DOCUMENTATION OF Farfalloni | RETROtransposon

RETROtransposon, a web application for the collection, analysis and identification of retrotransposon elements.

Maroua Akkari[a], Esteban Gómez Cifuentes[b] , Vithusaa Selvachandrarajah [c]
Student ID. 140067549[a], Student ID. 170531937[b], Student ID. 170772552[c]
MSc Bioinformatics – Software Development Group Project
Queen Mary University of London

1. INTRODUCTION

The transposable elements (TE) are small DNA fragments that can move, or jump, in the genome. These elements play an important role in the evolution of the human genome and help to the development of the form and function of many genes [1, 2]. There are two classes of TE: the DNA transposons, capable of cutting themselves and pasting in another part of genome, which represent 3% of the human genome; and the retrotransposons (approximately 40% of the genome) which share a similar action mechanism to the DNA transposon, differing in that they replicate using a reverse transcription of an RNA intermediate [2].

Likewise, the retrotransposons can be classified by those that have long terminal repeats (LTR), known as LTR retrotransposon; and those that don't have LTR, known as non-LTR retrotransposon.

The LTR retrotransposons, which represent 8% of the human genome and are genetic residues from infections cause by retroviruses (hence the homology between the genes present in the LTR retrotransposons and proteins from retroviruses), can be separated according to the reverse transcriptase that they encode; however, in humans, the most studied group is HERV (Human endogenous retrovirus) [3]. In the case of non-LTR retrotransposons, there are two subgroups: Long interspersed elements (LINE), where LINE-1 (17% of the human genome) is the only currently active retrotransposon family of this group; and Short interspersed nuclear elements (SINE) which did not evolve from autonomous elements and are not as abundant as LINE-1 (their main families, SVA and *Alu* elements, represent 12% of the genome) [1, 4, 5].

Although many of the LINE-1 members (>99.8%) have lost their function due 5' truncations, internal rearrangements and mutations - which is also the case for the HERV members, there is still a portion of retrotransposons with intact genes that retain their function [3, 6]. For this reason, RETROtransposon, a web-based application that presents in an orderly way (allowing to perform searches and view general statistics) information about the LINE-1 and HERV class members was developed. RETROtransposon also contains a small tool capable of identifying retrotransposon elements from data collected through mass spectrometry analysis and stores this information for future use.

2. SOFTWARE DEVELOPMENT AND THEORETICAL BASES

Currently there are several databases of retrotransposons such as L1Base2, dedicated to active LINE-1 members, HERVd, only for HERV retrotransposons, or dbRIP, a database of retrotransposon insertion polymorphism in humans [7-9]. Although these databases have complete information and are user friendly, all of them focus only in a specific type of retrotransposon.

RETROtransposon seeks to integrate the information of diverse classes and families of retrotransposons (currently has LINE1 and HERV, but it is expected to expand its database) with the use of a search engine that allows to identify elements of interest more easily. Besides that, the application has information about intact ORFs and theirs translated products that can be useful to identify possible retrotransposon in mass spectrometry files.

2.1. Data Collection

The data for the diverse retrotransposons were collected using the Table Browser tool from UCSC Genome Browser page (https://genome-euro.ucsc.edu/index.html). The human genome was used (assembly GRCh38/hg38) and RepeatMaster for the retrotransposon selection. For LINE-1 the data was filtered using the key word "L1" in the repFamily option meanwhile, for HERV, the key "LTR" was used in the repClass option. The elements that don't belong to the ERV or LINE-1 family were discarded through a simple search. Finally, the DNA sequences for each retrotransposon also were collected using Table Browser using the same search parameters mentioned before. All the information was save in files CSV (chromosome location, rename, repClass, etc) and FASTA (DNA sequence) files.

2.2. Open Reading Frame Identifications

For the identification of active open reading frames (ORFs) a program using the programming language python 2.7 (https://www.python.org/download/releases/2.7/) was design. The program is able to, first, search for intact ORFs (sequences that presents and Start and End Codon), and then filters, according to diverse parameters, the possible ORFs that code for one of the proteins present in the retrotransposon (**see HERV.py and L1.py files in the folder "ORF Identifications"**).

Since the python algorithm translates the DNA sequence and its reverse complementary, it's possible that a couple of sequences are repeated due to some retrotransposons have sequences for the positive strand and the negative strand. This inconvenience is easy to fix but since the aim was to avoid losing the least number of sequences, the decision was to leave the repeated amino acid sequences.

LINE-1:

LINE-1 retrotransposons consist in two non-overlapping ORFs known as ORF1 and ORF2. ORF1 codes for a RNA binding protein of approximately 500 amino acids, while ORF2 codes for a multidomain protein of approximately 1200 amino acids that has an endonuclease and a reverse transcriptase, in addition to a cysteine rich domain [10, 11]. Recently, a third ORF has been identified, ORF0, located in the antisense strand that codes for a predominately nuclear polypeptide which has an important role for the mobility of the LINE-1 elements [12].

The python algorithm takes the DNA sequences from the FASTA file obtained from UCSC Genome Browser and identify only the intact ORFs. In addition, knowing the regular size of the ORFs in LINE-1, a threshold of 300 amino acids were choose (small translated products have been reported in LINE-1) [4] to only translate (using a designed function) the more likely ORFs.

Finally, knowing the proteins that are coded from these ORFs, regular expression were designed to identify functional motifs in the translated products:

ORF1: codes for a protein that has a highly conserved C-terminal domain in the different families of LINE. Based in that information and reviewing multiples alignments obtained from L1Base2 database, the motif "RREWGPIFN[IT]L" was built [12].

ORF2: since this ORF has a bigger size than ORF1, two motifs, based in multiple alignments from conserved domains which were then verified using the reported sequences in L1Base2 [7, 14], were made. The motifs were: "CWRGCG" located near to end of the sequence, and "LKI[KT]GWRK[IS]YQ[AV]N" identified inside of the functional domain of the endonuclease coded for ORF2.

ORF0: although there isn't much information about ORF0, a small motif, "M[AV][GD][ATY]", were made base in the alignment reported in Denli A, *et al*., 2015 [5].

Using those motif, 816 identifications were made for ORF1, 3875 for ORF2 and 2 from ORF0. The CSV file generated for python algorithm was combined with the CSV file generated from Table Browser (since the same parameters were used to search for the general information and the DNA sequences, the order of retrotransposon were the same in both CSV files).

HERV:

Since HERV retrotransposons come from retroviral elements inserted in the genome after infection, their ORFs code for 3 common polyproteins of the retroviruses genome: Gag (~ 500 amino acids), Pol (~ 700 amino acids) and Env (~ 500 amino acids). In some retroviruses, it is possible to identify a fourth gene, Pro, between Gag and Pol; however, it is more common in HERV to find this protein as part of the Gag or Pol [15, 16].

As in LINE-1, the python algorithm identifies the ORFs that translate a protein with a size longer than 300 amino acids (proteins with this length have been identified) [17], and motifs were designed using conserved domains.

Gag: the Zinc Finger CCHC type domain presents in Gag was used to generate the regular expression "CX2CX4HX4C" [18].

Pol: a polyprotein that codes for a retrovirus reverse transcriptase, a RNase H and an integrase. For the correct identification of Pol two motifs from the reverse transcriptase were used ("LPQG" and "YXDD") and a functional motive from RNase H ("[GE]X3D") [16].

Env: Three motifs ("CX6C[CHRYS]", "CX7[CY][CS]" and "CX6F") were designed. These motifs were obtained from a common disulfide bond identified in a multiple alignment of the different families of HERV made by Laurence Benit, *et al*., 2001 [17].

Using those motifs, it was possible to identify 33 ORFs from Gag, 139 from Pol and 95 from Env. It is not a surprise to find fewer identifications in HERVs than LINE-1 since the HERV-K family is the only one that presents some function [15].

2.3. Software Architecture

RETROtransposon was designed using the programming language python 2.7 and the python microframework Flask (http://flask.pocoo.org/). A diagram of the software architecture used to build RETROtransposon can be seen in the Figure 1 and the folder hierarchy of the program can be seen in the Figure 2.

RETROtransposon used other tools and other programming languages, such as HTML and java, to work correctly. The main application, from now on calls app.py (see **app.py** in the main directory), connects everything using different functions as it's possible to see below.

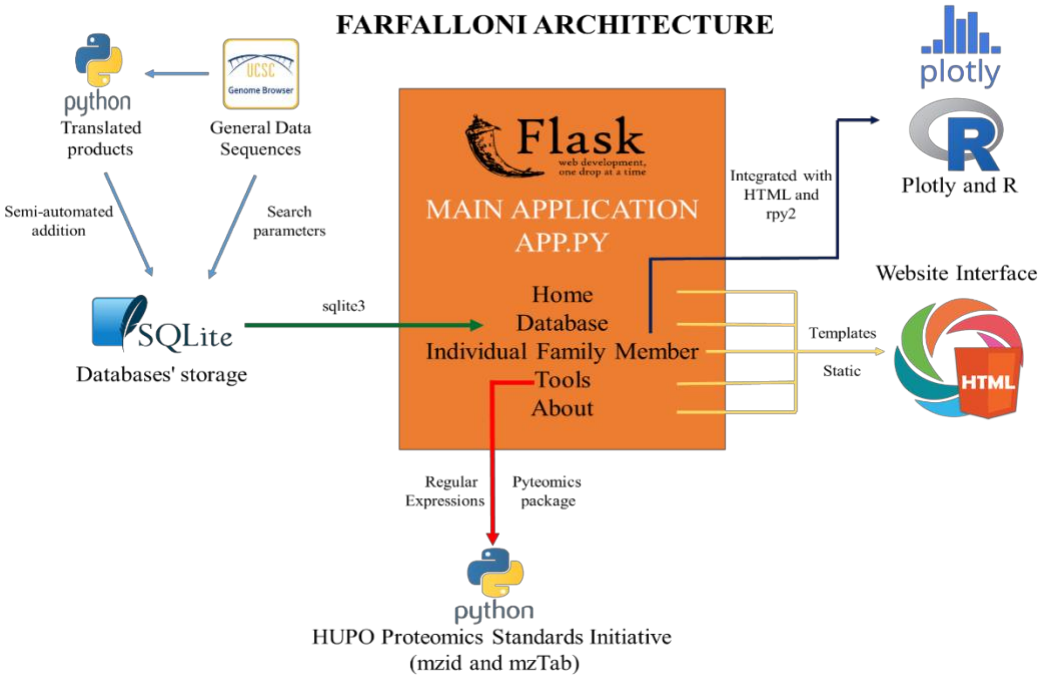


Figure 1. RETROtransposon Software Architecture.

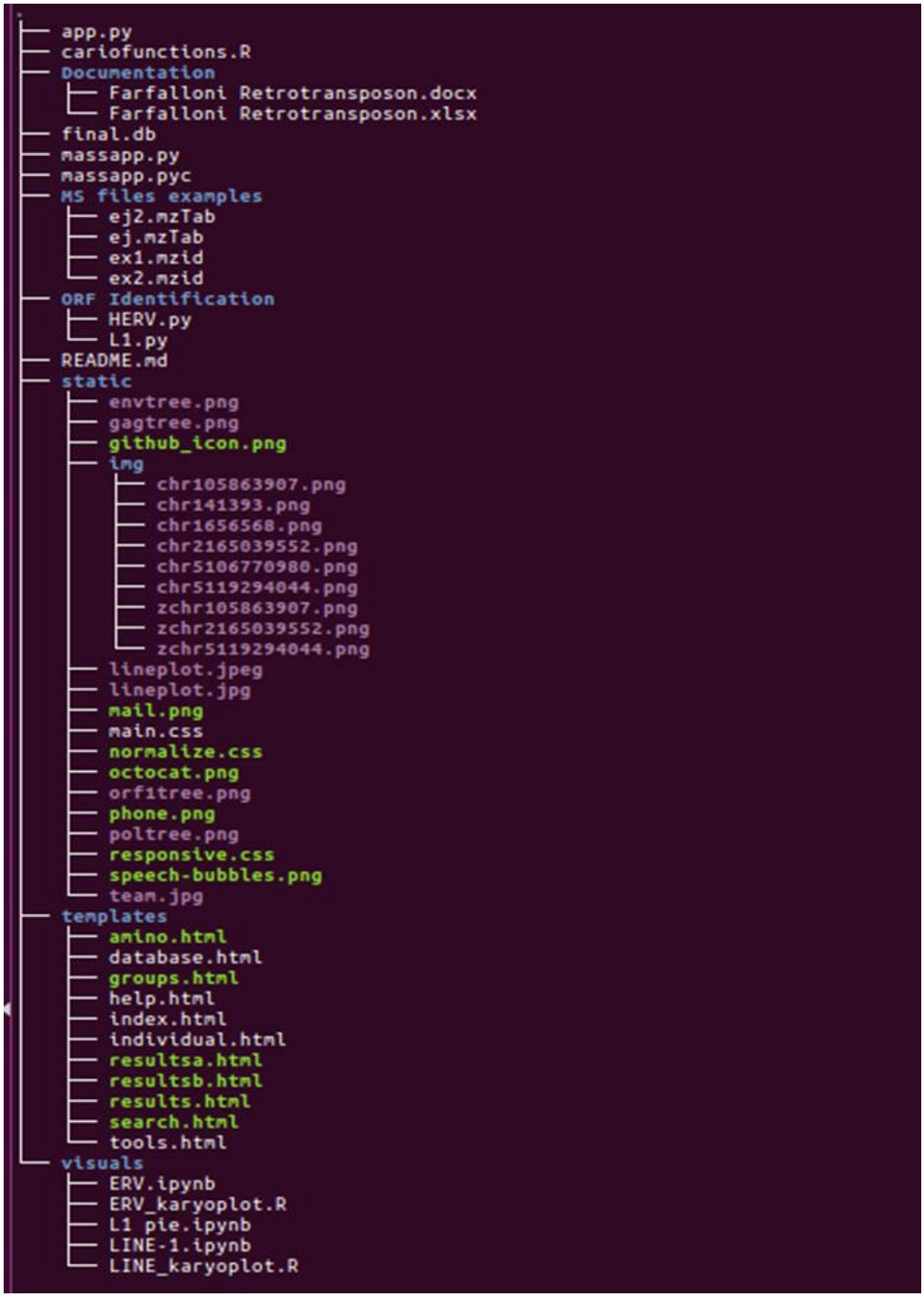


Figure 2. Directory tree of RETROtransposon Software.

2.3.1. Database Management

SQlite (https://www.sqlite.org) was used for the management of the RETROtransposon database. SQLite was choose since presents more features than Panda package and works really well as the database engine for most low medium traffic websites. Besides that, SQLite doesn't present problems of centralization and control as other databases such as MySQL or Oracle, which were design for biggest projects than RETROtransposon, which works locally and needs a local database.

To create the proper database, the HERV and LINE-1 CSV files (with the general information, the DNA sequence and the amino acid sequences if applies) where combined and compiled using the desktop application DB Browser for SQLite (http://sqlitebrowser.org/). DB Browser allows to import CSV files and to create database (.db files) from them. In addition, wtih DB Browser you can add indexes to call the database faster and use it in a more efficient way.

To connect the SQlite database with app.py (the main application), the package sqlite3 was used. Using the function "connect" from sqlite3, it is easy to link the database with app.py, then with "row_factory" it's possible to call the database row by row for an easy way to reach its information (see app.py, section "Calling the SQlite Database").

After the database was connected with the app.py, the "cursor" object was created to allow "execute" commands in the database (see app.py, section "App Website pages – database()"). The commands used more to get information from the database was "SELECT", which selects all the elements that pass the condition established by "WHERE", and "LIKE", which allows to make a search without expecting full matches.

This process is generally fast but with big databases could be slow; the use of indexes can make the search of information faster, but when "WHERE" has the condition "OR" instead of "AND", the search could remain slow since the command has to look for all the possible matches (see app.py, section "App Website pages – results()").

Due to the database having a large number of entries (approximately 2 millions rows) at the beginning there was a problem uploading the whole database and showing it through the HTML template "database". Since the point is to show all the data available in the database, the solution was grouping the retrotransposons by name in the main table (this also allowed to show the number of elements that share the same name) and then display a second table only with the information of each group in a different page (see HTML template "groups" in the template folder and app.py section "App Website pages – database()").

2.3.2. Interface Design

HTML was used to create the templates of RETROtransposon, a standard layout of head, body and footer was used throughout these files. The head contains external links to any fonts used (e.g. from Google fonts), CSS files and external jQuery script links. The footer is the GitHub octocat logo which links to the Farfalloni GitHub repository. The body of each page contains the navigation bar which is the same for each page, but depending on the page currently being used, changes slightly. What differs in this section depends on the content of each page, for example tables, paragraphs, images and any jQuery script would also be included here. Furthermore, interactive graphs that were made using an external website were displayed in the website using anchor tags inside of an unordered list to include the long external HTML links from plotly (see HTML template "database" in templates folder).

The HTML files are templates that are called in the app.py file to run the website on a local server. In a similar way, code can be called from within the HTML file to retrieve specific information from the database. Taking the database page as an example, a loop was created to call each row and column of the database (in the app.py file), then in the HTML file another loop was made to call specific rows and input them into a table, e.g. <td>{{re["genoName"]}}</td> takes the genome name from the database for each retrotransposon and puts them in a column of the table (see HTML template "database" in templates folder).

Another calling method used to connect the database to the HTML files was method='get' whenever an input submission form was used. For example, in the search page, if a user only wants to see LINE-1 retrotransposons with ORF2 sequences, this method will search for elements with the name "ORF2" via the app.py file and return the relevant information to the HTML page. A similar process happens whenever a user uploads information to an input submit form on a HTML page.

A HTML file alone will make a functional webpage, however it will be very plain and have no design, therefore in order to add style and personalise the design of the actual interface of the webpages, various CSS files had to be included; a CSS file adds style to a HTML page.

The **main.css** file (see the static folder) contains all the styling for RETROtransposons; for example, the general styles of the website including: margins, website name, font size and colours, colours and alignment of the navigation bar, margins and positioning of tables and images.

To connect the styling CSS files with the HTML files, they were added in the <head> of HTML file in link tags (which allow the use of external files/urls within the file). As HTML files work in ascending order, the normalize.css file was added first as a default style, then the main.css file was added so those changes would override the standard layout, then lastly the jQuery.css files were added (when necessary).

CSS files that needed to be included in the HTML <head> tag were:

- normalize.css to provide default cross browser consistency styling, which was downloaded from GitHub (…/necolas/normalize.css/blob/master/normalize.css)
- main.css which was created manually from scratch, to add style to each section of the web interface: the header (website name and navigation bar) and footer were designed and added to each HTML page so they all have the same template to fill with content. The body of each page is what differs depending on the content, for example tables, text and graphs.
- jquery.dataTables.min.css was added in <script> tags in this section of the HTML pages that needed JavaScript code to carry out particular functions e.g. pagination, and having drop down menus that could not be done using HTML and CSS alone. (see HTML template "database" in templates folder).

A flaw of using CSS and HTML alone was that pagination to the retrotransposon tables could not be implemented as the content of the table was data being called into the table from the database, thus losing functionality; to overcome this jQuery was used. jQuery is a JavaScript library that can be used for animation and manipulation of content in html files; an extremely useful language but more difficult to get the grasp of due to the short time period. All js needs to be in <script> tags to function correctly, in addition, pages that require JavaScript need to include jQuery libraries in script tags in the head section of the HTML document:

- (needed for all HTML pages that use JavaScript) http://cdn.datatables.net.1.10/16/js/jquery.dataTables.min.js
- (needed for HTML pages that use pagination in tables) https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js

On occasion, updating the style of a certain id or class in the main.css file would not work for some reason, despite trying many different solutions. To overcome this, style tags were added in some HTML documents to add specific CSS code inside of them.

All the above components are needed in order to make RETROtransposon a fully functioning, stylish website.

2.3.3. MSRetroFinder

MSRetroFinder is a small tool that allows to upload mzIdentML or mzTab files and performance a comparison of all the peptide identifications in those files against the RETROtransposon database and search if there is any match; in other words, the program shows all the identifications in the MS files that could have come from a translated products from LINE-1 or HERV retrotransposons.

In the case of mzIdentML, the pyteomics packages (https://pythonhosted.org/pyteomics/), and specifically, the mzid function was used to open and read the information of these files. Since the idea was to first have the information that app.py will use, a python module, calls **massapp.py**, was made that contains the function that get the information for the mzIdentML (see massapp.py, function mzidentml, in the main directory). The function uses pyteomics to generate a list of all the identifications from the mzIdentML, then app.py looks for all the peptides in the list and make a search using sqlite3 and the SQlite commands (see app.py, section "App Website pages – resultsb()").

Pyteomics is really helpful but slow to create the list. For that reason a function using regular expressions was made; however, even after seeing that using the regular expressions the creation of the sequences list was faster, at the end pyteomics was choose since the mzIdentML file is separated in different sections (Analysis Sotfware List, Sample Collection, Sequence Collection, Analysis Collection and Data Collection), and each of them, at the same time, are separated in sub sections that are connected between them, which make difficult to call information as if the file was a plain text file [19]. So, for example, it's possible to call all the Peptide Sequence using

regular expression but there is not a way to know which sequence has an identification since that information is found in other section of the file.

On the other hand, mzTab file is also separated in sections (Metadata, Protein Section, Peptide Section, PSM section and Small Molecule Section), but each section contains "independent" data; for example, the Peptide Section contains the sequences of all the peptides in the samples, while PSM section contains the sequences of all the peptides with an identification [20]. Knowing this is easy then to make a regular expression that calls all the peptides in the PSM section and save them in a list (see massapp.py, function mztab, in the main directory).

When the user uploads one of those files on the website application (in the HTML Template "Tools" file there is an "input type file" section that only accepts mzIdentML and mzTab files), app.py uses the function "request.args.get" to get the information about the file, make the peptide list and search for the possible matches in the MSRetroFinder Atlas and the RETROtransposon database.

If the tool finds a match in the database, app.py adds the information of the new peptide in the MSRetroFinder Atlas (a database with the information of previous identifications) using the SQLite command INSERT INTO. This Atlas can be checked for the users and even for the same tool, which can prevent to have repeated peptide sequences in the Atlas (see app.py, "App Website pages – resultsb()").

Although MSRetroFinder is a functional tool, the program needs to have the mass spectrometry file in the same directory of app.py and when the user uploads big files with a great number of peptide identifications, the program runs really slow and it could take a long time to get all the results (see Perspectives for some future recommendations).

2.2. Graphs Construction

2.2.1. Plotly

To create interactive, high quality visual graphs, there are several packages available; however, Plotly (https://plot.ly/) was the best package to work with due to its flexibility to produce a span of various plots. Unlike other tools and packages available, Plotly is an interactive, browser-based graphing library for Python. All visuals with Plotly were created with Jupyter Notebook to execute and plot charts using the Plotly module, followed by its internal modules (see LINE-1.ipynb in the visuals folder). This module results in the interactivity of the chart. It is crucial to ensure that useful packages, such as Pandas and Plotly are already installed, so that they can be called into the python notebook.

The Pandas dataframe module is a convenient method of displaying the database table from the CSV files in to the python notebook. The distinct columns of the database can be viewed and loaded nicely with Pandas. It is important to give the correct file path to the CSV files, in order to view the data. Firstly, the frequency for the chromosomes (genoName) were represented as an

interactive bar chart for both classes of retrotransposons. In this way, users can see the distribution of the retrotransposons at each chromosome. In order to count the chromosome numbers for all LINE and ERV entries individually, the pandas "Series" function will form a data structure for one column of the dataframe. This will be the 'genoName' column. The "series.count" function can be applied once series has been assigned to the "genoName" column. The output will show the chromosomes and their counts. All chromosomes (including the unknowns), and their individual counts, were stored in two separate lists. Having lists will make it easier to plot the variables against each other, using an X and Y axis format. Another list called data was created, which takes the two lists as its arguments where all parameters for the bar chart were specified within. This included the specific colours, widths and opacity of the bars. The list is finally called via the "py.iplot" function, which takes the list name and a file name to produce the interactive bar chart.

Donut charts are another version to a standard pie chart. These charts make it easier for users to see the differences between the slices, and are space efficient due to the blank space inside the donut chart. The interactivity of these plots was chosen to be kept as it allows users to focus on the different frequencies by arc lengths. Both donut charts were produced using Plotly, but for different information for LINE and ERV.

The ERV donut was generated to represent the 5 different replication families. Using Pandas "series.count" function, the frequency for each replication family were obtained. A dictionary was created, storing the ERV replication family names and the counts (frequency). A dataframe was created from this dictionary, in order to calculate the percentage from the frequency and its sum. Once the Plotly modules are called, the donut chart can be formed using the percentages for the ERV family names calculated previously. Another dictionary was essential to store all the parameters for the chart, such as the values, labels, size, title and layout. Due to identical LINE replication family names, a donut chart was produced for the ORF's in the same way as the ERV donut chart. Having similar types of charts for both retrotransposons families will make it easier for the user to follow and understand, as well as giving a balance in the visual graphics on the webpage.

The "iplot" function and a file name were given for every Plotly chart created, along with the 'edit chart' option. This enables users with a Plotly account to make adjustments and to save the plots if necessary. The hover feature in Plotly gives users the power to engage and navigate through the information; an advantage of using interactive charts. Moreover, the flexibility to publish the plots onto a webpage is enabled via each charts' online editing page. Here, the option to embed the plot in a HTML format was chosen to insert all plots on the database page (see HTML template "database" in template folder).

2.2.2. Karyotype

The karyotypes for the retrotransposon families were created using the R package, karyoploteR (https://bioconductor.org/packages/release/bioc/html/karyoploteR.html). An advantage of KaryoploteR is that this package allows any data to be plotted along the genome to acquire a genome-wide observation [21]. In this way, a clear view of the distribution of features amongst the chromosomes can be viewed. The chromosome coordinates, genoStart and genoEnd, were plotted along the genome, so that the distribution of the retrotransposons can be seen. As the karyoplotR tool is a plotting tool, the data will need to be loaded into R first, followed by a series of plotting functions to plot the data to the genome [21]. Once the data to be plotted from the database table is extracted, it is stored in two functions; one for the start coordinate, and another for the end coordinate. These functions will need to be called for each chromosome specifically. Hence, the start and end coordinates for every chromosome can be retrieved from the data file. The GRanges object is used to create a data frame that stores the chromosome, and the individual start and end chromosome functions. The package has the plotKaryotype function, which will automatically generate the genome specified ('hg38'), retrieved from the USCS genome browser [21]. The kpPlotDensity function computes and plots the densities of the features along the genome. A density plot was chosen to be used rather than the regions, as it gives a more refined plot, with clearer regions. Parameters such as the genome, the data storing the chromosome coordinates, and customisations including colours, can be included in this function to produce the karyoplots (see ERV_karyoplotR and Line_karyoplotR in visuals folder).

2.2.3. Individual Localization

The library KaryoploteR from the package biocLite of R was used to create individual retrotransposon localization images. An R script with two functions (localization, which makes a graph with the localization of the particular retrotransposon in its respective chromosome, and zoom, which creates a zoom of the first image), called **cariofunctions.R**, was designed and then it was connected to python and app.py using the rpy2 packages. The functions from cariofunctions.R take as arguments 4 variables: the chromosome where the retrotransposon is located, the start position of the retrotransposon, the end position and a name for the image (.png) file (see cariofunctions.R in the main directory).

Rpy2 is a packages which helps python to use functions and scripts from R. In this case, rpy2.robjects.packages, and specially the function STAP, allows to read the functions from cariofunctions.R and adds the variables that KaryoploteR needs to run. Since the "Individual page" (see app.py, section "App Website pages) calls from the RETROtransposon database the information of a specific retrotransposon, it´s possible to use its general data to run the R functions. The results are saved in a specific folder and then the images are called and displayed in the HTML Template "Individuals" (see templates folder).

2.2.4. Phylogenetic Trees

A visual representation for the relationships between the translated products for the retrotransposons was achieved via a phylogenetic tree. Phylogenetic trees will classify the retrotransposon families according to how closely related they are. The phylogenetic trees for both retrotransposon family products were created using the software MEGA7 (http://www.megasoftware.net/). This integrated software is useful to create phylogenetic trees from implementing sequence alignments and distance scores. Individual ORF/polyprotein FASTA files for LINE and ERV were uploaded into the software. The FASTA names for each sequence were identified by family name, followed by chromosome location (i.e. HERVK-int_chr8). The sequences can be viewed first once uploaded. The option to crop the sequence ends was selected. Trimming the sequence will ensure that indels are not included in the alignment. The ClustalW alignment was chosen out of ClustalW and MUSCLE to run multiple alignments on the sequences. ClustalW is dependent on the following: (i) computing pairwise distance, (ii) clustering analysis of the sequences and (iii) iterated alignment of most similar groups of sequences. The main difference between these tools is that ClustalW uses a progressive algorithm, whereas MUSCLE uses an iterative algorithm. However, ClustalW is good at handling gaps and using matrices to compute distances. The algorithm performs computes an estimate distant matrix between each sequence pair from the pairwise alignment scores [22]. The alignment had a gap opening penalty of 10, gap extension penalty of 0.2, Protein weight matrix: Gonnet and a divergent cut-off at 30%. These parameters were set by default by the program. Once the sequence alignment was completed, pair-wise distance was performed on these sequences. More distant groups of sequences are aligned until the global alignment is reached [22]. The Day-Hoff matrix was chosen as a parameter, along with bootstrap of 100, due to the large number of sequences.

The pair-wise distance was the only option in MEGA7, which evaluated the difference between the sequences, to generate a distance by combining these differences to create an estimated tree. Once the distance matrix is calculated, the tree can be built. The consensus option was chosen for the tree, as a consensus tree combines multiple trees with the same family name to produce the phylogenetic tree.

Independent trees were created for LINE (ORF1), and ERV (Gag, Pol, Env). One problem encountered with LINE was the size of the FASTA files. ORF0 had two sequences, whereas OFR2 had greater than 3000 sequences. The software is not able to generate trees for files of this sizes, and was noted to be very slow for alignment, in particular ORF2. One modification to the FASTA files could be that each sequence could be identified by only replication family name (i.e. HERVK) to get a more refined phylogenetic consensus tree. In particular, for the ORFs as there are many replication names for LINE-1, which could have resulted in a smaller tree.

3. REQUIEREMENTS AND INSTALATIONS

The following software packages are required to be downloaded and were run in the following order:

- PYTHON 2.7.14
- R 3.4.1
- FLASK (python framework)
- RPY2 2.7.0 (python package)
- PYTEOMICS (python package)
- BIOCLITE, KARYOPLOT, CROP (R libraries)

This software is written mainly in the programming language Python (version 2.7.14). The interface was built using external libraries: HTML, CSS and JQuery. The software will work with Ubuntu (version 16.04) or Linux with Mac OS (version 10.11 or higher). Use *pip install python* to download python from Linux/Ubuntu.

Flask will need to be installed in Python, preferable version 2.7.14, using *pip install flask* command. It is necessary to have the rpy2 package (version 2.7.0) installed (*pip install rpy2*) to embed the r codes into Python. This package is compatible with both Microsoft and Mac OS (however, can be troubling to install). Pyteomics was installed in Python, using the command *pip install pyteomics*.

The installation of KaryoploteR via Bioconductor source (version 3.7) is required in order to view certain visualizations (chromosome localizations and karyoplots). KaryoploteR works with Mac OS version 10.13.3 (High Sierra) and Ubuntu (version 10.16), both which are compatible with R version (>=3.4) (it's important to check the R version that you are installing since the compatibility in R is a complicated issue). The following will need to be entered to download this package: source ("https:// Bioconductor.org/biocLite.R), followed by biocLite ("karyoplotR") command in R.

The installation of crop in R is needed to obtain the correct sizes for the localization images, (as using HTML disables hover function). Use install.packages("crop"). Along with the crop package, sometimes is necessary to install the ImageMagick program. To install, use brew ImageMagick command (Linux) or source apt-get ImageMagick (Ubuntu).

To ensure the accurate installation, it is vital to copy the "Farfalloni" folder from Github to any folder of the computer that will be used. This is followed by running the python app.py command in Linux/Ubuntu. Check to see how to make the final database file, which will need to be copied to the "Farfalloni" folder. This will generate a http link (http://127.0.0.1:5000/), which will need to be copied into your browser bar to take the user to the software. The following will need to be

typed once link is copied as follows: http://127.0.0.1:5000/index.html (note: index.html will need to be entered to take users to homepage).

Make sure of not moving any file in a different directory (such as app.py, massapp.py or cariofunctions.R), to have the database file in the same folder of app.py and, if the MSRetroFinder tool is going to be used, have the mzTab or mzIdentML files in the same directory of app.py.

For all installation of packages, see the relevant documentation links.

How to create the database?

- Get the information (CSV file with general information and the sequence FASTA file) from Table Browser using the filter "L1" in repFamily from LINE-1 and "LTR" in repClass from HERV.
- Using the python algorithms HERV.py and L1.py from the folder "ORF Identifications" uploads the FASTA files – make sure to have the name of the FASTA files right or to change the names in the python files. Wait to get the results in CSV files with the translated products.
- After you have the translated products, it's time to combine the CSV file with the general information with the CSV file with the translated products. Since Table Browser generates the lists in the same order, you only have to copy the information from one CSV file to the other. Name the new columns, the one with the translated products, "ORF1", "ORF2" and "ORF3".

- Finally, we only keep the more important columns, so you have to delete some of them. The columns in the final files are: genoName, genoStart, genoEnd, strand, repName, repClass, repFamily, repStart, repEnd, sequence, ORF1, ORF2 and ORF3 (Figure 3).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	genoName	genoStart	genoEnd	strand	repName	repClass	repFamily	repStart	repEnd	sequence	ORF1	ORF2	ORF3	
2	chr1	159383130	159384355	+	MSTA-int	LTR	ERVL-MaLR	358	1595	TGGTGGAAG	Not ORF	Not ORF	Not ORF	
3	chr2	226492179	226492451	-	HUERS-P3b-i	LTR	ERV1	-1274	6144	GAATCATTTT	Not ORF	Not ORF	Not ORF	
4	chr3	6291167	6291531	+	MLT1C	LTR	ERVL-MaLR	59	426	ACTTCAGAG	MAAAAAAA	Not ORF	Not ORF	
5	chr4	26214342	26214429	-	MLT1K	LTR	ERVL-MaLR	0	595	ATTTTTAAA	Not ORF	MGGGRTTTR	MTYHHHHHHHHGGGFFF	
6	chrX	29359332	29360290	+	LTR5_Hs	LTR	ERVK	1	968	TGTGGGGAA	MESTEVAN	MAURA	MVITHASAA	
7														
8														

Figure 3. The CSV file for LINE-1 or HERV has to look like this before to make the SQLite database.

- With the CSV files for LINE-1 and HERV you can use DB Browser for SQLite to combine both CSV files and create the "final.db" database. The database is going to have to tables: one called "ERV" with the retrotransposon information and a second one called "Fill", which is the one that we are going to use to generate the MSRetroFinder Atlas.

You have to make the last table and it must have the columns “Amino” and “Tissue” (Figure 4).

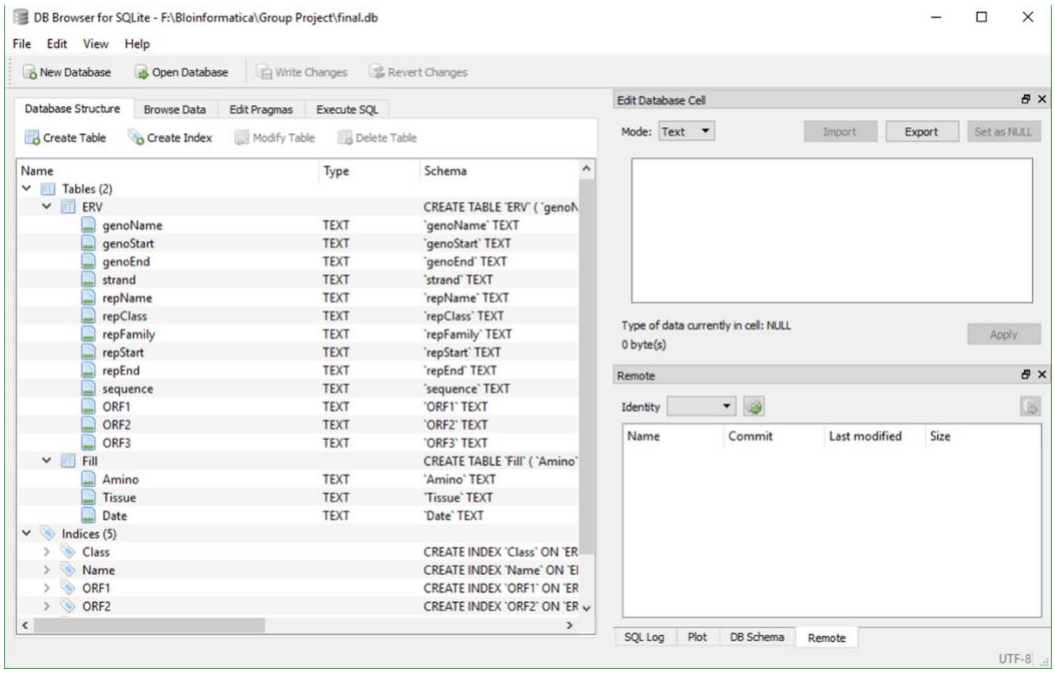


Figure 4. Using DB Browser for SQLite to make the “final.db” database with the tables for the RETROtransposon data and for the MSRetroFinder Atlas.

- Save the database going to “File” and then “Set Encryption” and copy it into the same folder that the main application app.py. Make sure that the database is called “final.db” and the tables are called “ERV” and “Fill”.

4. INTERFACE DESCRIPTION

HTML was used to create the templates of this website based application, a standard layout of head, body and footer was used throughout these files. The head contains external links to any fonts used (e.g. from Google fonts), CSS files and external jQuery script links. The footer is the GitHub octocat logo which links to the Farfalloni GitHub repository. The body of each page contains the navigation bar which is the same for each page, but depending on the page currently being used, changes slightly. What differs in this section depends on the content of each page, for example tables, paragraphs, images and any jQuery script would also be included here. Furthermore, interactive graphs that were made using an external website were displayed in the website using anchor tags inside of an unordered list to include the long external HTML links from plotly (see templates/database.html).

The HTML files are templates that are called in the app.py file to run the website on a local server. In a similar way, code can be called from within the HTML file to retrieve specific information from the database (see final.db). Taking the database page as an example, a loop was created to call each row and column of the database (in the app.py file), then in the HTML file another loop was made to call specific rows and input them into a table, e.g. <td>{{re[”genoName”]}}</td> takes the genome name from the database for each retrotransposon and puts them in a column of the table (see templates/database.html in the directory).

Another calling method used to connect the database to the HTML files was method=‘get’ whenever a input submission form was used. For example, in the search page, if a user only wants to see LINE-1 retrotransposons with ORF2 sequences, this method will search for elements with the name “ORF2” via the app.py file and return the relevant information to the HTML page. A similar process happens whenever a user uploads information to an input submit form on a HTML page.

A HTML file alone will make a functional webpage, however it will be very plain and have no design, therefore in order to add style and personalise the design of the actual interface of the webpages, various CSS files had to be included; a CSS file adds style to a HTML page.

The main.css file (found in the static folder) contains all they styling for RETROtransposons, for example the general styles of the website including: margins, website name, font size and colours, colours and alignment of the navigation bar, margins and positioning of tables and images.

To connect the styling CSS files with the HTML files, they were added in the <head> of HTML file in link tags (which allow the use of external files/urls within the file). As HTML files work in ascending order, the normalize.css file was added first as a default style, then the main.css file was added so those changes would override the standard layout, then lastly the jQuery.css files were added (when necessary).

CSS files needed to be included in the HTML <head> tag were:

normalize.css to provide default cross browser consistency styling, which was downloaded from GitHub (https://github.com/necolas/normalize.css/blob/master/normalize.css)

- main.css which was created manually from scratch, to add style to each section of the web interface: the header (website name and navigation bar) and footer were designed and added to each HTML page so they all have the same template to fill with content. The body of each page is what differs depending on the content, for example tables, text and graphs.
- jquery.dataTables.min.css was added in <script> tags in this section of the HTML pages that needed JavaScript code to carry out particular functions e.g. pagination, and having drop down menus that could not be done using HTML and CSS alone. (see templates/database.html).

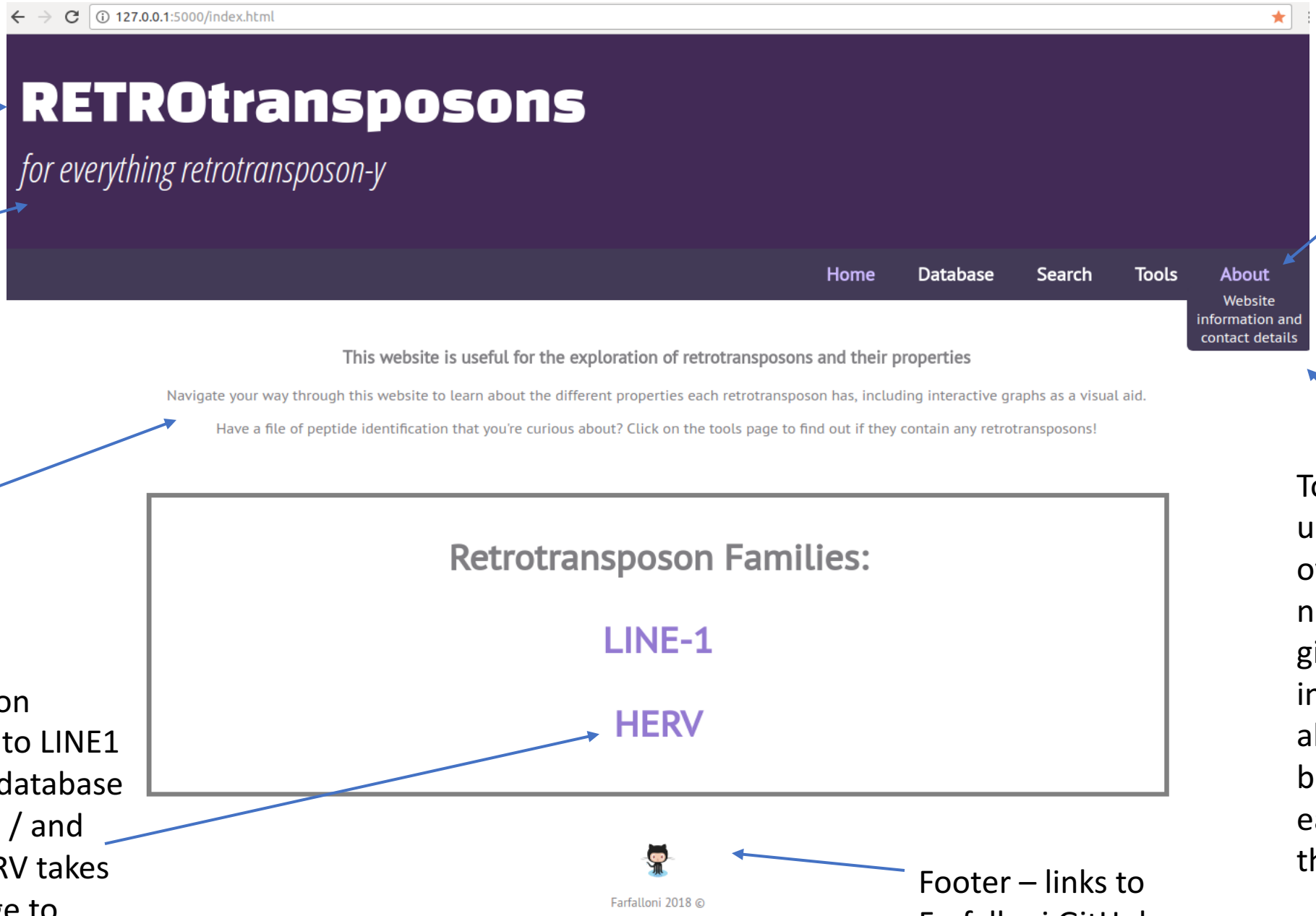
A flaw of using CSS and html alone was that pagination to the retrotransposon tables could not be implemented as the content of the table was data being called into the table from the database, thus losing functionality; to overcome this jQuery was used. jQuery is a JavaScript library that can be used for animation and manipulation of content in html files; an extremely useful language but more difficult to get the grasp of due to the short time period. All js needs to be in <script> tags to function correctly, in addition, pages that require JavaScript need to include jQuery libraries in script tags in the head section of the HTML document:

- (needed for all HTML pages that use JavaScript)
  http://cdn.datatables.net.1.10/16/js/jquery.dataTables.min.js
- (needed for HTML pages that use pagination in tables)
  https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js

On occasion, updating the style of a certain id or class in the main.css file would not work for some reason, despite trying many different solution. To overcome this style tags were added in some HTML documents to add specific CSS code inside of them.

All the above components are required in order to make RETROtransposons a fully functioning website.

The next few pages show the web interface, explained:



Website name

Website slogan

Navigation bar – links to different pages of the website

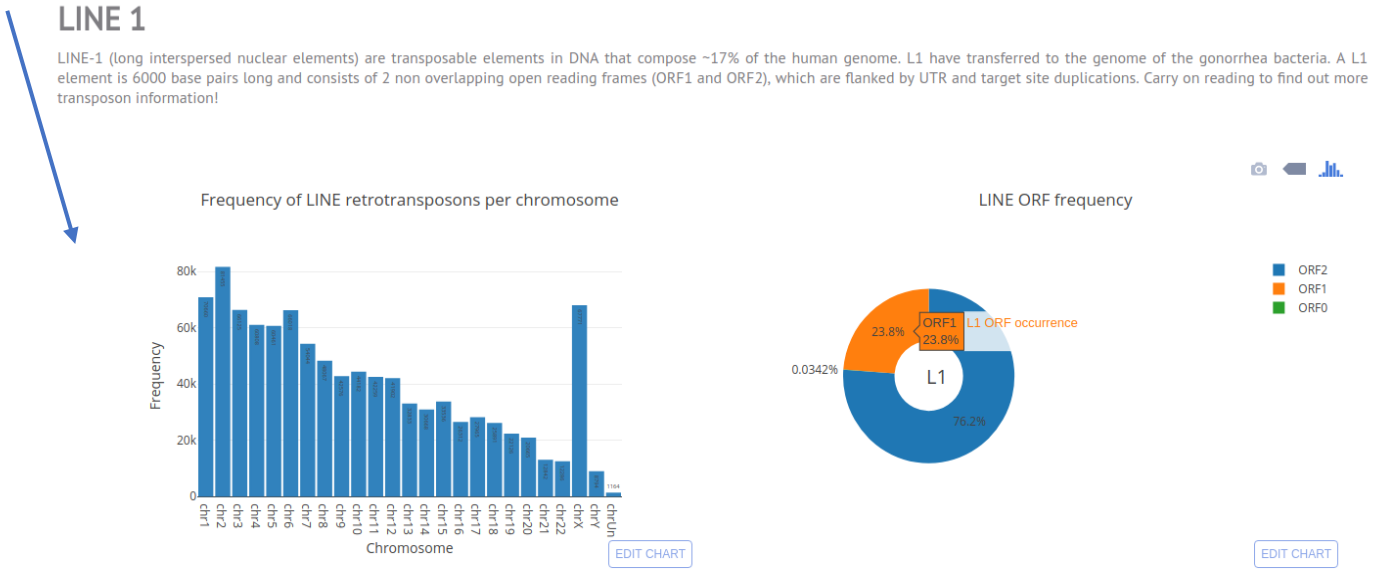
Website introductory paragraph

Tooltips appear upon hovering over links in navigation bar, gives user information about what can be found on each page of the website

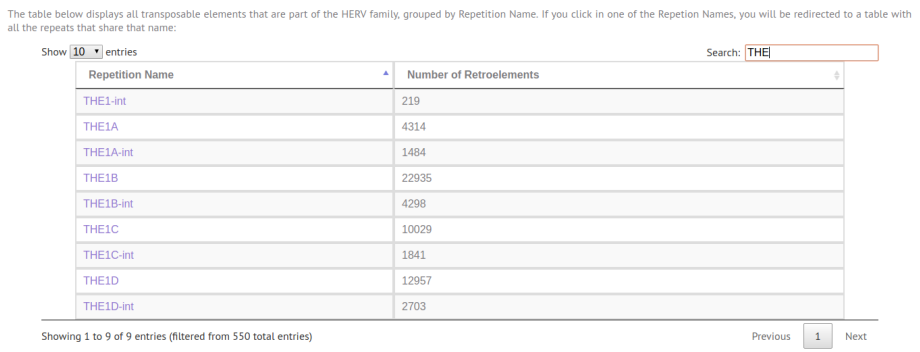
Retrotransposon families - links to LINE1 information – database and visual aids / and clicking on HERV takes user down page to HERV information

Footer – links to Farfalloni GitHub repository

Users can click on the database page or the retrotransposon families on the home page to see this information. Brief explanation of what LINE1 are and interactive (user can hover over graph for exact frequencies) graphs showing the frequency of retrotransposon elements per chromosome and ORF frequency, respectively

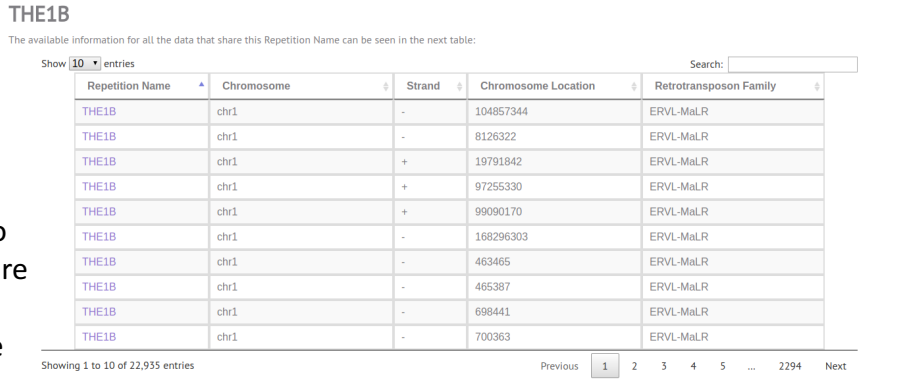


Below the various graphs is a table of retrotransposon elements, grouped by the same repetition name (as many have the same name but span different locations on the chromosome), for ease of use the number of elements is displayed next to each. The table is interactive and lets user search for a specific repetition name if known, sort them as/descending and can show 10,20,30 etc. entries per page.



Drop down menu for user to view various other graphical representations of the retrotransposons found in the database, e.g. the frequency of retrotransposons per chromosome along the length of each (as shown), in addition to phylogenetic trees which show the relationship between family members

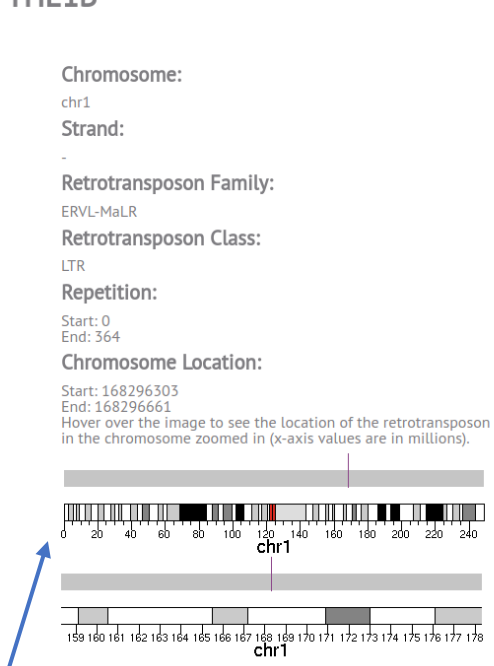
After selecting a specific element, users are taken to another table with all the retrotransposons that share the same name, displaying more useful information about the elements – strand, location, chromosome number.



Selecting a desired repetition name will take the user to another page, which displays all relevant information (as above) as well as its DNA and amino acid (if applicable) sequences.

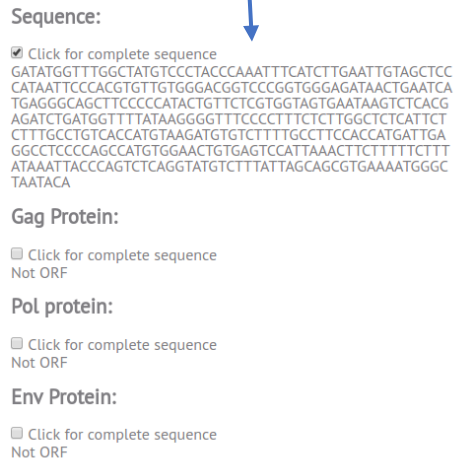
NB: information for both LINE1 and HERV retrotransposons are shown in a similar layout

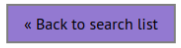
Information page for each specific element is produced, conveniently showing all the information in the database.



An image is produced on demand for each retrotransposon element, showing the location of the element on the chromosome, which upon hovering will show a zoomed in version of the location

DNA and amino acid sequences are also displayed, initially showing only one line of sequence and if the user wants to see the complete sequence they can select the checkbox





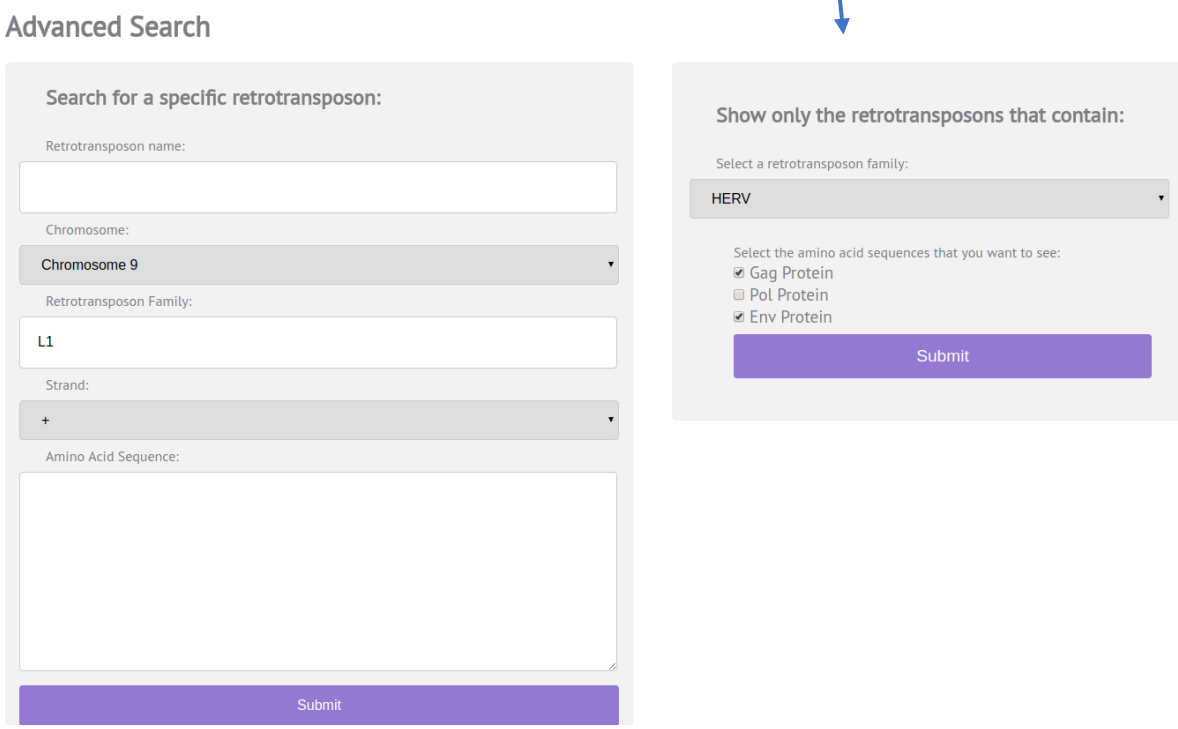
Takes users back one page to the list of retrotransposon elements generated from their query

LOCATION UNKNOWN

There are some chromosomes that are unknown but have retrotransposon elements (scientists are still researching their meaning), if a user selects to view these chromosomes the above text is generated as the location is not yet known.

A new page is generated for each retrotransposon element selected from the tables in previous pages, or from the advanced search page in which users can search for retrotransposons using specific bits of information. For example they can select from a dropdown menu of chromosomes, the strand (reverse – or forward +), they can also search using an amino acid sequence, which will search for a match in the database and if found, will generate a new information page (shown left).

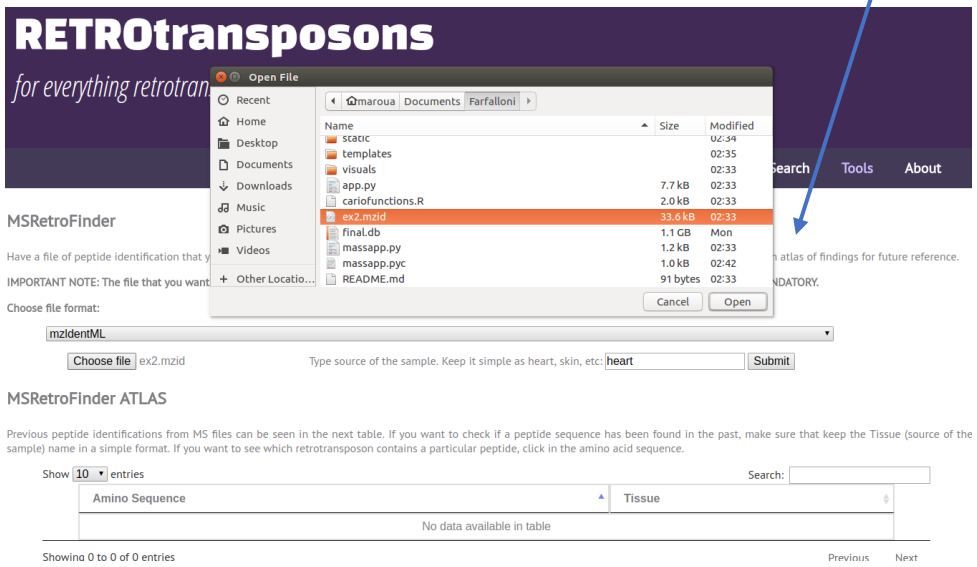
Users also have the option to search specifically for retrotransposons that contain certain amino acid sequences; they can select either family LINE1 or HERV and a checkbox list will appear that allows for selection.



Query not found

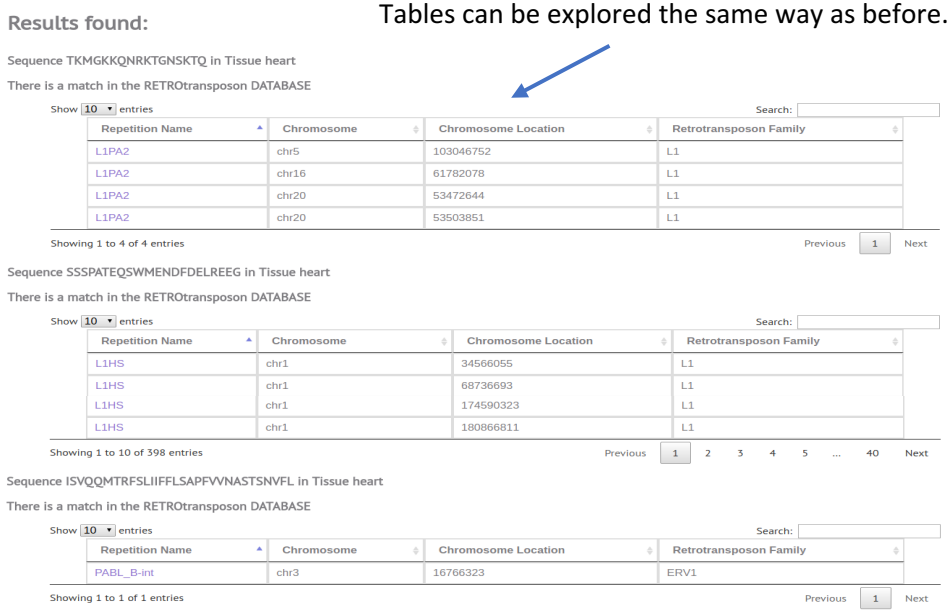
If user searches for a retrotransposon element that does not exist e.g. selects all three ORFs (1,2,0) for a LINE1 query, or enters an invalid amino acid sequence, then a new page will open with this message.

Here, the user can upload a file of peptide identification in mzTAB/mzID format using the 'choose file' button, which opens the user's directory and enter the tissue source of the sample.

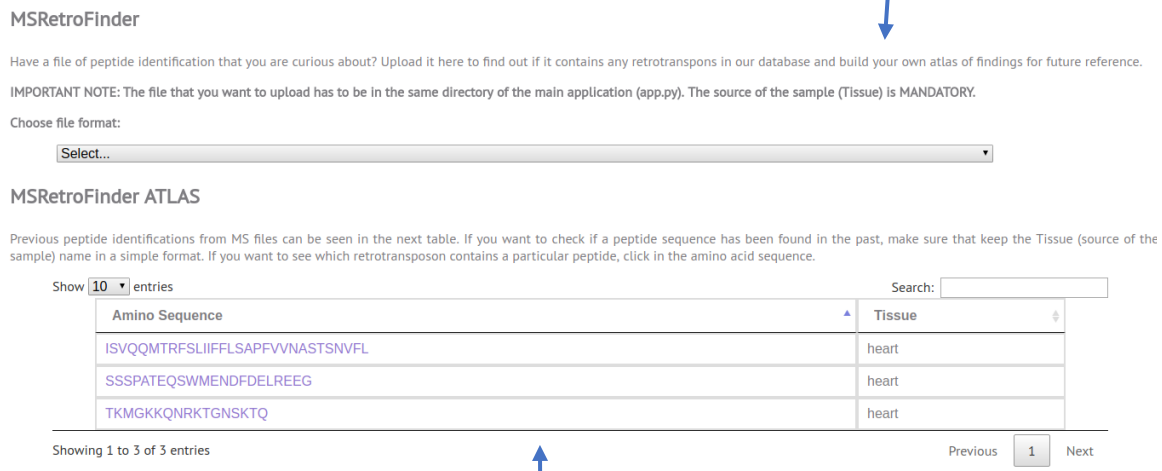


Using the uploaded file, this tool will search for sequence matches to the database and generate any results found. As shown below there were three sequences with matches, displaying the retrotransposon elements that share the same sequence.

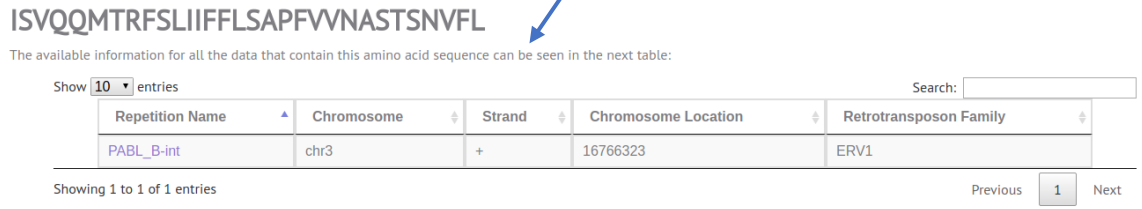
Tables can be explored the same way as before.



Each time a user uploads a file and matches are found, they are stored within this web application and presented as a retrotransposon expression atlas. Whereby the next time a user is on this tools page, their previous matches will be shown in a table (as below), and the tool will search first to see if there are already matches in the atlas; if there are then they will be presented, if not a new search will begin



Clicking on any of the amino acid sequences in the atlas will take the user to a new page and present a table of retrotransposon elements that share the same sequence(s), which can again be navigated to open a new page with specific retrotransposon information



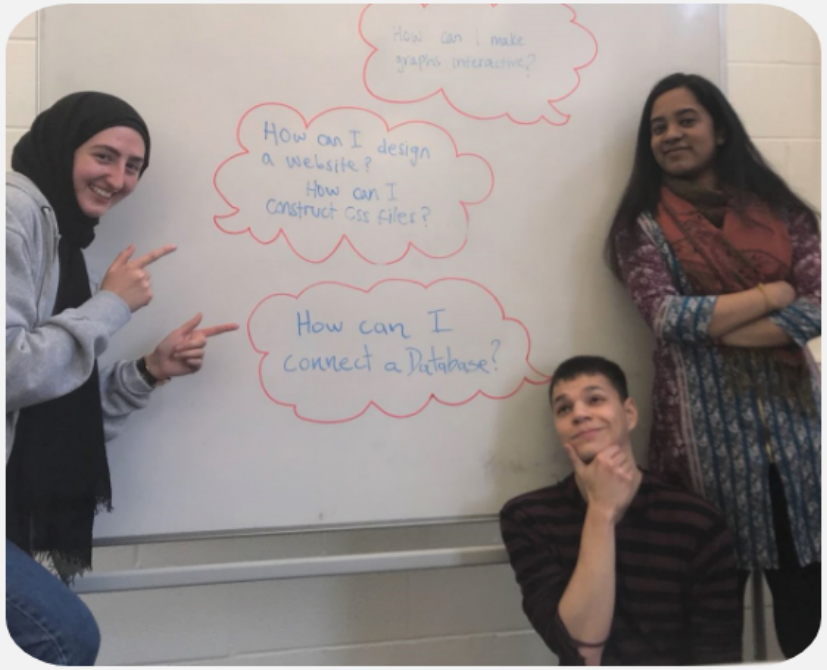
If no matches to the database from the uploaded file are found this message will open on a new page

About page split into contact and website information

Information about the creators of RETROtransposons and an image

About us

We are a group of MSc Bioinformatics students at Queen Mary University of London (Mile End) who have developed this website as a means of exploring retrotransposons. Enjoy this cute quirky picture of team Farfalloni.



Farfalloni contact details. If on a mobile device, a 'call this number' pop up will appear. Clicking on the mail link will open and email directly to Farfalloni. Lastly a link to the Farfalloni GitHub repository.

Contact us

Feel free to reach out to us on any of these platforms and check out our GitHub repository:

0208-745-1634
farfalloni@hotmail.co.uk
https://github.com/eagomezc/Farfalloni

Website information lets user know what this website can be used for and about data collection

Website information

This website is useful for the exploration of retrotransposons and their properties. Navigate your way through this website to learn about the different properties each retrotransposon has, including interactive graphs as a visual aid. Have a file of peptide identification that you're curious about? Click on the tools page to find out if they contain any retrotransposons! The database was generated using information from the UCSC Table Browser and tabulated using sqlite3 (both linked below).

Glossary

Retrotransposon: a transposon whose sequence shows homology with that of a retrovirus.

Open reading frame (ORF): a continuous stretch of codons that contain a start codon (usually AUG) and a stop codon (usually UAA, UAG or UGA).

Line-1: are transposable elements in the DNA of some organisms and belong to the group of Long interspersed nuclear elements (LINEs).

HERV: Endogenous retroviruses are the most important LTR retrotransposons in mammals, including humans where the Human ERVs make up 8% of the genome.

Farfalloni: Pasta in the shape of a large butterfly or bowtie.

Some useful key words used in the website and their meanings, which enables users to fully understand the scientific language used – allowing even users with basic scientific understanding to enjoy this website and its functions

What's next for RETROtransposons?

This website is currently a prototype however we will be adding more useful features and tools in the future; any suggestions are welcome.

Informs users of future development of RETROtransposons

Useful links

- Example mzTab/mzID files
- Plotly for interactive graphs
- UCSC for raw data

Links to example mass spectrometry files for users to use on the tools page, as well as links to the raw data and graph construction websites.

5. PERSPECTIVES

- Security: the final database doesn't have any encryption or security parameters that can prevent it from getting corrupted. Applies some security measures (creates passwords and security keys) and uses best practice methods in database development can make the database more secure.
- Even after applying indexes, that improved the speed of some functionalities of the database by 30 seconds, loading the database and using MSRetroFinder is slow. The use of strong database environments such as MySQL or Oracle, and upload the website application onto a server could be good solutions to improve the speed of the database.
- It's expected to upload new retrotransposon families in the RETROtransposon database.
- Make better phylogenetic analysis to obtain more information about the relationship between the different members of a retrotransposon family.
- A future functionality that can be really helpful is the possibility of download the particular information that a user wants in plain text files such as CSV and FASTA.
- For good practice, it's expected to improve the hierarchy of the software main directory. App.py needs the python modules and the R script in the same directory to be able to use them, which it's not the best way to present a software.
- Another issue with the directory is that the application needs to have the mzTab or mzIdentML file that the user wants to use in the same directory of app.py. Without this condition MSRetroFinder crashes when someone use it. There are ways to do it with the request function and the methods GET and POST, but it's necessary to read more about the topic.

6. BIBLIOGRAPHY

1. Cordaux, R., & Batzer, M. (2009). The impact of retrotransposons on human genome evolution. *Nature Reviews Genetics*, *10*(10), 691-703. http://dx.doi.org/10.1038/nrg2640
2. Goodier, J. (2016). Restricting retrotransposons: a review. *Mobile DNA*, *7*(1). http://dx.doi.org/10.1186/s13100-016-0070-z
3. Villesen, P., Aagaard, L., Wiuf, C., & Pedersen, F. (2004). *Retrovirology*, *1*(1), 32. http://dx.doi.org/10.1186/1742-4690-1-32
4. Boissinot, S., Roos, C., & Furano, A. (2004). Different Rates of LINE-1 (L1) Retrotransposon Amplification and Evolution in New World Monkeys. *Journal Of Molecular Evolution*, *58*(1), 122-130. http://dx.doi.org/10.1007/s00239-003-2539-x
5. Singer, M. (1982). SINEs and LINEs: Highly repeated short and long interspersed sequences in mammalian genomes. *Cell*, *28*(3), 433-434. http://dx.doi.org/10.1016/0092-8674(82)90194-5
6. Brouha, B., Schustak, J., Badge, R., Lutz-Prigge, S., Farley, A., Moran, J., & Kazazian, H. (2003). Hot L1s account for the bulk of retrotransposition in the human

population. *Proceedings Of The National Academy Of Sciences*, *100*(9), 5280-5285. http://dx.doi.org/10.1073/pnas.0831042100

7. Penzkofer, T., Jäger, M., Figlerowicz, M., Badge, R., Mundlos, S., Robinson, P., & Zemojtel, T. (2016). L1Base 2: more retrotransposition-active LINE-1s, more mammalian genomes. *Nucleic Acids Research*, *45*(D1), D68-D73. http://dx.doi.org/10.1093/nar/gkw925
8. Paces, J. (2002). HERVd: database of human endogenous retroviruses. *Nucleic Acids Research*, *30*(1), 205-206. http://dx.doi.org/10.1093/nar/30.1.205
9. Wang, J., Song, L., Grover, D., Azrak, S., Batzer, M., & Liang, P. (2006). dbRIP: A highly integrated database of retrotransposon insertion polymorphisms in humans. *Human Mutation*, *27*(4), 323-329. http://dx.doi.org/10.1002/humu.20307
10. Dai, L., LaCava, J., Taylor, M., & Boeke, J. (2014). Expression and detection of LINE-1 ORF-encoded proteins. *Mobile Genetic Elements*, *4*(3), e29319. http://dx.doi.org/10.4161/mge.29319
11. Martin, S. (2006). The ORF1 Protein Encoded by LINE-1: Structure and Function During L1 Retrotransposition. *Journal Of Biomedicine And Biotechnology*, *2006*, 1-6. http://dx.doi.org/10.1155/jbb/2006/45621
12. Denli, A., Narvaiza, I., Kerman, B., Pena, M., Benner, C., & Marchetto, M. et al. (2015). Primate-Specific ORF0 Contributes to Retrotransposon-Mediated Diversity. *Cell*, *163*(3), 583-593. http://dx.doi.org/10.1016/j.cell.2015.09.025
13. Januszyk, K., Li, P., Villareal, V., Branciforte, D., Wu, H., & Xie, Y. et al. (2007). Identification and Solution Structure of a Highly Conserved C-terminal Domain within ORF1p Required for Retrotransposition of Long Interspersed Nuclear Element-1. *Journal Of Biological Chemistry*, *282*(34), 24893-24904. http://dx.doi.org/10.1074/jbc.m702023200
14. Kines, K., Sokolowski, M., deHaro, D., Christian, C., Baddoo, M., Smither, M., & Belancio, V. (2016). The endonuclease domain of the LINE-1 ORF2 protein can tolerate multiple mutations. *Mobile DNA*, *7*(1). http://dx.doi.org/10.1186/s13100-016-0064-x
15. Villesen, P., Aagaard, L., Wiuf, C., & Pedersen, F. (2004). Identification of endogenous retroviral reading frames in the human genome. *Retrovirology*, *1*(1), 32. http://dx.doi.org/10.1186/1742-4690-1-32
16. Coffin, J., Hughes, S. and Varmus, H. (1999). *Retroviruses*. Plymouth: Cold Spring Harbor Laboratory Press.
17. Benit, L., Dessen, P., & Heidmann, T. (2001). Identification, Phylogeny, and Evolution of Retroviral Elements Based on Their Envelope Genes. *Journal Of Virology*, *75*(23), 11709-11719. http://dx.doi.org/10.1128/jvi.75.23.11709-11719.2001
18. EMBL-EBI, I. (2018). *Zinc finger, CCHC-type (IPR001878) < InterPro < EMBL-EBI*. *Ebi.ac.uk*. Retrieved 14 February 2018, from https://www.ebi.ac.uk/interpro/entry/IPR001878
19. Jones, A., Eisenacher, M., Mayer, G., Kohlbacher, O., Siepen, J., Hubbard, S., Selley, J., Searle, B., Shofstahl, J., Seymour, S., Julian, R., Binz, P., Deutsch, E., Hermjakob, H.,

Reisinger, F., Griss, J., Vizcaíno, J., Chambers, M., Pizarro, A. and Creasy, D. (2012). The mzIdentML Data Standard for Mass Spectrometry-Based Proteomics Results. *Molecular & Cellular Proteomics*, 11(7), pp.M111.014381.

20. Psidev.info. (2018). *The twenty minute guide to mzTab*. [online] Available at: http://www.psidev.info/sites/default/files/v_1_0_0_20minute_guide_mzTab.pdf [Accessed 14 Feb. 2018].

21. Anon, 2018. Package 'KaryoplotR'[ebook]. Available at: https://bioconductor.statistik.tu-dortmund.de/packages/3.5/bioc/manuals/karyoploteR/man/karyoploteR.pdf [Accessed 15 February 2018].

22. Thompson, J.D., et al, 1994. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice [e-journal]. Available at:https://www.ncbi.nlm.nih.gov/pubmed/7984417 [Accessed on 14 February 2018]