PHPT1	ENST00000247665.12	['001', '002', '003']
2	PHPT1	ENST00000371661.5	['001', '002', '003', '004', '005']

8. The usefulness of the specific transcript listings is demonstrated by, for example, the observation that while the two rightmost exons in each of the visualised transcripts appear to start and end in the same place, they in fact begin at the same point (Offset 2031) but end at different points (2300 and 2298). This level of information is useful for testing 3rd Party Differential Splicing Quantification software, and the Bar Chart feature was added to this suite specifically for that reason.