Software Documentation for ‘Melonomics’

Team Melon

Modupeh Betts

Andrew Knowles

Nadim Rahman

Madeleine Rhodes

Table of Contents

1 Introduction

1.1 Summary

This document is the guide for a prototype of an online bioinformatics web application (Melonomics) 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*'. This document contains information on what the program does, how it should be used and on its limitations and priorities for future development.

1.2 Intended audience for this document

This is a prototype web application and there are some limitations and technical issues in its current state. Therefore, as well as serving as a detailed overview of what the web-app can currently do, this guide is also aimed at developers who might want to help improve the functionality of the software and help in its eventual online deployment. The skills needed to understand the front and back-end of this software include: a good understanding of Python, Python-Flask, HTML, CSS, JavaScript, BLAST (and how to run it via Biopython), R and Plotly for R. Scientific understanding of the type of data that the program is optimised to analyse is also necessary. The program is designed to take in multiple *de novo* assembled RNA-sequences and perform differential gene expression analyses- and so knowledge of transcriptomics and RNA-seq technology is desirable.

1.3 System requirements

1.3.1 Current prototype state

The software is optimised to run with:

- ≥Python 2.7
- BLAST+ 2.6.0
- R 3.3.2 - Sincere Pumpkin Patch
- Linux/MacOS/Windows
- Any Web Browser

1.3.2 Finished state

The software would be optimised with:

- Linux/MacOS/Windows
- Any Web browser

This serves as the requirements for users (assuming the server is active).

1.4 Other information

University: Queen Mary University of London
Department: School of Biological and Chemical Sciences

Masters programme: *MSc Bioinformatics*

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

Project Supervisor: Professor Conrad Bessant --- c.bessant@qmul.ac.uk

Team members: Modupeh Betts (bt16015@qmul.ac.uk), Andrew Knowles (bt16612@qmul.ac.uk), Nadim Rahman (bt16661@se16.qmul.ac.uk), Madeleine Rhodes (bt16029@qmul.ac.uk).

1.5 Software specification

This prototype is designed to match the goals of a detailed specification which we were tasked with addressing. An extract of the most pertinent aspects of the 'official' specification can be found in the appendices (appendices, Figure 1).

The software is designed to analyse and interpret data from transcriptomics experiments. It is optimised for the analysis of results from *de-novo* assembled RNA-seq data. The basis of the statistical analysis is focussed around the associated FPKM values for each assembled transcript, a value which indicates the sequencing coverage for that transcript, and identifying the differentially expressed and top genes from the data. The software outputs publication standard interactive plots that can be downloaded.

The three primary aims for the project listed in the specification were for the software to be able 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."

1.6 Summary of Melonomics

The prototype software that we have developed allows a user to upload and analyse fasta files containing sequences from *de novo* assembled RNA-seq runs. These fasta files have to be from three different sample types- a control and two other experimental states but can be uploaded in any order preferred by the user. These

files are BLASTed against a locally hosted reference genome and then the top gene match for each transcript is returned.

The data from the BLAST is then passed to analysis which takes place in R. This is based around the analysis of the differential gene expression across the three sets of experimental data. The output from this is then passed back to the user in the form of interactive graphs that help with the interpretation of the data.

1.7 Scientific background to the software

The cost of sequencing has declined by several orders of magnitude since the turn of the millennium (National Human Genome Research Institute, 2016). Consequently, there has been a rapid growth in the 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- most notably with the rise of the 'omics' technologies (Kildegaard *et al*., 2013). The increasing use of these technologies have led to an exponential rise in the number of high-throughput sequencing experiments being conducted and with the sizes of typical data-sets being generated by labs growing concomitantly (National Human Genome Research Institute, 2016).

These developments represent a paradigm shift for the biological sciences; increasingly, modern biology can be thought of as a form of data science. 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.

1.8 Where Melonomics fits in

One of the 'omics' technologies which has benefited greatly from these technological leaps is transcriptomics, where the RNA transcripts present in a cell/tissue are analysed. Typically, transcriptomes from different developmental stages or physiological conditions are compared (Wang, Gerstein & Snyder, 2009). Transcriptomics has particularly benefited from the development of RNA-Seq, a method which provides an accurate measurement of levels of transcripts and their isoforms (Wang, Gerstein & Snyder, 2009). Melonomics is intended to be a transcriptomics tool that can help in the initial exploration and analysis of transcriptome data-sets.

The type of data that Melonomics is designed to analyse is *de novo* assembled RNA-seq transcripts across three different experimental states. Theoretically, data from experiments involving any species can be used with this prototype. Melonomics is optimised for differential gene expression analysis. Using RNA-seq technology, gene expression can be captured at particular time-points and in certain biological

contexts- through this the differential expression of genes can be examined across two or more biological conditions (Dembélé & Kastner, 2015). Studying differentially expressed genes can reveal crucial genomic 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 (Taube *et al.*, 2015; Tang *et al*., 2016).

As stated in the software specification, the user should be able to upload FASTA files with *de novo* assembled RNA-seq transcripts with the associated 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' (Malone & Oliver, 2011). The differential gene analysis is based around these FPKM values as they provide an accurate indication as to the level of expression for the sequences.

1.9 Design Philosophy

From the inception of the project, we approached the development of Melonomics with a simple design philosophy- from a minimalist perspective. We wanted our code to be as concise as possible and to read elegantly, with the functions and commands behind the key components of the software being clearly defined and easy to understand for future developers. The software was always intended to be handed in as a work-in-progress and so this meant we were careful for the comments in the code to be comprehensive and completely unambiguous. With regards to the website itself, we decided that we wanted to give it a slick and professional look; something that would be easy to navigate but engaging for the user.

2 User Guide

2.1 Basic User Requirements

The software is accessed via the URL that is generated by Flask. Users must have a web-browser which can interact with the framework. However a server 'administrator' will have to run the application for users to access it. The administrator's requirements have been discussed in Mechanics of Software (section 3). The software currently has to be run through *final.py*, generating the locally hosted server. The user will also need suitable input fasta files for the BLAST and analysis. Fasta files have been discussed below (section 2.3).

2.2 General App Navigation

To use the software, the user should select "Analyse Transcripts" on the home page. This will redirect to the file upload pages. Firstly the user is prompted to upload one or more control sample files (Group 1). Once complete, a button beneath the upload section redirects the user to a separate page to upload sample files from the first experimental state (e.g. Group 2). This is repeated for a second experimental state (Group 3), however the button redirects the user while the BLAST is carried out.

Once the BLAST is completed the R analysis is automatically undertaken. Once this is finished, the user is presented with a web page informing them that their results are ready. This page also contains options to open PDF or HTML version of specific R analyses in a separate tab. For example "Hierarchical Cluster Analysis" (HCA) will redirect to a PDF showing the HCA plot, while "Variance PCA" will open a HTML showing an interactive combined cumulative and explained variance plot.

The home page has links to other pages: "The Software", "About Us" and "Contact". Currently the first two selections redirect users to relevant pages. *The software* broadly describes what Melonomics does. *About Us* provides information about the authors of the software and this project. The contact us page is yet to be linked with the official team email, and currently just linked to the email address of one of the authors.

2.3 Required File Types and Formats

The software is optimised to analyse fasta formatted files (example: *filename.fasta*) of *de novo* assembled RNA-sequences with their FPKM values derived from an experimental design. They are focussed on differential gene expression across three different experimental states. Therefore users need to upload at least three files containing RNA-sequences and from three different experimental conditions. Although users are limited to uploading just three sample types, these files can be

named flexibly. However the files need to be named so that they reflect their experimental state and they must end with the '.fasta' suffix. For example a suitable set of three files would be 'control.fasta', '8hours.fasta' and '24hours.fasta'. Importantly, the user can upload multiple files for each sample type- the software is able to process replicates.

Figure 1 is an example of the required format that each file should consist of; 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 input files.

```
>asmbl_104;2.95
GCTGGGGCCTTCATCTTCCCGAGGGCGGTGAGTGCCCTG
CTGGCAGGGCCAGC
>asmbl_105;9.89
AGGGTCTCAACCATTAATCTCCAATCTTGATATCTATCCCC
ATCCCTTTCTCCCTACACCCGGGGCGGCATCTCAGGCAG
CAATTTGACTCCTAGCA
>asmbl_106;2.95
CGAGGGCGGTGAGTGCCCTGCTGGCAGGGCCAGCTCTC
CAATCTTGATATCTATCCCCATCCCTTTCTCCCTACACCCG
GGGCG
```

Figure 1. An example of the required format of the sequences in each input fasta file; each sequence is associated with an identifier and FPKM value.

2.4 Parameters of the BLAST Search

The standard BLAST of the software queries sequences against a locally hosted nucleotide database 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.

BLAST is carried out against the 'nt' NCBI database. This is an extensive database which consists of nucleotide sequences from the DNA Data Bank of Japan (DDBJ), European Molecular Biology Laboratory (EMBL) and GenBank of the NCBI (Tao *et al*, 2011). The nucleotide sequences stored are from a wide variety of organisms, including *Pteropus alecto*.

Keeping only the closest BLAST-match for each input sequence enables for 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 in accordance with

the input filenames. All other BLAST parameters follow the default setting, found at BLAST® Command Line Applications User Manual (2008).

2.5 Output of BLAST

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

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

The most important columns are the query_ID and subject_ID. The query_ID refers to the description line of a given sequence from the fasta file. The top hit gene ID is known as the subject_ID (BLAST, 2008), however this will be referred to as gene ID.

The number of queries is dependent on the number of sequences within each fasta file. As stated in section 2.4, the top match and the returned parameters. This output is not very informative for analysis and is not displayed to the user, 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 and is further sorted depending on the sample name or type. This results in a matrix of genes, FPKM values and sample types. The matrix is generated through python Pandas and consequently transferred and saved as a CSV file. This is fed into R for analysis, as an appropriate input file to produce plots. Python Pandas was utilised due to the ease of generating matrices with specific values at certain cells of the matrix.

2.6 Downstream Analysis

A matrix generated from BLAST output is read in to R (version 3.3.2). The R code generates analysis, described below. The data was split into classes, required for the differential analysis of genes. Further information of the analysis carried out and the packages used can be found in section 3.8.

2.6.1 Principal Component Analysis

A PCA is a crucial statistical approach which is used in exploratory data analysis of differential gene expression studies. Of course, what a researcher is most interested here is which genes are expressed differently across changing biological conditions. A PCA is used to reduce the dimensionality of a data set, reducing the number of

variables into principal components. This can help get rid of redundant information across data sets being compared across different experimental states. In gene expression analysis this can be particularly useful as it can be that the majority of the gene expression levels are shared across different experimental states. In conclusion: PCAs can help identify the level of similarity between different samples and help in the identification of the genes with significantly different expression levels.

2.6.2 Hierarchical Cluster Analysis (HCA)

A major focus of the statistical analysis is based around HCA. In hierarchical clustering, clustering algorithms are used to compress large-scale genome-wide expression data; the genes in a data set are be subdivided into a smaller number of categories which then can be analysed themselves (D'haeseleer, 2005). The transcriptome data sets that Melonomics is designed to be able to handle might contain sequences matching to thousands, perhaps tens of thousands of genes; therefore the application of hierarchical clustering is an important part of the analysis. The results of a HCA can be visualised in various ways but in this instance are viewed as dendrogram.

2.6.3 Heatmaps with associated dendrograms

Heatmaps are widely used aspect of data exploration in differential gene expression analysis. In our analysis, the heatmaps produced are shown with dendrograms which are associated with results from the HCA. These help to highlight how the gene groups which have been clustered together from the different samples differ from each other in respect to the expression of the cluster groups. The heatmap outputs are colour-coded so that the user can see how the clusters compare in the expression level across the different samples.

Although the heatmaps in the output are always shown with dendrograms, the dendrograms themselves can be viewed separately to visualise how the clustering changes across the experimental states.

2.6.4 Top Genes Table

Another part of the output of Melonomics is a data-table which lists the top genes, by p.value, in terms of which is most differentially expressed. In other words; these are the genes that when comparing the control to the other two experimental states, are most differentially expressed across these different groups. It provides a quick and useful overview as to which genes are driving the differences in the expression between the different groups.

2.6.5 Volcano Plots

A volcano plot graphically represents the fold change in expression for individual sequences on the x-axis, while the statistical significance is plotted on the y-axis. With this statistical approach the level of dysregulation can be easily visualised and the strength of this dysregulation is clearly indicated by how high the data-point is on the y-axis. This analysis is very important is in a transcriptome analysis there is a huge amount of data and thus variation- volcano plots help cut through the statistical noise that can cloud a transcriptome analysis.

3 Mechanics of the Software

3.1 Software Architecture

The software consists of defined frontend and backend roles, shown in fig. 2. The user interacts with a frontend interface generated using flask and associated css and templates. The general folder structure of the application is shown in fig 4. If the app is downloaded, file paths may need to be changed in all main scripts of the app folder. The web page consists of a section to “Analyse Transcripts” which enables the user to upload multiple fasta files. These fasta files are stored in a data folder generated by the flask framework. The folder itself is organised into control and disease states 1 and 2.

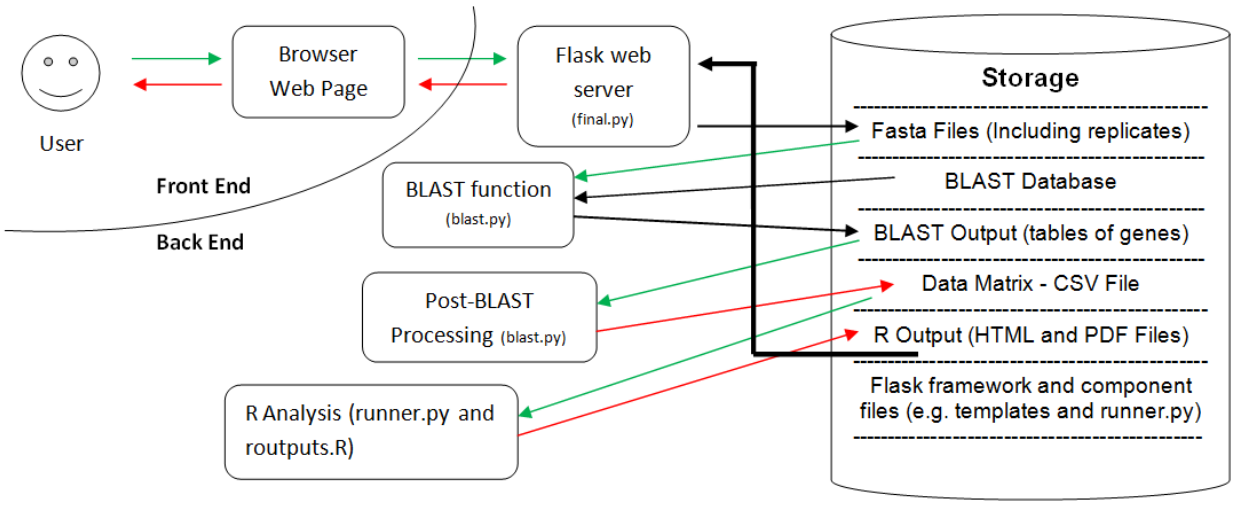

Figure 2. Software architecture for Melonomics. This rough diagram presents the flow of data and the different components required. The main input and output files of the software have been represented by green and red arrows respectively. The black arrows represent general data flow.

The BLAST function requires fasta formatted files as input, along with a reference database which is locally hosted and used to make comparisons against (fig.3) Following this stage, gene IDs of correlated sequences are obtained from the saved BLAST output and fed into a post-BLAST processing stage. This generates a matrix of gene IDs with their matching sequences’ FPKM value, which are further mapped to the type or name of sample. The matrix is stored within the data folder as a CSV file, ready for analysis.

The R analysis is carried out with a python script optimised for R commands. For non-interactive plots, this generates a PDF file which is saved in a data folder.

Alternatively interactive plots are saved in HTML format. These PDF and HTML files are returned to the user through the interface. With PDF files, the user is able to download the file, thereby storing their results.

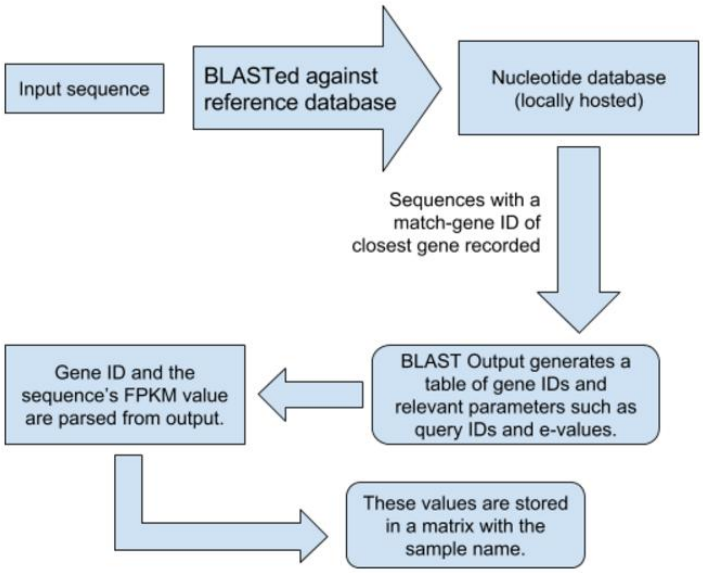

Figure 3. Diagram indicating the workflow for the BLAST component of the software, which in itself was split into various tasks shown.

```
app
  L static
  |      L [Any files within the static folder]
  |
  L templates
  |         L [HTML Templates]
  |
  L data
  |    L [User-uploaded fasta files]
  |    L [Output from BLAST and R (when the app is run)]
  |
  L BLAST.py
  |
  L final.py
  |
  L routputs.R
  |
  L runner.py
```

Figure 4. General distribution of files within the application. This can be found on GitHub, within the README file outside of the main app folder.

3.2 The Flask Framework

3.2.1 Overview

Flask is a Python web framework; with Flask it is possible to build a micro-framework web-applications. Our web-application has been built around Flask. As well as displaying the HTML pages of the web-pages that the user interacts with, it functions as the interface between the front-end and back-end of the software. Being a Python-based micro-framework, with Flask it is simple to integrate Pythonic modules and packages with it and this is what makes Flask such a powerful tool for the creation of a bioinformatics web application.

Flask is designed for the production of a website or web-application based around python. The different URLs of a Flask application are linked to their own 'route', which are each linked to a specific python function. These routes are the interface between the web application, the data folders and the different components of the analysis.

3.2.2 Running Flask

Flask can be installed via http://flask.pocoo.org/docs/0.12/installation/, where comprehensive documentation can be found.

There are multiple ways to run our flask-app. For example an Integrated Development Environment (IDE) such as PyCharm can be used to open the application. The app folder (and subfolders) simply needs to be run as a 'project' with the necessary packages installed- and then the website can be run. Using an IDE for future development of Melonomics is recommended for developers. This is largely due to the ease of mapping different files within various directories and the ease of running the server to visualise the templates. Additionally, the debugging mode works well with IDEs such as Pycharm.

An alternative to running the app via an IDE is to invoke the file with all app routes (*final.py*) on command line (e.g. *python final.py*). A URL is provided, which should be pasted into a web browser to present the main web page. The app routes enable for different functions to be carried out and rendering of various templates presenting a web page. The functions are imported into *final.py* to run with their specific URL extension, presenting with a unique HTML template. For example for the BLAST function, BLAST.py (the file containing all commands to carry out a BLAST search) is imported into final.py to run with the URL route extension '/Blast'. The ease of importing files with their own functions was a major advantage of flask. In other cases, these URL route extensions correspond to a HTML template. Multiple

templates with CSS and JavaScript styles were used to produce neat and elegant web pages.

3.3 Required Installations/Packages for Flask

3.3.1 OS

To run any flask scripts, ensure that the os package is installed (Python, 2017). This was required to carry out OS tasks, e.g. changing and setting working directories. Its use was crucial due to the file structure system of flask framework.

3.3.2 Redirect

The redirect package is very self explanatory. It was used to redirect users to an alternative web page. This was coupled with the url_for package; specifically url_for(template_name) was passed as an argument. This was used to provide flow to the software and logical links between different components. Its use was especially useful due to the generally modular property of Melonomics.

3.3.3 Render_template

HTML is generated in the form of templates which are displayed on screen as web pages. HTML escaping must be done separately to maintain application security. With flask, Jinja2 template engines are configured, however requires the package render_template. Once a specific template has been rendered, flask automatically searches for the template within the templates folder (Flask, 2017). Despite repetitive HTML having to be generated, the usage of this module allows for web pages to be generated in desired styles. Furthermore templates can be called on at corresponding locations within the software, generating a structured website or application. A normal argument passed through the render_template variable would be a HTML template which has been generated and saved within the templates folder.

3.3.4 Request

This package was required to upload fasta files to the data folder of the server. However to provide a greater level of structure to the software, the data folder was split depending on the types of samples. The package enabled for any number of files uploaded on a specific web template to be saved in a certain directory. In addition to the added structure, this enabled for a more efficient BLAST script and function, further downstream.

3.3.5 Send_file

To return R analysis results to the user, send_file was utilised. In particular, this was required for the PDF files which contain specific R plots (such as hierarchical cluster analysis) which can be viewed using send_file. The name and path of the file was specified within the send_file argument. The file is presented in an additional tab on the browser.

3.3.6 Send_from_directory

This package was used to direct to a specific directory, without fully specifying paths. This was of particular importance when incorporating JavaScript from the HTML widget plots produced by R analysis. The directories were specified to organise the structure and allow compatibility of HTML interactive graphs to flask framework.

3.3.7 Url_for

A key element of web pages includes the ease of navigation and linkage between one web page to another. In some cases, the linkage can be automatic, flowing from one stage to another. The url_for package (coupled with redirect) allows developers to code for a redirection of users from one web page to another. For example, the user uploads their last set of fasta files, and once submitted, the BLAST is carried out. The package enabled for flow between the two tasks by specifying a template to load.

3.4 HTML Templates for Flask

HTML templates are crucial to the aesthetics of specific web pages (table. 1). For example, R_Downloads.html consists of various buttons which redirect users to relevant R analysis. This enables Melonomics to look more professional and with CSS and Javascript, specific themes, fonts, etc. can be incorporated. Furthermore HTML templates provide a level of structure and flow to the software, e.g. linking web pages to inform users what the software is doing. The following table provides a description of what each template is required for.

Table 1. Presents the HTML templates present within the templates folder. A description provides information about the templates function.

HTML Template	Description
ABOUTUS.html	The page which contains information about the authors of the software.
INDEX.html	This is the home page, the first page a user is greeted with. Also, contains button to start data analysis.
error.html	A 404 erro page.
layout.html	Contains the CSS and Javascript maintained for other pages.
Rdownloads.html	Informs the user that R analysis is complete and presents links which enable users to view specific analysis.
software.html	A web page describing Melonomics.
upload.html	Users upload their control sample fasta files.
upload2.html	Users upload their second sample fasta files.
upload3.html	Users upload their third sample fasta files.

3.5 CSS and JavaScript

Javascript was used in the project to incorporate specific styles for the HTML templates which generated the web pages. This brought about a more professional-looking software and website, which in the future can be incorporated for a domain. Additionally, CSS (a markup language) was crucial for the styles utilised for the software presentation (Keller and Nussbaumer, 2010). HTML on its own, describes the content of the web pages. With CSS, specific styles can be implemented.

3.6 BLAST

3.6.1 Overview

All BLAST commands are stored within the *BLAST.py* file. This is imported into *final.py* to run when a user has uploaded all fasta files. Alternatively the *BLAST.py*

file can be invoked on command line and generates the relevant output files. These include the BLAST output for each fasta file and also a matrix of the gene IDs, FPKM values and sample names, stored in a CSV file. Therefore BLAST.py also handles post-BLAST processing to ensure desired output for analysis in R.

3.6.2 Local vs Online

The alignment was carried out locally to reduce the time taken to receive results. This benefits the user as in theory much larger fasta files would be input. The main parameters of a local BLAST follow that of its online compatriot. However, local enables for further enhancements, such as specifying the number of CPUs utilised. This can be increased to reduce the time taken. Alternatively, sending large queries to the NCBI website for online BLAST would not be feasible and may require additional permissions from NCBI. The local BLAST does have major drawbacks, including downloading and storing large BLAST databases to query against. Additionally, these databases must be kept up to date on a regular basis. However these disadvantages were outweighed by the advantages of local BLAST against online.

3.6.3 Required Installations and Packages

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 (Python, 2007) and appending the biopython packages to the path. The NcbiblastnCommandline (Altschul *et al*, 1990) 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 (Camacho *et al*, 2008) 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.

The post-BLAST stages require python pandas to generate matrices. This can be installed with SciPy (https://www.scipy.org/install.html), thereby including other packages such as NumPy. Pandas was utilised as it provided an efficient method to generate dataframes or matrices which were required for R analysis. The matrices generated were compatible with the dictionaries which were generated from the gene IDs, FPKM values and sample names.

Packages for post BLAST processing include re (Python, 2017) (regular expression) 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 (Python, 2017), 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.

3.6.4 Parameters

Biopython's NcbiblastnCommandline consists of standard parameters utilised within the current code. The constructed command is printed for each sample passed through. Query refers to the input fasta file and db is used for the type of database, which has been set to 'nt' for nucleotides. The output file (out) specifies the name of the BLAST output file and outfmt 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.

3.7 R Overview

The statistical analysis performed by Melonomics is undertaken by the programming language of R (R Core Team, 2016). R is a hugely flexible program and is benefitted from the vast array of packages available which can augment your analysis.

The R code is run through the python script *runner.py* which imports a package called subprocess (Python, 2017). The use of subprocess was crucial as it simplified the process of importing the R script (*routputs.R*) with the necessary libraries. Utilising subprocess enabled for incorporation of the unaltered R script into flask with

minimal issues. An alternative approach to the integration of the R script would have been with the use of Rpy2 (Rpy2, 2017). However, we found that Rpy2 in this context would have been over-complicated; requiring numerous changes to the python and R script for full integration.

The R analysis is configured to analyse differential gene expression of *de novo* assembled RNA-seq sequences extracted from three different experimental conditions - one a control and two different experimental states. After the BLAST search is undertaken for each of the input files against the database of nucleotides, the closest gene-match is assigned to each input sequence- with the FPKM value being kept as assigned to the entry sequence. The R analysis is based around analysis of this output.

3.8 R packages

3.8.1 Plotly

Plotly is an online data analytics and visualisation tool which is well suited for integration with flask (Sievert *et al*., 2016). Plotly was an obvious choice for us as one of the most important tasks was the visualisation of the data-sets that we will be analysing with R. One of the advantages of plotly is that the graphs it produces are interactive; for example a user can hover their cursor over a data point and detailed information about that point will be displayed. For an analysis such as ours, where one graph might contain a huge number of individual data points, this interactivity is extremely useful. Plotly graphs were used to generate HTML plots which were saved and then incorporated into the flask framework.

3.8.2 Gplots

Gplots is a comprehensive package which extends the functionality of the data plotting of various analysis approaches in R (Warner *et al.*, 2016). The only use of this package here was to create the heatmap using the function *heatmap.2*, this function provides a number of extensions to the *heatmap* function that is offered by R, such as increased design which made it favourable for this software. Should added functionality such as distance and clustering methods be included as an option (section 4.9), this package would have been the ideal choice.

3.8.3 Dendextend

Dendextend is a package which improves and adds to the functionality of Dendrogram objects in R (Galili, 2015). It allows for the adjustment of the graphical parameters of dendrogram objects in addition to allowing for simple comparison of dendrograms. One of the main parts of the analysis of Melonomics is the hierarchical cluster analysis and so this package is well-suited to the software.

3.8.4 DT (DataTable)

This package allows for R data objects to be displayed as tables as part of html pages, it does this by providing an interface to the 'DataTables' JavaScript library (Xie, 2016). Our flask application displays web-pages through the rendering of HTML templates and so this package is useful for integrating the output of R into the web-pages that the user interacts with. A particularly useful feature. This package allows the user to sort each column of the table from highest to lowest value or vice versa and allows particular genes to be searched for in a search bar.

3.8.5 RColorBrewer

This is a simple package which is designed to help provide or improve the colour schemes for graphics generated through R (Neuwirth, 2014). It is important that applications such as Melonomics provide engaging graphics to help users understand their results and so packages such as these are important to help make the application look more polished and professional.

3.8.6 Limma

The limma package was imported into from the Bioconductor source (http://bioconductor.org/biocLite.R) . The package is used for the analysis of gene transcript expression data that arises from RNA-seq technologies (Ritchie, 2015). The main reason for using this package is that it has the capability to handle large volumes of data from multifactor designed experiments through the use of linear models that can accurately assess differential gene expression.

4 Limitations and Future Development

4.1 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.

4.2 Multiple Input Files

Currently, the software is configured only to analyse data from three different experimental states; a control and then two different other states. This is because our prototype was built around an example data-set which had control sequences, then data from '+*8 hours of infection*' and '+*24 hours of infection*'- three different states. In its future development a critical improvement would be to change the configuration so that a user can upload a control data set and data from any number of different experimental states.

These changes should be relatively simple to implement as the underlying mechanics of the application would function in the same way; the files would be passed between the different components of the software in the same manner- but it would just have to be more flexible.

In regards to Flask, currently the uploaded files are stored in three separate data folders depending on the sample type. The user has to upload each of three types of sample as prompted by the websites different upload pages. This constitutes a rigid system, which could be improved with changes to the file upload templates. Ideally the user should be able to tell the application how many different sample types to 'expect' before uploading. A more dynamic folder system would have to accompany this, generating folders as the user uploads specific types of files.

The software currently allows many types of files to be uploaded to the website. If a .fasta file is not uploaded, an error in the BLAST will occur. To filter the file type to accept solely fasta files, the package - werkzeug can be used to specify suitable extensions. Furthermore files must be uploaded or else an error page appears, this can be easily fixed by having a logical (if statement) that keeps the user on the same page or redirects to the home page when the submit button is hit without uploading a file.

4.3 Multiple Users

Currently only one user can use the software at a time. One priority in the scaling up of the software is that it could 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 (SQLAlchemy, 2017) 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.

4.4 Flask Framework

Despite the ease of using python flask, one major disadvantage lies with its incompatibility with R's HTML plots. These plots include their own javascript which is separate from the javascript used in the flask framework. Therefore compatibility with flask and issues with file paths currently exist. Although, this issue was overcome by manually indicating the HTML plot Javascript file paths, within the *final.py* script. An alternative (and more efficient) solution may have been to use an alternative type of web framework, such as Shiny- however this would represent a major change to the underlying structure of the application.

4.5 BLAST

Currently it takes a large amount of time for even a small number of Fasta files to be blasted against the reference database; in one trial it took 2 hours to blast -and

retrieve the results for- 9 short sequences. Since transcriptome data files can include hundreds of thousands of sequences a priority would be to improve the speed of the blast search. It is likely that these slow speeds are due to the trails being run on relatively slow consumer laptops. To reduce the time taken, the BLAST searches could be carried out on a High Performance Cluster (HPC). Through the use of a HPC, the memory and storage requirements could easily be met and the massive computing power should drastically reduce the time of the BLAST search.

4.6 Advanced BLAST settings

Comprehensive 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 the e-value cut-off for sequence matches is already set. Several options should be added to the BLAST functionality:

- The option to choose the type of database to BLAST against (e.g. specifying a reference genome of a particular organism)
- The ability to change the threshold of the match scores for the sequences.
- Altering the number of gene hits returned for each sequence.

The ability for the user to pick which database to BLAST against is a challenge. One approach would be to separate functions dependant on the database type selected. However a more robust and tidy method would be to pass an additional variable (e.g. dbName) through the BLAST function, which differs depending on the the user's requirement. A disadvantage to this additional functionality includes the vast storage requirements. This makes saving the databases to a server infeasible.

Furthermore, it would be useful for the blast output to include the e-value and number of gene IDs returned for each query sequence. This would provide additional useful information for the user. 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.

4.7 BLAST Output

The BLAST output returns all matches found between the query sequence and the database. A useful improvement for this output would be for sequences which don't match with anything to be returned in a separate file, with their associated sequence

IDs, as these could potentially be novel genes or undiscovered isoforms. This could be a valuable tool for researchers. Furthermore, it would provide an explanation as to why the number of sequences queried may not match to the number of gene IDs.

An additional improvement would alter the BLAST output to have a cumulative FPKM score for two or more sequences which map to the same gene ID. This would be incorporated through list comprehensions to check whether a given gene ID is present within the gene IDs list. Furthermore, the relative FPKM values must be added and kept within the FPKM value list.

4.8 R script

The R script is currently configured only to analyse three types of samples; a control and samples from two other experiment states. Therefore, all the statistical analysis are currently based around three classes or groups which are present on the plots. However, the R script needs to be flexible enough to deal with any number of sample types- each with an undefined number of replicates.This could be altered by including clauses which consider the types of samples present within the matrix generated by BLAST. Therefore the number of classes or groups defined would be dynamic and dependent on the user. Alternatively the code could be placed into functions which are called on depending on the types of samples present.

4.9 Dendrogram Distance and Clustering Methods

The heatmaps with associated dendrograms have a default clustering method of complete linkage. With some samples, this may not offer optimum grouping, resulting in unreliable grouping of samples. Ideally, these settings would be chosen by the user, therefore including a dropdown list in the flask framework prior to R analysis. Additionally, defined functions (with variables passed through) for the dendrograms can be used to incorporate a chosen type of distance or cluster method.

An alternative method would be for R analysis to try all distance and clustering methods and return the heatmaps to the web page. The user would then decide which combination is best for their data. Therefore analysis would become increasingly tailored to the user and analysis would be more reliable. The types of distance methods include Manhatten, Maximum, Canberra, Binary and Minkowski. Clustering methods include Centroid, Median, Mcquitty, Average, Single, Ward D and Ward D2.

4.10 Principal Component Analysis (PCA) Functionality

A major issue with the PCA in this software is the graphical representation of the principal component scores. Currently, the software only shows a comparison of the

first five principal components. With the huge datasets that this software is designed to analyse, only displaying the first 5 principal components will limit further analysis. Resolving this problem would involve adapting the R script so that it can handle an undefined number of principal components.

One idea for further functionality is to integrate a plotly graph which has two drop-down menus containing all the principal components. This would allow users to select which two components they wish to compare which would then be displayed on a graph.

4.11 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. For example, importing the package bcrypt enables for password hashing and verification (Pypi, 2017).

6 References

Altschul, S.F., Gish, W., Miller, W., Myers, E.W. & Lipman, D.J. (1990) "Basic local alignment search tool." *J. Mol. Biol.* 215:403-410.

BLAST® Command Line Applications User Manual [Internet] (2008). Bethesda (MD): National Center for Biotechnology Information (US); Options for the command-line applications. Available from: https://www.ncbi.nlm.nih.gov/books/NBK279675/.

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.

Dembélé, D. and Kastner, P. (2014) Fold change rank ordering statistics: a new method more detecting differentially expressed genes. *BMC Bioinformatics*, 15:14. DOI:10.1186/1471-2105-15-14.

D'haeseleer, P. (2005). How does gene expression clustering work? *Nature Biotechnology*, 23: 1499-1501.

Flask (2017). *Quickstart — Flask Documentation (0.12)*. [online] Available at: http://flask.pocoo.org/docs/0.12/quickstart/ [Accessed 15 Feb. 2017].

Galili, T. (2015). dendextend: an R package for visualizing, adjusting, and comparing trees of hierarchical clustering. *Bioinformatics*, 31(22): 3718-3720. DOI: 10.1093/bioinformatics/btv428

Keller, M. and Nussbaumer, M. (2010). CSS Code Quality: A Metric for Abstractness Or Why Humans Beat Machines in CSS Coding. *Seventh International Conference on the Quality of Information and Communications Technology*. IEEE, 116-121.

Kildegaard, H.F., Baycin-Hizal, D., Lewis, N.E. and Betenbaugh, M. J. (2013) The emerging CHO systems biology era: harnessing the 'omics revolution for biotechnology. *Current Opinion in Biotechnology,* 24(6):1102-1107. DOI:10.1016/j.copbio.2013.02.007.

Malone, J.H. and Oliver, B. (2011) Microarrays, deep sequencing and the true measure of the transcriptome. *BMC Biology,* 9:35. DOI: 10.1186/1741-7007-9-34.

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

National Human Genome Research Institute. (2016) The cost of sequencing a human genome. [online] Available at: https://www.genome.gov/sequencingcosts/ [Accessed 16 Feb. 2017].

Neuwirth, E. (2014). RColorBrewer: ColorBrewer Palettes. R package version 1.1-2. https://CRAN.R-project.org/package=RColorBrewer

Pypi (2017). *bcrypt 2.0.0 : Python Package Index*. [online] Available at: https://pypi.python.org/pypi/bcrypt/2.0.0 [Accessed 15 Feb. 2017]

Python (2017). *13.1. csv — CSV File Reading and Writing — Python 2.7.13 documentation*. [online] Available at: https://docs.python.org/2/library/csv.html [Accessed 14 Feb. 2017].

Python (2017). *15.1. os — Miscellaneous operating system interfaces — Python 2.7.13 documentation*. [online] Available at: https://docs.python.org/2/library/os.html [Accessed 14 Feb. 2017].

Python (2017). *17.1. subprocess — Subprocess management — Python 2.7.13 documentation*. [online] Available at: https://docs.python.org/2/library/subprocess.html [Accessed 15 Feb. 2017].

Python (2017). 28.1. *sys — System-specific parameters and functions — Python 2.7.13 documentation*. [online] Available at: https://docs.python.org/2/library/sys.html [Accessed 14 Feb. 2017].

Python (2017). *7.2. re — Regular expression operations — Python 2.7.13 documentation*. [online] Available at: https://docs.python.org/2/library/re.html [Accessed 14 Feb. 2017].

R Core Team (2016). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL:https://www.R-project.org/.

Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., and Smyth, G.K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research,* 43(7): e47.

Rpy2 (2017). *Documentation for rpy2 — rpy2 2.8.4 documentation*. [online] Available at: https://rpy2.readthedocs.io/en/version_2.8.x/ [Accessed 16 Feb. 2017].

Sievert, C., Parmer, C., Hocking, T., Chamberlain, S., Ram, K., Corvellec, M. and Despouy, P. (2016). plotly: Create Interactive Web Graphics via 'plotly.js'. R package version 4.5.6. https://CRAN.R-project.org/package=plotly

SQLAlchemy (2017). *SQLAlchemy - The Database Toolkit for Python*. [online] Available at: http://www.sqlalchemy.org/ [Accessed 15 Feb. 2017].

Tao, T., Madden, T., Christiam C., *et al*. *BLAST FTP Site*. 2011 May 29 [Updated 2016 May 10]. In: BLAST® Help [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2008-. Available from: https://www.ncbi.nlm.nih.gov/books/NBK62345/

Tang, H., Hammack, C., Ogden, S.C., Wen Z., Qian, X., Li, Y., Yao, B., Shin, J., Zhang, F., Lee, E.M., Christian, K.M., Didier, R.A., Jin, P., Song, H. and Ming, G. (2016) Zika Virus Infects Human Cortical Neural Progenitors and Attenuates Their Growth. *Cell Stem Cell*, 18(5): 587-590. DOI: 10.1016/j.stem/2016.02.016.

Taube, J.M., Young, G.D, McMiller, T.L., Chen, S., Salas, J.T., Pritchard, T.S., Xu, H., Meeker, A.K., Fan, J., Cheadle, C., Berger, A.E., Pardoll, D.M. and Topalian, S.L. (2015) Differential Expression of Immune-Regulatory Genes Associated with PD-L1 Display in Melanoma: Implications for PD-1 Pathway Blockade. *Biology of Human Tumours,* 21(17): 3969-3976. DOI: 10.1158/1078-0432.CCR-15-0244.

Wang, Z., Gerstein, M. and Snyder, M. (2009) RNA-Seq: a revolutionary tool for transcriptomics. *Nat Rev Genet*., 10(1): 57-63. DOI: 10.1038/nrg2484.

Warnes, G.R., Bolker, B., Bonebakker, L., Gentleman, R., Liaw, W.H.A, Lumley, T., Maechler, M., Magnusson, A., Moeller, S., Schwartz, M. and Bill Venables (2016). gplots: Various R Programming Tools for Plotting Data. R package version 3.0.1. https://CRAN.R-project.org/package=gplots

Xie, Y. (2016). DT: A Wrapper of the JavaScript Library 'DataTables'. R package version 0.2. https://CRAN.R-project.org/package=DT

7 Appendices

Figure 1. The official specification (edited for clarity) from the project briefing, which Melonomics is based on

---

"The overall aim of this module is to give you the experience of working together within a team to produce a working prototype of a web-based software tool for interpreting the results of transcriptomics experiments. Specifically, results obtained from RNA-seq followed by de novo assembly…

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."

Figure 2. Software FAQs

These FAQs re-iterate much of the content already covered in the documentation (lots of it is copy & paste). It's included in this report as in a real-life scenario,

potential developers on GitHub, when looking at a documentation, may want easily accessible answers. A similar page to this would be integrated in the final version of the web-app.

Is the software suitable for your dataset?
This software is currently in its prototype stage. It is optimised to analyse multiple *de novo* assembled RNA-seq data-sets which have come from an experiment designed around the analysis of differential gene expression. For the functionality to be effective, each input sequence needs to have an associated FPKM value with it, which gives an indication of the coverage of the sequences. The statistical basis of this differential analysis is based around the differences in FPKM values across the different genes identified and so it is an essential component of the dataset that you upload. Currently, you need to have data from a control and two different experimental states for the software to be effective.

What format should my files be in?
The files need to be in a FASTA format. Each sequence should have a header marked by a > and followed by its identifier and FPKM value. An example of the desired format can be found below:

```
>asmbl_104;2.95
GCTGGGGCCTTCATCTTCCCGAGGGCGGTGAGTGCCCTG
CTGGCAGGGCCAGC
>asmbl_105;9.89
AGGGTCTCAACCATTAATCTCCAATCTTGATATCTATCCCC
ATCCCTTTCTCCCTACACCCGGGGCGGCATCTCAGGCAG
CAATTTGACTCCTAGCA
>asmbl_106;2.95
CGAGGGCGGTGAGTGCCCTGCTGGCAGGGCCAGCTCTC
CAATCTTGATATCTATCCCCATCCCTTTCTCCCTACACCCG
GGGCG
```

Do my files have to be named in a specific way?
Not necessarily. However, your files should be named in a manner which ensures that the experimental state of each file is clearly defined (e.g. control1.fasta, etc).

What type of analysis will be performed?
The software is optimised to measure the differential expression of RNA from an organism which has had RNA-sequences extracted across multiple experimental conditions. Firstly, a BLAST search is undertaken for each of the input files. Each RNA-sequence from the files is BLASTed against a reference database of nucleotides and the closest gene-match is assigned to each inputted sequence- with the FPKM value being kept as assigned to the entry sequence.

After the BLAST search, which might take some time, the results of the statistical analysis should be made available. There will be a Principal Component Analysis (PCA), a hierarchal cluster analysis (HCA), heat-maps with their associated dendrograms and volcano plots. The section below will explain how each This statistical approach is useful in differential gene expression analysis HCA, Heat-maps and associated dendrograms, volcano plots as well.

What's a Principal Component Analysis (PCA)?

A PCA is a crucial statistical approach which is used in exploratory data analysis in differential gene expression studies. Of course, what a researcher is most interested here is which genes are expressed differently across changing biological conditions. A PCA is used to reduce the dimensions of a data-set, reducing the number of variables into principal components. This can help get rid of redundant information across data-sets being compared across different experimental states. In gene expression analysis this can be particularly useful as it can be that the majority of the gene expression levels are shared across different experimental states. In conclusion: PCAs can help identify the level of similarity between different samples and help in the identification of the genes with significantly different expression levels.

What's a Hierarchical Cluster Analysis (HCA)?

A major focus of the statistical analysis is based around HCA. In hierarchical clustering, clustering algorithms are used to compress large-scale genome-wide expression data; the genes in a data set are be subdivided into a smaller number of categories which then can be analysed themselves (D'haeseleer, 2005). The transcriptome data sets that Melonomics is designed to be able to handle might contain sequences matching to thousands, perhaps tens of thousands of genes; therefore the application of hierarchical clustering is an important part of the analysis. The results of a HCA can be visualised in various ways but in this instance are viewed as dendrogram.

What are the heatmaps with associated dendrograms?

Heatmaps are widely used aspect of data exploration in differential gene expression analysis. In our analysis, the heatmaps produced are shown with dendrograms which are associated with results from the HCA. These help to highlight how the gene groups which have been clustered together from the different samples differ from each other in respect to the expression of the cluster groups. The heatmap outputs are colour-coded so that the user can see how the clusters compare in the expression level across the different samples.

Although the heat maps in the output are always shown with dendrograms, the dendrograms themselves can be viewed separately to visualise how the clustering changes across the experimental states.

What is the 'Top Genes Table' showing?

Another part of the output of Melonomics is a data-table which lists the top genes, by p.value, in terms of which is most differentially expressed. In other words; these are the genes that when comparing the control to the other two experimental states, are most differentially expressed across these different groups. It provides a quick and useful overview as to which genes are driving the differences in the expression between the different groups.

What are Volcano Plots?

A volcano plot graphically represents the fold change in expression for individual sequences on the x-axis, while the statistical significance is plotted on the y-axis. With this statistical approach the level of dysregulation can be easily visualised and the strength of this dysregulation is clearly indicated by how high the data-point is on the y-axis. This analysis is very important is in a transcriptome analysis there is a huge amount of data and thus variation- volcano plots help cut through the statistical noise that can cloud a transcriptome analysis.