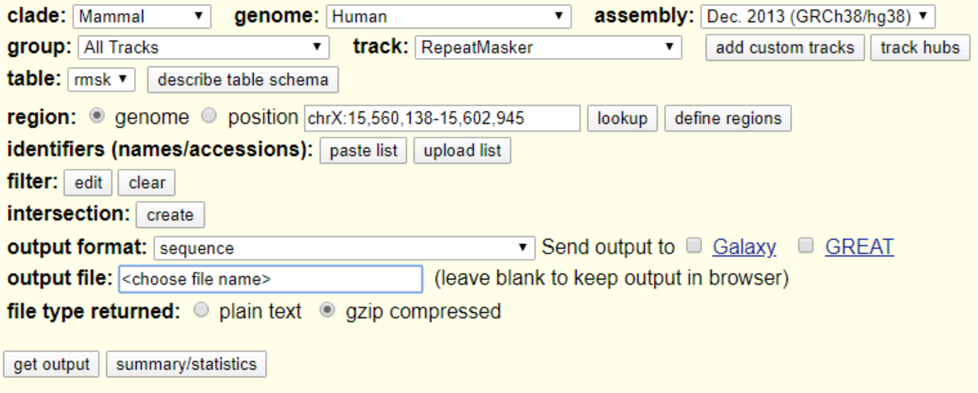
Acquiring Retrotransposon Data From UTSC Table Browser

To skip the long process of acquiring and processing the data from UTCS Table Browser and upload data from a smaller example file, download the example file from the github repository DanielRGrant/retrobase, from the folder /data/example_data. Then, skip to command 10.

1) From UTSC acquire retrotransposon data with the following inputs. Select filter and enter the retrotransposon class enclosed in asterisks; (e.g. *l1*):



clade: Mammal ▾ genome: Human ▾ assembly: Dec. 2013 (GRCh38/hg38) ▾

group: All Tracks ▾ track: RepeatMasker ▾

table: rmsk ▾

region: ● genome ○ position chrX:15,560,138-15,602,945

identifiers (names/accessions):

filter:

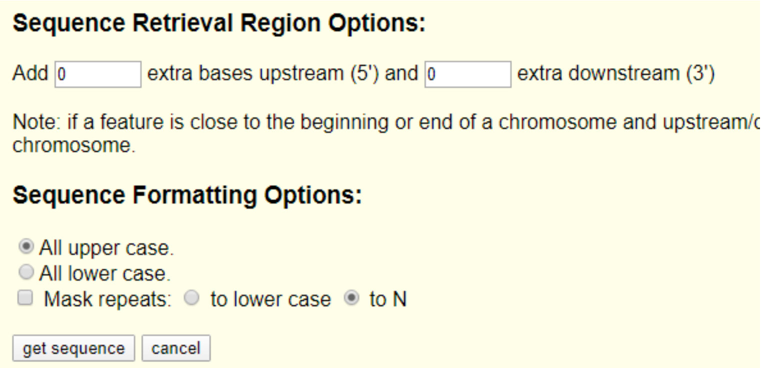
intersection:

output format: sequence ▾ Send output to □ Galaxy □ GREAT

output file: <choose file name> (leave blank to keep output in browser)

file type returned: ○ plain text ● gzip compressed

2) Click get output **and ensure settings are as follows on the following page and click get sequence:**



Sequence Retrieval Region Options:

Add 0 extra bases upstream (5') and 0 extra downstream (3')

Note: if a feature is close to the beginning or end of a chromosome and upstream/c chromosome.

Sequence Formatting Options:

● All upper case.
○ All lower case.
□ Mask repeats: ○ to lower case ● to N

3) Navigate to the location in the cloned repository containing the file, `retrobase_commands-0.1-py3.whl`.

Pip install using:

```
pip3 install retrobase_commands-0.1-py3.whl
```

4) Call the command

```
retrobase_commands translate --input_file=<input_file> --
output_file=<output_file> superfamily=<superfamily>
```

Where:

- `<input file>` is the path to the file you downloaded from UCSC table browser

-`<output file>` is the path to create the output file

-`<superfamily>` is the retrotransposon superfamily/class

5) Install local blast from https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/

6) Create a local blast database using:

```
makeblastdb -in <input file> -parse_seqids -blastdb_version 5 -title
"<database title>" -dbtype prot
```

7) Get protein sequences in FASTA format from uniprot representing known retrotransposon protein sequences.

If you are following these steps to test the code you can use the 5 .fasta files located in the github repository DanielRGrant/retrobase, in the folder /data/standard_orf_seqs

8) Use sequences acquired in step 7 as query sequences in psi-blast queries against the database created in step 6 using the following command:

```
psiblast -query <RT -protein-FASTA> -db <database-title> -out
<output-file-name> -outfmt 5 -max_target_seqs=1000000
```

Where:

-`<RT-protein-FASTA>` is a protein sequence obtained in step 7

-`<database-title>` is the title of the database created in step 6

-`<output-file-name>` is the name of the psi-blast output that must be formatted: `<protein name>_psiblast.txt`

`-max_target_seqs` set to a very large number ensures all sequences with e-values under the threshold will be included

9) Create a JSON file relating DNA records, translated protein sequences, predicted protein labels and associated data using the retrobase_commands command:

```
retrobase_commands assign_proteins --dna_input_file=<dna_input_file>
--protein_input_file=<protein_input_file> --
psiblast_directory=<psiblast_file_path> --outputfile=<outputfile> --
superfamily=<superfamily> --protein_names=<protein_names> --
genome=<genome>
```

Where:

-<dna_input_file> is the path to the path and filename to the fasta file output from UTSC table browser

-<protein_input_file> is the path and filename to the FASTA created by the tanslate command

-<psiblast_directory> is the path to the files output by psiblast query. These file names must be formatted

-<outputfile> is the name of the JSON file to be created

-<superfamily> is the name of the retrotransposon class

-<protein_names> is the names of the proteins queried using psiblast. The names must be written the same as those in the psiblast output filenames. Multiple protein names should be input by repeating the option as so: --protein_names=Gag --protein_names=Pol --protein_names=Pro

-<genome> is the name of the genome from which the original retrotransposon sequences came (e.g. hg_38)

10) Add the path to the Retrobase base directory to the python path

11) Upload records to database using the retrobase_commands command:

```
retrobase_commands upload_records --input_file=<input_file> --
path_to_django_settings=< path_to_django_settings>
```

Where:

-<input_file> is the path and filename to the JSON file created in step 9

12) Acquire protein function data from Uniprot using the command:

```
retrobase_commands enter_uniprot_data --accession_ids=<accession_id>
--protein_names=<protein_name>
```

Where:

-<accession_id> is the accession ID from of the Uniprot entry from which the standard sequence of a given protein came. For multiple accession IDs, simply use the option multiple times in the command. Multiple accession IDs must be listed in **the same order** as the protein names

-<protein_name> is the name of the protein for which the Uniprot data is being collected (must be written as in Retrobase database). For multiple protein_names, simply use the option multiple times in the command. Multiple protein names must be listed in **the same order** as the accession IDs.

The accession IDs of the example .fasta files are as follows:

Env: Q69384

Gag: Q7LDI9

Pol: Q9BXR3

Pro: Q9Y6I0

Rec: Q69383