

Software documentation

Team Melon

Thursday the 16th of February, 2017

Modupeh Betts

Andrew Knowles

Nadim Rahman

Madeline Rhodes

Table of contents

Introduction.....	
Summary.....	
Intended audience.....	
Software requirements.....	
General information(??).....	
Official software specification.....	
Scientific background to the task.....	
Summary of our software package.....	

User guide.....

- Introduction
- Required file types and formats
- Parameters of the BLAST search
- Output of BLAST

Statistical analysis

- Analysis in R
- Volcano Plots
- PCA
- Output of R

Mechanics of the software.....

- Required installations and packages
- Summary
- Software architecture
- Folder-structure?
- Flow-chart?
- Flask?
- Blast?
- R?

Limitations.....

Opportunities for future development.....

Conclusion?

References

Introduction

Summary

This document is the guide for a prototype of an online bioinformatics web-app that was developed for a group project undertaken as part of a masters course. The official title of the project was: '*Software Development Group Project 2017*'. It contains information on what the program does, how it should be used and on its limitations and priorities for future development.

Intended audience

This web-app is designed to be simple to use and could theoretically be deployed as a public website, accessible to anyone with internet connection. Therefore this guide is intended for any researcher with basic knowledge in RNA-seq technology who might want to use the online program.

General information

University: Queen Mary University of London

Masters programme: *MSc Bioinformatics*

Time period for the project: 10/01/2017 - 17/02/2017

Team members: Andrew Knowles, Madeline Rhodes, Modupeh Betts, Nadim Rahman

Official software specification

The specification that we were tasked with addressing by the examiners is listed *ad verbatim* below:

“The software should allow the user to upload a set of FASTA files containing the sequences of *de novo* assembled transcripts and their FPKM values from multiple samples (details of the format are provided below in the sample data section). It will then allow the user to:

1. Infer which genes each transcript relates to (if any) by sequence homology.
2. Identify transcripts that have significantly different expression between different samples.
3. Explore the expression data graphically. Visualisations might include principal components analysis, volcano plots, and hierarchical cluster analysis.

The software is expected to be a working prototype, which is to say that it will provide a working demonstration of the functionality described above but would need to be passed to professional developers/web designers to turn into a fully polished web application. Documentation should therefore be provided, both within the program code and in a dedicated document, to explain how the software is structured and how it works.”

Scientific background to the software

The cost of sequencing has declined by several orders of magnitude since the turn of the millennium ([do reference](#)). Consequently there has been a rapid growth in use of high-throughput sequencing technologies leading to a transformation in the way many fields of biological research are conducted and to the inception of entirely new research areas altogether. The increasing use of these technologies have led to an exponential increase in the amount of data-generation and with the sizes of typical data-sets being generated by labs growing concomitantly ([do reference](#)).

These developments represent a paradigm shift for the biological sciences. They pose major opportunities and challenges for the bioinformaticians who are tasked with the development of algorithms and software packages which can analyse these mountains of data in a statistically and scientifically relevant way.

One of the ‘omics’ technologies which has benefited greatly from these technological leaps is transcriptomics, which is a field which studies the transcripts present in a cell/tissue- typically comparing different developmental stages of physiological conditions ([do reference](#)). Transcriptomics has particularly benefited from the development of RNA-Seq, a method which provides a precise measurement of levels of transcripts and their isoforms ([do reference](#)).

Where our 'Melonomics' fits in

The type of data that our prototype software is designed to be able to analyse is multiple *de novo* assembled RNA-seq transcripts across different experimental states. Transcriptomics using RNA-seq data allows the detailed study of species which do not have a relevant sequence genome available ([do reference](#)).

With RNA-seq data, gene expression can be captured at particular time-points and in certain biological contexts- in other words, the differential expression of genes can be compared across two or more biological conditions ([do reference](#)). Studying differentially expressed genes can reveal crucial information about the functioning of diseases such as [cancer](#) or how the progression of a viral infection can affect the [gene expression of the host](#).

As stated in the software specification, the user should be able to upload FASTA files with RNA-seq, *de novo* assembled transcripts with the FPKM values for each sequence. The FPKM values give an indication of the sequencing depth of the assembled sequences and it stands for 'expected fragments per kilobase of transcript per million fragments mapped' ([do reference](#)).

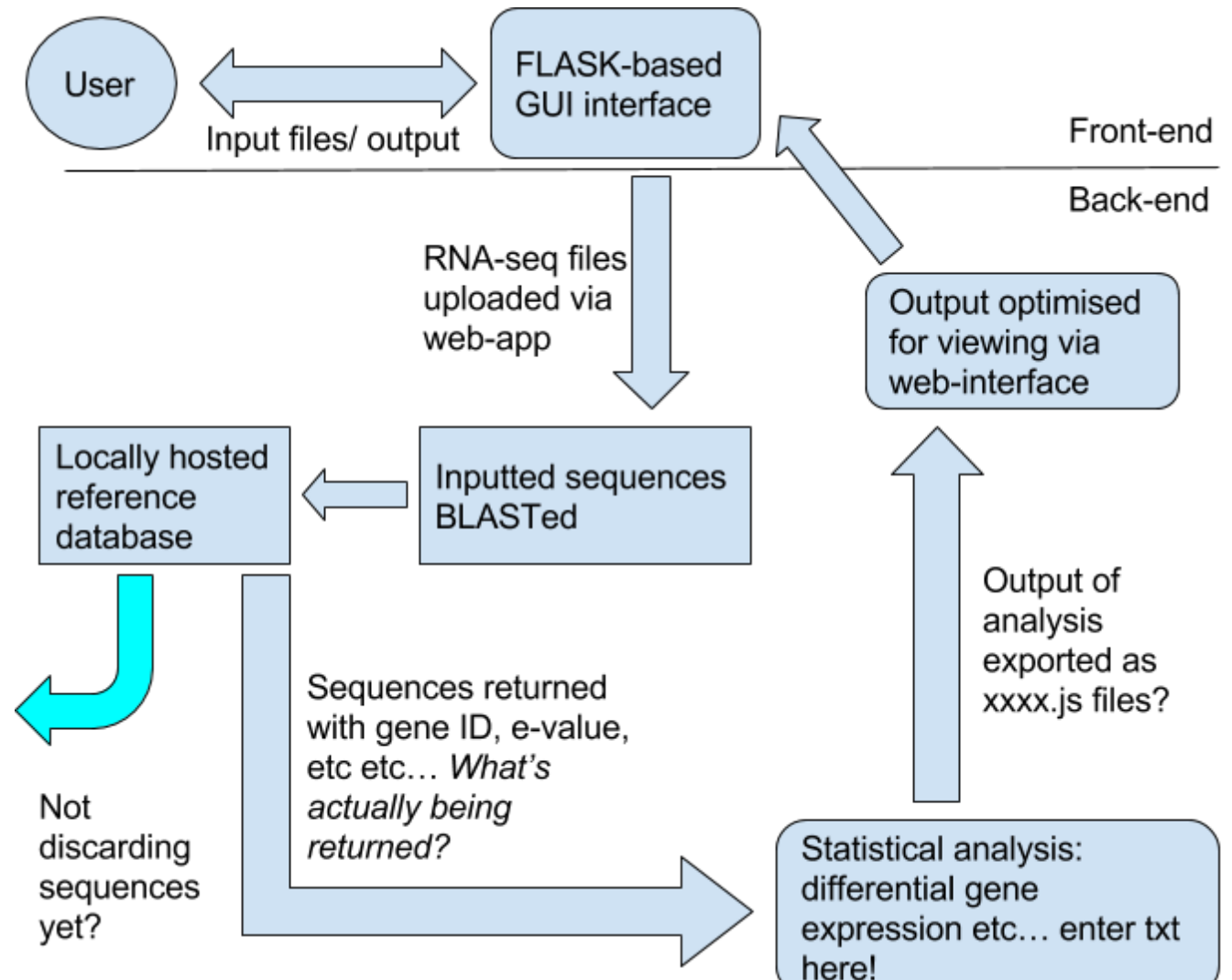
The software is optimised for analysis and comparison of RNA-seq data-sets across two or more different biological conditions; the output of the analysis is focused around the differential expression of genes according to the biological context of the sample.

Summary of our software package

The package that we have developed allows a user to upload two FASTA files (?) with data from *de novo* assembled RNA-seq experimental data. These two files then get 'blasted' against a locally hosted reference genome and then the top gene match for each transcript is returned, along with other parameters for the hits. Ambiguous matches are discarded.







The data from the BLAST is then passed onto analysis which takes place in R, which primarily examines the differential gene expression across the two sets of experimental data. The output from this is then passed back to the user for detailed analysis.

Software Architecture



Software Structure- do this when finished?

Software Structure

-  **data**
The data directory is the directory that stores the individual marcxml records generated by the import scripts.
-  **import**
The import directory contains the import scripts to import the marcxml data. There are 2 scripts, one for the initial import and the second for the nightly syncing.
-  **solr**
The solr directory contains 2 open-source packages already pre-configured for the ViewFind application. Apache Tomcat is the widely adopted Java Servlet Engine that runs Apache Solr. To start or stop solr, use the solr/tomcat/bin/startup.sh and solr/tomcat/bin/shutdown.sh scripts respectively.
-  **web**
The web directory holds the web-base front end component of the ViewFind application.
 -  **conf**
The conf directory contains the config.ini file which is the global configuration file for the application.
 -  **css**
The css directory holds the css files that are related to the HTML templates.

User Guide

Introduction

Required File Types and Formats

The software is optimised to analyse multiple files of *de novo* assembled RNA-sequences with their FPKM values derived from an experimental design focussed on differential gene expression. However, some of the functionality works with just one input file. Therefore, one or more files containing RNA-sequences are required.

Below is an example of the format of each file required; the number on the right is the FPKM value for each sequence and the letters/numbers on the left represent an arbitrary identifier for each sequence which come with the transcriptome assemblies. The identifiers do not have to be common across the inputted files.

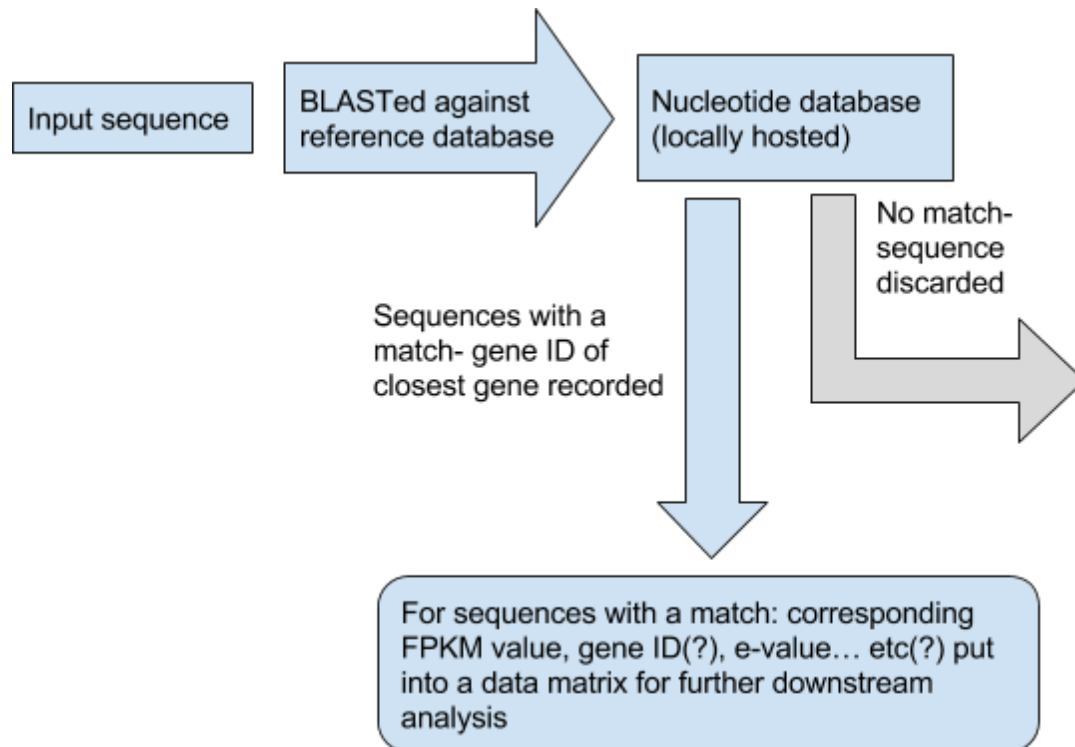
```
>asmb1_104;2.95
GCTGGGGCCTTCATCTTCCCGAGGGCGGTGAGTGCCCTG
CTGGCAGGGCCAGC
>asmb1_105;9.89
AGGGTCTCAACCATTAATCTCCAATCTTGATATCTATCCCC
ATCCCTTTCTCCCTACACCCGGGGCGGCATCTCAGGCAG
CAATTTGACTCCTAGCA
>asmb1_106;2.95
CGAGGGCGGTGAGTGCCCTGCTGGCAGGGCCAGCTCTC
CAATCTTGATATCTATCCCCATCCCTTTCTCCCTACACCCG
GGGCG
```

Figure 1.1(??) An example of the required format of the sequences in each input file; each sequence is associated with an identifier and FPKM value

Parameters of the BLAST Search

The standard BLAST of the software queries sequences (in the specified format, section 2.2) against a **locally hosted (right?)** nucleotide database (REFERENCE) to identify matches between the inputted RNA-sequences and genes in the reference genome. For each sequence present within a fasta file, a top hit is returned, depending on e-value and % identity match. An e-value threshold of 0.001 is used to determine more specific gene matches. Sequences which do not match to any genes in the reference database are discarded from the downstream analysis.

Keeping only the closest BLAST-match for each inputted sequence enables a more seamless and uncluttered analysis using downstream R-packages. A tabular output file of results is produced for each sample (or fasta file), named specific to the type of sample. All other BLAST parameters follow the default setting, found at REFERENCE.



Output of BLAST

The BLAST output produces tables of query matches in the following format:

*query_ID; subject_ID; %_Identity; alignment_length; mismatches; gap_opens;
query_Start; query_End; evalue; bit_score*

The number of queries is dependant on the number of sequences within each fasta file. As stated in section 2.3, the top match and its information is obtained. This output is not very informative for analysis, thereby making way for further processing of the output.

Post BLAST processing is required to meet a crucial aim of the software in analysis of samples. Once each sample is uploaded and BLASTed, the gene IDs corresponding to sequences in each sample file are parsed, along with associated expression values e.g. FPKM. Therefore each gene ID has a specific FPKM value (appended to a dictionary) and is further sorted depending on the sample name or type. This results in a matrix, with a dictionary of gene IDs and FPKM values within a dictionary for samples. The matrix is generated through python Pandas and consequently transferred and saved as a CSV file. This is fed into R for analysis.

Mechanics of the Software

Required Installations and Packages

BLAST

The BLAST function requires BLAST+ (Camacho *et al*, 2008) to be installed in order to run a BLAST query. Installers can be downloaded from: https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download. Secondly, Biopython (Cock *et al*, 2009), downloadable from <http://biopython.org/wiki/Download>, enables for command line BLAST to be carried out. This was utilised to avoid sending a vast number of queries to the BLAST website. Furthermore, local (standalone) BLAST reduces time taken to produce output, particularly in comparison with the online alternative. However it is also worth noting that the time taken for local BLAST is dependent on computing power. Thirdly, standalone BLAST requires a database to be downloaded, as this would be utilised to query against. The nucleotide database applied in the software can be found at the NCBI ftp site: <ftp://ftp.ncbi.nlm.nih.gov/blast/db/FASTA/>. Once downloaded, the database must be formatted specific to nucleotides. The database must be kept up to date, dependent on any new NCBI releases.

To run any Biopython packages, they must be brought into the python environment, this may require importing sys [REFERENCE] and appending the biopython packages to the path. The NcbiblastnCommandline [REFERENCE] from biopython enabled for the BLAST to be constructed on the command line or run as part of a function in a script. Alternatively the blastall [REFERENCE] command carries out the same function with the same results, however this does not support usage through a script. Furthermore with NcbiblastnCommandline, the package allows for multiple fasta files to be passed through with output generated.

Packages for post BLAST processing include re [REFERENCE] to parse out the gene IDs and FPKM values from BLAST output. Regular expression was utilised due to noticeably specific patterns noted for the parsed objects. An alternative would be to produce the output of BLAST in XML and use Biopython's XML parser to obtain gene IDs. However this was not implemented due to issues obtaining the query ID, required to obtain the FPKM values. It may still be possible to use this method in the future. A second package, csv [REFERENCE], enabled for the matrix produced to be written to a csv file. This was utilised to provide an appropriate output file type for R analysis. Furthermore, its inclusion resulted in code being more concise. The final, most crucial package utilised in this section, was python Pandas (McKinney, 2010). Similar to csv, its usage resulted in extremely concise code, which was easier to

follow. The package was utilised to produce a matrix of gene IDs, FPKM values and sample names for comparative analysis.

Software Components

BLAST

Biopython's NcbiblastnCommandline consists of standard parameters utilised within the current code. The constructed command is printed (in a slightly different format) for each sample passed through. Query or -i refers to the input fasta file and db or -d is used for the type of database, which has been set to 'nt' for nucleotides. The output file (out or -o) specifies the name of the BLAST output file and outfmt or -m specifies the format of the output, with 5 being tabular, displayed simply in a text file. An e-value threshold has been set to 0.001 and the number of matches (max_target_seqs) is set to 1, showing the top match for each sequence. Finally to reduce the time taken to run queries, -num_threads was set to 4. This increases the number of CPUs used from the default 1, to 4.

Focus for Future Development of the Software

Online deployment

The ultimate goal would be to have the software deployed online, accessible to anyone with an internet connection. Currently the prototype of the software can only be run on a computer with all of the necessary files and packages installed- such as the reference database and the R-packages necessary for the downstream analysis. Ultimately this would require the purchase of a domain name and the optimisation of a server to accommodate the multiple packages, data-files and scripts necessary for the running of the programme.

Allowance for multiple users

Currently only one user can use the software at a time. So one priority would be the scaling up of the software so that it can be used by multiple users simultaneously. The features that we would aim to incorporate are as follows:

- A sign-up page which allows a user to create an account or sign in through Facebook/Google +
 - Users creating an account locally and not through an external social networking site would require the creation of a secure data-base holding passwords and user log-in details, along with features allowing passwords to be reset- for example if a user has forgotten their password
- Ability for users to retrieve past data-sets of previous analyses and upload new files for new analysis
 - Since the data-sets that will be being analysed will typically be very large, a limit to the amount of the server's storage space might have to be set per individual user

One possible approach to this problem would be to utilise SQLAlchemy (REFERENCE) to keep track of server traffic. The web page would have to be adapted to include a login page, changing of details and potentially even a user profile page. This can be generated as separate scripts imported into the flask framework (e.g. login.py). An alternative option with regards to a login feature was to just introduce a credentials section prior to uploading fasta files. Credentials include the individual's name or email address and the title of the job sent to the user. This would still require a database, most likely through SQLAlchemy to save the individual's name or email address, job title and ID.

Perhaps instead of describing the features in one block of text we should use bullet points? Or combine the use of both?

Advanced BLAST settings

Extensive search parameters should ideally be added to the BLAST function of the software. In its current state, the input files are BLASTed against a reference database but all of the parameters are predetermined- **for example what the cut off point should be in regards to the E-value of the returned sequences**. Several options should be added to the BLAST functionality:

- The option to choose the type of database to BLAST against
 - The hosting of multiple reference databases may take up a huge amount of space on the server, therefore it might perhaps be more suitable for users' to be able to upload their own reference databases for temporary use
- **The ability to change the threshold of the match scores for the sequences(?)**
-

The user should be given an option to choose the type of database to BLAST against. Furthermore, e-value and number of gene IDs returned per sequence would provide more information to a user regarding their query sequences. To implement this, separate functions could be utilised (for different databases) in python scripts and called upon depending on the parameters set by the user. In terms of flask, this would be relatively simple to implement, however this includes inefficient and repetitive BLAST code. Alternatively the BLAST function can be adapted by passing all parameters through it. So instead of only passing the file name through the function, the e-value threshold, database name and number of top hits can also be included in the function brackets.

Further downstream analysis

Currently, the downstream statistical analysis consists of a number of tests: a Volcano plot comparing x and x, a PCA plot.... Etc....

Further tests should be added such as....

Improving the security

When implementing further settings, e.g. login feature, data protection and security become a more prominent issue. For example, a database for login features would

contain names or email addresses and passwords. To overcome this problem SQLAlchemy can be adapted with packages to encrypt information, providing some protection against a security breach.

Contact

Modupeh Betts..... email :

Andrew Knowles..... email :

Nadim Rahman..... email :

Madeline Rhodes..... email :

References

Camacho C., Coulouris G., Avagyan V., Ma N., Papadopoulos J., Bealer K. and Madden T.L. (2008). "BLAST+: architecture and applications." *BMC Bioinformatics*, 10:421.

Cock P.A., Antao T., Chang J.T., Bradman B.A., Cox C.J., Dalke A., Friedberg I., Hamelryck T., Kauff F., Wilczynski B. and de Hoon M.J.L. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25:1422-1423.

McKinney, W. (2010). Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56.