VISUALIZATION OF GENE EXPRESSION DATA FROM CHINESE HAMSTER OVARY (CHO) CELLS

by

Sansheng Chen

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the Master of Science in Bioinformatics and Computational Biology

Summer 2017

VISUALIZATION OF GENE EXPRESSION DATA

FROM CHINESE HAMSTER OVARY (CHO) CELLS

by

Sansheng Chen

Approved: _____

Kelvin H. Lee, Ph.D.
Professor in charge of thesis on behalf of the Advisory Committee

Approved: _____

Kathleen F. McCoy, Ph.D.
Chair of the Department of Computer and Information Sciences

Approved: _____

Babatunde A. Ogunnaike, Ph.D.
Dean of the College of Engineering

Approved: _____

Ann L. Ardis, Ph.D.
Senior Vice Provost for Graduate and Professional Education

ACKNOWLEDGMENTS

I would like to express my deep gratitude to Professor Kelvin H. Lee. He expertly guided me through this project and inspired my passion for doing this project. I am sincerely grateful for his guidance and mentorship, and it was my pleasure working with him.

I would like thanks to Professor Behnam Abasht, Professor Thomas Hanson, and Professor Carl Schmidt for spending their valuable time participating in the committee meetings and for giving me invaluable advice and friendly responses.

I am also indebted to my laboratory colleagues for their kind and endless help at all times, especially Madolyn Macdonald who gave me generous support. I appreciate their patience in helping me prepare presentations and thesis writing in this project.

Finally, I am grateful to my parents and closest friends. They gave me love and encouragement during the completion of this project.

Thank you!

Sansheng Chen

TABLE OF CONTENTS

Appendix

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Recombinant therapeutic proteins have become one of the preferred treatments for many human diseases. CHO cell lines are the most frequently used mammalian cell system for the production of therapeutic proteins. With the increasing number of studies in CHO cells, understanding the basic information of CHO cells is essential to make further improvements in therapeutic protein production. As the genome sequence provides a basis for genetic engineering and transcriptomic data profiles gene expression information, a platform that allows scientists to quickly go through CHO cell genome and transcriptome information will be extremely helpful. Reference genome sequences and transcriptome annotation information of CHO cells can be visualized by JBrowse on CHOgenome.org, a publicly available website. However, currently no tool allows users to easily visualize gene expression levels under certain conditions from previous experiments. To build a tool for gene expression data visualization, custom Python scripts were created to load transcriptomic data from previous studies and generate new files in a JBrowse acceptable format, which enables the visualization of mRNA expression data in JBrowse. The tool made up by these Python scripts is named 'ExpressGENiE', and can process data either from RNA-Seq or DNA microarray experiments by choosing certain functions. Through the use of 'ExpressGENiE', new files that contain gene expression data are generated and can be visualized on the JBrowse instance available on CHOgenome.org. This now enables users to easily find the expression level of certain genes from publicly-available experiments.

Chapter 1

INTRODUCTION

1.1 Motivation

Recombinant therapeutic proteins, a class of biopharmaceuticals, are drugs derived from the genetic manipulation of living organisms (microorganisms, plant cell cultures, insect and mammalian cell lines) [Kantardjieff and Zhou, 2014; Walsh, 2006]. These proteins have become an important therapeutic option in a variety of human diseases, such as insulin to treat diabetes and interferon (IFN) to treat viral hepatitis [Dingermann, 2008]. Originally, therapeutic proteins were extracted from human or other animals' blood or tissues, such as the extraction of insulin from pigs and cattle [Lens and Evertzen, 1952]. In 1978, the first recombinant therapeutic protein, human insulin, was synthesized via genetic engineering of *Escherichia coli* [Goeddel et al., 1979]. With the advances in cell engineering technology, recombinant proteins manufactured by living cells can now be engineered to have better therapeutic characteristics, such as higher efficacy and lower immunogenicity [Grillberger et al., 2009].

For the production of recombinant proteins, mammalian cells are the predominant host cell platform [Berlec and Štrukelj, 2013]. Of the 211 biopharmaceuticals that gained regulatory approval by July 2014, 30 (14.2%) were produced in yeast, 63 (29.8%) in *E.coli*, and 115 (54.5%) in mammalian cells [Walsh, 2014]. Of the therapeutic proteins produced in mammalian cells, 69 (60%) were produced by Chinese Hamster Ovary (CHO) cell lines, making CHO cell lines the

most important mammalian cell system for recombinant therapeutic protein production [Walsh, 2014].

The prevalence of CHO cells as a preferred production platform for biopharmaceuticals is due to a number of attractive properties, with the most important ones being product safety and manufacturing adaptability. Compared with *E. coli* and yeast, CHO cells have the ability to perform human-compatible post-translational modifications (e.g. glycosylation), which are less likely to cause immune responses, thus reducing product safety concerns [Hammond et al., 2012, Xu et al., 2011, Jayapal et al., 2007]. In addition, CHO cells do not transmit most human pathogenic viruses, and therefore, are considered a safe host [Wurm et al., 2011]. Finally, CHO cells also have a manufacturing adaptability advantage in that serum-free (SF) suspension culture conditions can be used for large-scale commercial therapeutic protein production, increasing process safety and decreasing production costs [Kim et al., 2012,Baik et al., 2011].

1.2 Genome

Methods for enhancing the productivity and quality of recombinant therapeutic proteins produced by CHO cells are of immense interest to biopharmaceutical companies. Genetic modifications (i.e., knock down, knock out, knock in) are important approaches used to design cell lines with increased productivity or with other desirable features for recombinant therapeutic protein production. The CHO and Chinese Hamster (CH) genome sequences provide a basis to facilitate the genetic engineering of CHO cell lines. With the genome sequence and annotation available, it is possible to perform targeted genetic editing in CHO cell lines. The genome sequence can also be used as a foundation to interpret various types of –omics data.

For instance, the genome sequence is often required for mapping RNA sequencing reads in transcriptome studies. The mapping provides a framework to merge sequencing reads into a complete mRNA transcript. Overall, the genome sequence provides an opportunity to better understand CHO cell physiology.

Currently, the CHO-K1 and CH genome information is available at CHOgenome.org [http://www.chogenome.org], which is an online database for CHO-specific sequence information and provides a common resource for the CHO community [Hammond et al., 2012, Kremkow et al., 2015]. The CHO-K1 cell line is the predominantly used ancestral CHO cell line [Xu et al., 2011]. The CHO-K1 cell line 2014 genome annotation contains 27,843 predicted genes and 31,545 transcripts [Kremkow et al., 2015] and its assembly contains 265,786 contigs and 109,152 annotated scaffolds [Xu et al., 2011]. While the CHO-K1 genome information facilitates the study of CHO cells, the inherent genomic instability of CHO cells, which could lead to a wide-range of rearrangements of its genome within two adjacent generations [Xu et al., 2011], makes it impossible to use CHO-K1 genome as the CHO reference genome. The unstable ancestral CHO-K1 genome increases the difficulty of using the CHO-K1 genome to fully represent all CHO cell lines or all CHO-K1 cells in one population. Thus, the CH genome is being used by the community as a key CHO reference genome. The CH genome consists of 24,044 predicted genes, of which 82% are annotated [Kremkow et al., 2015, Lewis et al., 2013].

1.3 Transcriptome

Transcriptomics is a useful approach to understand the biology of CHO cells, especially when a fully assembled genome is available. The transcriptome, which encompasses the full range of messenger RNA molecules in one cell or a group of

cells, is the initial product of genome expression and can subsequently be further translated into the proteome [Brown, 2002]. As transcriptomic data can comprehensively profile gene expression information, expressed genes can be identified [Becker et al., 2011; Birzele et al., 2010] and gene expression levels can be measured through transcriptomic studies. In addition, transcriptomic studies have been used to investigate gene expression changes in combination with proteomic studies. For example, these two approaches have been used together to better understand the mechanism of a metabolic shift which is associated with reduced glucose consumption and lactate production in continuous culture of mammalian cells (mouse-mouse hybridoma cell line) [Korke et al., 2004]. In addition, they have also been used to study the characteristics of high-producing mammalian cell lines, particularly their protein synthesis and growth/death pathways [Seth et al., 2007].

1.4 Visualization of Data

Genomic and transcriptomic data contains a large amount of biological information for CHO cells. Visualization of genome and transcriptome data has many practical applications that can help scientists understand and gain insights from the genome and transcriptome information. For users who have a gene of interest, the visualization of publicly available gene expression data can directly demonstrate how the gene is expressed under different experimental conditions, which can be used for further studies. For example, the visualized gene expression data from previous experiments can be used as a database for users to identify high producing CHO cells and then study their characteristics [Sanny et al., 2006]. Additionally, users can obtain reference expression levels of genes of interest under various conditions from previous

experiments [Yee et al., 2008], which could help users to decide which culture condition to use in their experiments.

There are available genome browsers that enable users to obtain visualization data or visualize their own data. The UCSC genome browser [https://genome.ucsc.edu] displays genome and transcriptome information from many species. Integrative genomics viewer (IGV) [http://software.broadinstitute.org/software/igv/] is another power visualization tool that can be downloaded to the user's computer for private data visualization. In addition, plenty of organism specific databases use web-based genome browsers to visualize relevant data [Wang et al., 2013], such as Rat Genome Database (RGD) [http://rgd.mcw.edu] and Wormbase [http://www.wormbase.org/#012-34-5]. Both the RGD database and Wormbase genome viewers are based on JBrowse. When users are using UCSC genome browser, IGV, or JBrowse to visualize transcriptome data, these tools can only display limited information, such as the track that display genes which were detected by a DNA microarray experiment (Figure 1.1), or visualize a BAM file that displays reads mapping information from an RNA-Seq experiment (Figure 1.2). These data visualization tools are unable to visualize gene expression levels.

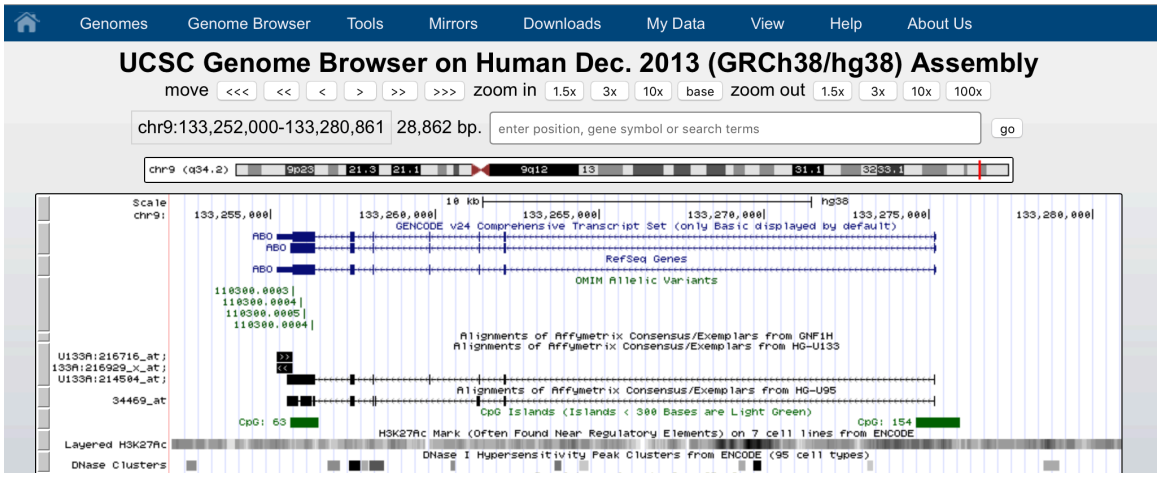

Figure 1.1 Tracks from UCSC genome browser show expressed mRNA transcripts identified by past DNA microarray experiments (Alignments of Affymetrix) [https://genome.ucsc.edu].

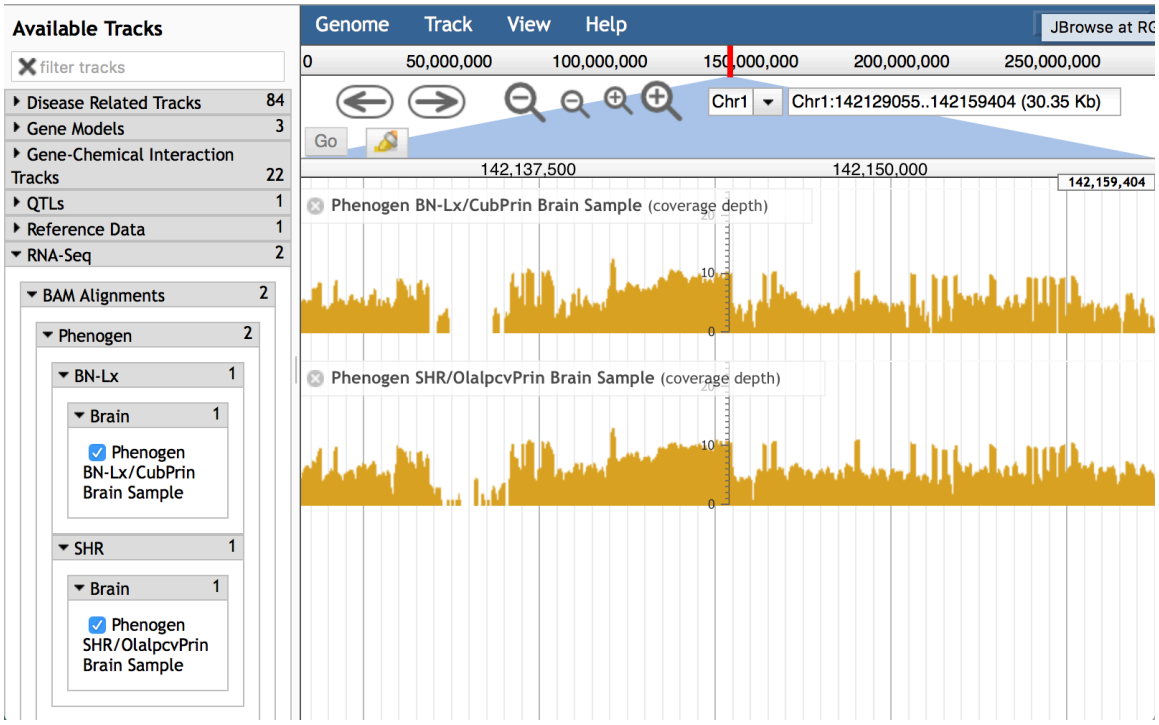

Figure 1.2 BAM files from RGD show read mapping data from RNA-Seq experiments [http://rgd.mcw.edu].

CHOgenome.org has a built-in genome viewer (based on the web-based program JBrowse [Skinner et al., 2009]), which displays the annotated CHO-K1 genome and the chromosome-sorted CH genome [Kremkow et al., 2015]. The CHO-K1 genome is organized by scaffold in JBrowse because scaffolds are the largest segments of continuous sequences and gaps built during genome assembly. Currently, the length of total CHO-K1 sequence is 2,399,786,748 base pairs, which consists of 109,152 scaffolds and a scaffold N50 of 1,147,233 base pairs [Xu et al., 2011]. The selected scaffold displays all genes located within this scaffold, which makes it easy for users to locate the genes of interest and also be aware of the nearby gene information. However, there is currently no tool on CHOgenome.org to visualize transcriptomic data directly using gene expression data. A challenge to implementation is that gene expression data files prevent the direct or clear display of the gene expression levels on a browser.

To visualize the gene expression level from either DNA microarray or RNA-Seq experiments, a platform for converting processed gene expression data from transcriptomics studies to a format that enables visualization of the data in a web-based genome browser is developed in this project. To demonstrate the utility of this tool, publicly available gene expression data is converted to new files compatible with visualization in JBrowse, enabling mRNA expression data to be added to CHOgenome.org.

1.5 Background: DNA Microarray and RNA Sequencing

Currently, there are two main technologies to measure gene expression levels: DNA microarrays and RNA sequencing (RNA-Seq).

Although less commonly used today, DNA microarrays were used in the past to obtain transcriptomic data to study how environmental factors (e.g.: culture temperature, hyperosmotic stress) influenced gene expression in CHO cells [Baik et al., 2006; Shen and Sharfstein, 2006; Vishwanathan et al., 2015]. More recently, with the improvement of sequencing accuracy and cost, RNA sequencing is another option for studying the transcriptome. For instance, RNA-Seq can be used to construct a transcripts database for CHO cells [Rupp et al., 2014], or to study how genes influence gene amplification [Kondratova et al.,2015; Vishwanathan et al., 2015].

1.5.1 DNA microarray

DNA microarray technologies were designed to measure the expression level of thousands of genes within a genome in a single experiment [Trevino et al., 2007]. The basic workflow of DNA microarray technology is shown in Figure 1.3. Each array consisting of thousands of different oligonucleotides (primarily PCR products or oligonucleotides) attached to a solid surface, often glass. These oligonucleotides are used as probes. The complementary DNA prepared from mRNA of the organism of interest is incorporated with fluorescent dyes. Then, the cDNA will hybridize with the probes on the array and the results are detected by laser scanning [Harrington et al., 2000].

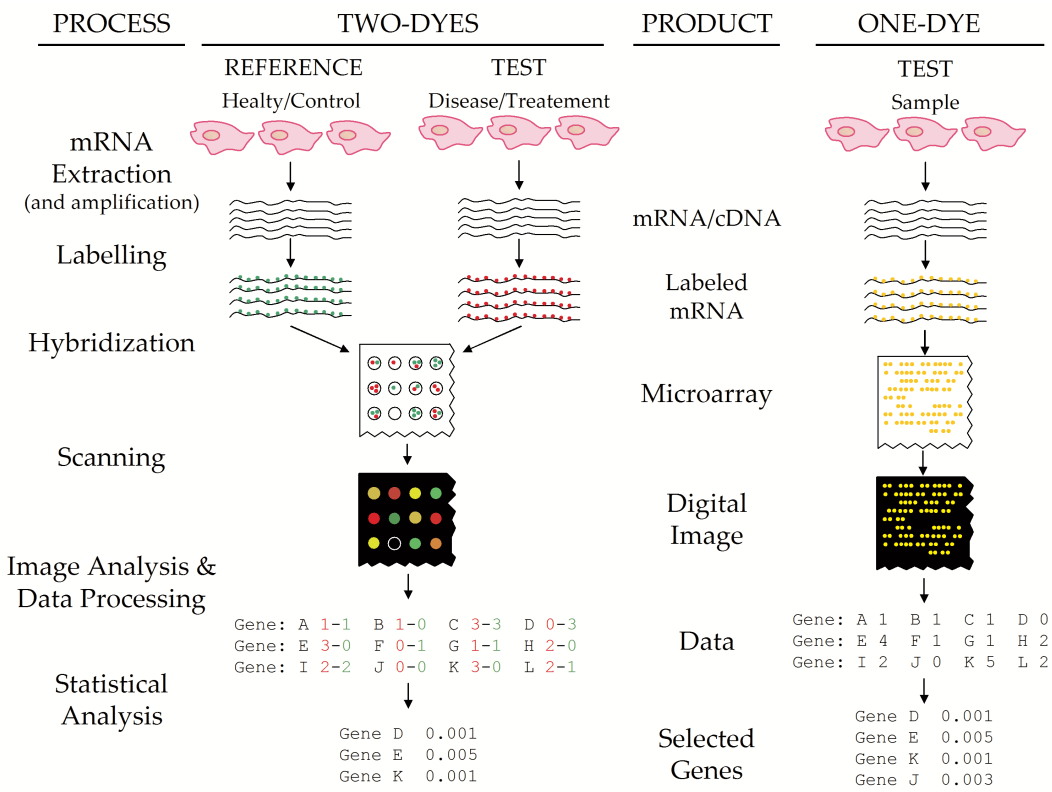

Figure 1.3 Workflow of DNA microarrays [Trevino et al., 2007].

1.5.2 RNA sequencing

RNA-Seq is based on deep-sequencing technologies. The RNA-Seq data can be used to estimate gene expression levels by mapping next generation sequencing (NGS) reads to transcripts [Benjamin et al., 2014] and counting the read density within a region of sequence. The unit is usually FPKM/RPKM (Fragments Per Kilobase Per Million Fragments Mapped/Reads Per Kilobase Per Million Reads Mapped).

The basic workflow of RNA-Seq has several steps (Figure 1.4). The first step is to convert the population of RNA extracted from an organism under a test condition to a library of cDNA fragments with adaptors attached to one or both ends. The adaptors are short sequences that are specific to a platform. The second step is to

sequence the molecules (cDNA with adaptors), which can be done with or without amplification (e.g. PCR), in a high-through manner to obtain short sequences from one end (single-end sequencing) or both ends (pair-end sequencing). Single-end sequencing involves sequencing fragments from one side, while pair-end sequencing is sequencing fragments from both sides (Figure 1.5). Any high-throughput sequencing technology can be used for RNA-Seq. For example, the Illumina IG, also referred to as 'Solexa IG Sequencer', is a next-generation sequencing system provided by Illumina. The last step is either mapping the short sequences to a reference genome or reference transcripts, or assembling the sequences *de novo* without a reference genomic sequence, to produce a genome-scale transcription map that consists of the transcript structure and/or level of expression of each gene. The short sequences, referred to as reads, are typically 30-400 bp, depending on the DNA-sequencing technology used [Benjamin et al., 2014, Wang et al., 2009].

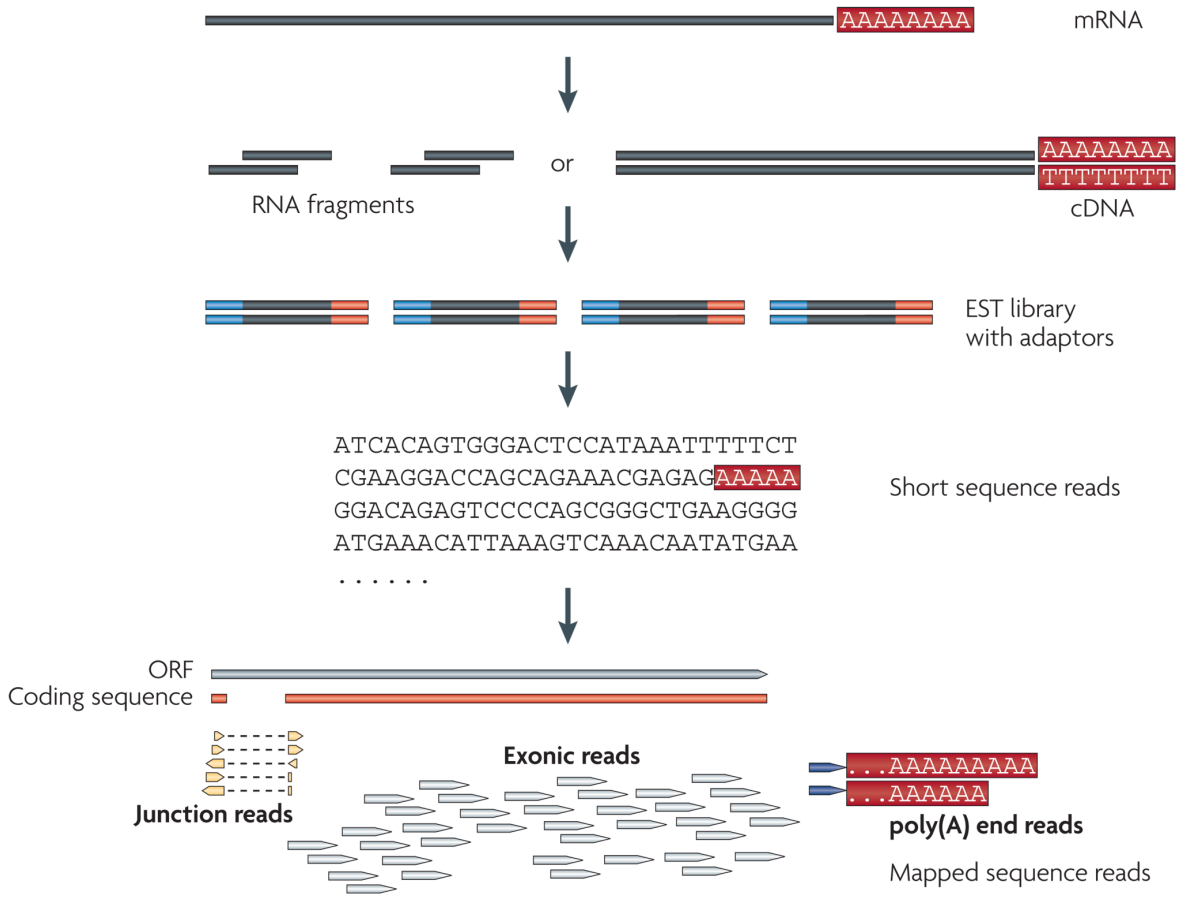

Figure 1.4 Workflow of RNA-Seq [Wang et al., 2009].

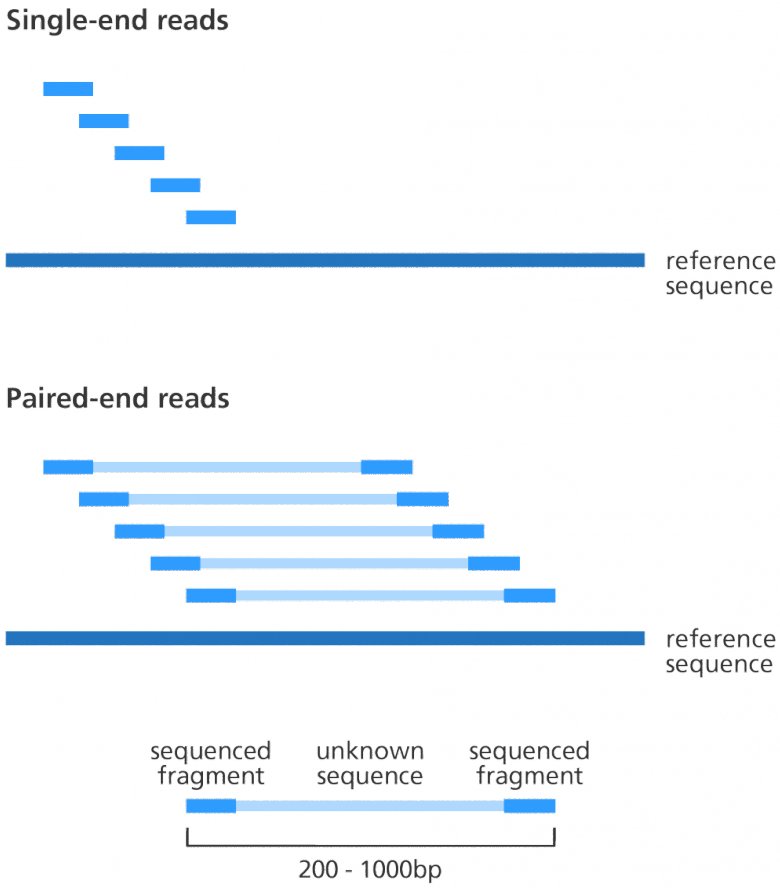

Figure 1.5 Single-end sequencing and paired-end sequencing. [http://www.yourgenome.org/facts/how-do-you-put-a-genome-back-together-after-sequencing]

1.5.3 Comparison of DNA microarray and RNA-Seq

DNA microarray and RNA-Seq technologies each have distinguishing advantages and disadvantages (Table 1.1).

DNA microarrays can be used to directly measure the expression level of a group of target genes at the same time and generate an output file of relatively small size. The DNA microarray output file contains data from the entire dataset and is usually only several megabytes (MB) large. These are remarkable advantages of DNA

microarray technology, which provide users a clear and easy way to analyze the output file [Bumgarner, 2013]. One big limitation of DNA microarray is that it is based on complimentary known genome sequences to detect the expression level of sequences [Bumgarner, 2013], which makes the method unsuitable for samples with unknown genome. Another limitation is that DNA microarray data has higher background noise compared with RNA-Seq, because DNA microarrays lack a direct measure of relative concentration as the detected signal level is not linearly proportional to the concentration of the DNA hybridizing to the microarray [Bumgarner, 2013]. In addition, it is difficult to prevent other homologous sequences from binding to the same probe, which is a common issue for complex mammalian genomes that have multiple related DNA/RNA sequences [Bumgarner, 2013].

In contrast, RNA-Seq sequences the whole transcriptome and generates a large amount of information. Thus, efficient tools are needed to store, retrieve, and process the large amount of output data [Wang et al., 2009]. The RNA-Seq output file is much bigger than DNA microarray output file. Unlike the DNA microarray output file, each RNA-Seq sample has an output file that is usually more than 1 gigabyte (GB). The sequence coverage is another challenge for RNA-Seq, because the number and level of different transcript isoforms are often unknown. Due to the fact that one gene can produce different transcripts, calculating the coverage of the transcriptome is also more complicated than calculating genome-sequence coverage [Wang et al., 2009]. Despite these limitations, RNA-Seq has clear advantages over microarrays for transcriptomics. RNA-Seq can be used to 1) detect transcripts for genomic sequences that have not yet been determined; 2) reveal the sequence variations in the transcribed regions; 3) show the connectivity between multiple exons, and 4) define the precise

location of transcription boundaries with a single-base resolution [Wang et al., 2009]. Another advantage of RNA-Seq is that it has a lower background noise. RNA-Seq does not have an upper limit of what levels can be quantified, which enables measurement of a large range of expression levels [Wang et al., 2009].

Table 1.1 Comparison between DNA microarray and RNA-Seq

Features	**DNA microarray**	**RNA-Seq**
Target	*Genes of interest*	*Whole genome*
Output data size	*Small*	*Big*
Sequence coverage	*No*	*Yes*
Known sequences (basis)	*Yes*	*No*
Background noise	*High*	*Low*

1.6 Background: JBrowse

JBrowse is a genome browser built using standard HTML and Java Script, making it compatible with almost all modern web browsers [Skinner et al., 2009, Westesson et al., 2013, Skinner and Holmes, 2010]. It contains important genome browser features including the ability to select tracks (track refers to file displayed on JBrowse), display the exon-intron structure of genes, and traverse through the genome using a navigation bar (the user interface of JBrowse is shown in Figure 1.6). JBrowse does not need to communicate with a back-end server for moving across the genome, zooming in and out, and reordering tracks. Instead, this functionality is on the client side, which makes these operations much faster, enabling users to easily view and compare data from multiple sources at different points in the genome [Skinner et al., 2009].

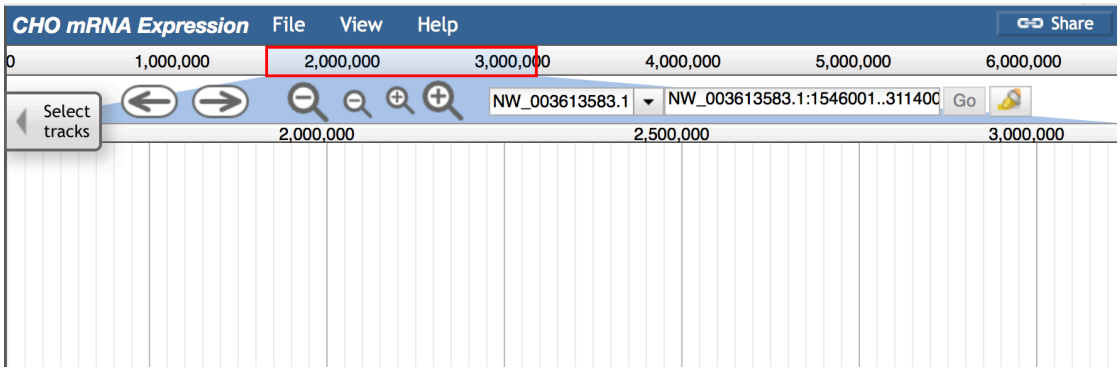

Figure 1.6 The user interface of JBrowse.

JBrowse can be used for visualizing genome annotations and displaying next-generation sequencing data [Westesson et al., 2013, Skinner and Holmes, 2010]. JBrowse can visualize sequence data (nucleotide sequence), feature glyphs (different features, such as exon and intron, can be related), and quantitative data as tracks. In addition, these elements can be combined or viewed together to highlight parts of the data. Currently, JBrowse can display different tracks that contain sequence (FASTA), feature (GFF, BED, GenBank), quantitative (Wiggle, BigWig), alignment (BAM), variant (VCF), and image data [Buels et al., 2016].

1.7 Gap and solution

After the mRNA expression data is collected, bioinformatics tools help to process and visualize the data. Based on the current methods available, DNA microarrays produce numerical data, which are usually used to represent the gene expression level, while RNA-Seq raw data requires other tools for further processing to obtain the actual gene expression levels. There are tools for the visualization of differential gene expression from RNA-Seq data including edgeR [Robinson et al., 2010], DESeq [Anders and Huber, 2010], and Cuffdiff [Trapnell et al., 2012]. These

tools are powerful and can illustrate the differential genes expression among multiple conditions. However, no conversion methods exist to go from the raw DNA microarray or RNA-Seq data to gene expression levels, which are viewable per sample and per dataset in a genome browser for visualizing the gene expression level. Therefore, displaying the gene expression levels of samples from several datasets on a single genome browser instance is the novel goal of this study.

In this study, the current DNA microarray dataset is based on Affymetrix' GeneChip technology. The current RNA-Seq datasets are based on Illumina RNA-Seq technology. The platform is based on custom Python scripts that can generate compatible format files from raw data files (DNA-microarray data) or from the output files of the Tuxedo protocol [Trapnell et al., 2012] (RNA-Seq data).

Chapter 2

MATERIALS AND METHODS

2.1 Materials

The reference genome used in the study is the Chinese Hamster Ovary (CHO) K1 genome. The study involves two different types of data: DNA microarray data and RNA-Seq data. In particular, there is one DNA microarray dataset: GSE30321 [Clarke et al., 2011] and two RNA-Seq datasets: GSE59487 [Kondratova et al., 2015] and GSE75094 [Lee et al., 2016]. The dataset ID is the ID of the dataset used in the Gene Expression Omnibus (GEO) database [http://www.ncbi.nlm.nih.gov/geo/].

Scripts written in Python version 2.7 were used to process these data.

2.2 Method

To visualize mRNA expression data, new files were generated by using custom Python scripts. When processing RNA-Seq data, the Tuxedo protocol [Trapnell et al., 2012] was used with Python scripts.

This study involves multiple different types of files. Two reference files contain basic CHO-K1 gene annotation and scaffold information respectively. Two assistant files were generated in order to help process the data. For each DNA microarray dataset, there are two raw data files and two different types of files were generated for data visualization. For each RNA-Seq dataset, there is one raw data file per sample and five different types of files were generated for data visualization. For the entire DNA microarray data type or the entire RNA-Seq data type, there is one file

per data type generated for data visualization. Additionally, one file was generated for data visualization, which combined general information from both data types. These files are summarized in Table 2.1. The term ‘track’ is used to refer to the corresponding data of each file shown on JBrowse.

Table 2.1 Summary of files

File Source	File Name/Format	Description
Reference files	*CHO-K1-v1.refseq_2014.annotation.gff3*	*The annotation GFF3 file contains the annotation of the genome assembly from the Chinese hamster ovary cell line CHO-K1*
	CHO-K1eweswssss-v1.scaffolds.fasta	*The FASTA file contains the genome sequence from the Chinese hamster ovary cell line CHO-K1*
	Scaffold size text file	*Contains scaffold ID and scaffold size*
	Gene symbol GTF file	*Contains gene ID, symbol and position for each gene*
DNA microarray dataset	*Raw table text file*	*Contains the raw data results*
	Target gene ID/Symbol/Name text file	*Contains target genes' commonly used Symbol and the probe ID that is unique to the dataset*
	Sample text file	*Contains the gene symbols, gene positions, and gene expression levels within each sample*
	BigWig file	*Used to visualize the expression level for target genes*
	Dataset GTF file	*Used to visualize the identification/symbol (commonly used) for target genes*
RNA-Seq dataset	*Raw reads file*	*Contains the raw reads information*
	Sample BAM file	*Contains reads mapping information*
	Sample GFF3 file	*Contains information about annotated assembly transcripts*
	Sample bigWig file	*Contains gene expression level information.*
	Dataset GTF file	*Contains differentially expressed genes information*
	Dataset GFF3 file	*Contains the source for each part of a sequence*
DNA microarray or RNA-Seq	*Data type GFF3 file*	*Contains the source (dataset ID) for each part of a sequence*
DNA microarray & RNA-Seq	*Overall GFF3 file*	*Contains the source (data type) for each part of a sequence*

2.2.1 Reference tracks

The reference track files 'CHO-K1-v1.refseq_2014.annotation.gff3' and 'CHO-K1-v1.scaffolds.fasta' were obtained from CHOgenome.org. The JBrowse application was used to generate the reference tracks and to visualize these files. The reference gene and the reference mRNA tracks were generated from the 'CHO-K1-v1.refseq_2014.annotaion.gff3' file, while the reference DNA track was generated from the 'CHO-K1-v1.scaffolds.fasta' file.

2.2.2 Assistant files

Two assistant files, consisting of a text format file and a GTF format file, were created to help generate or modify files in this study. The reference 'CHO-K1-v1.scaffolds.fasta' file is the foundation for generating the assistant text format file that contains scaffold sizes. The reference 'CHO-K1-v1.refseq_2014.annotation.gff3' file is the basis for generating the assistant GTF format file that contains gene symbols. Each file was generated using custom Python scripts (for details see Chapter 3 Results, section 3.1.2.1).

The scaffold size text format file has two columns per line, one for scaffold ID and another for scaffold size. In the 'CHO-K1-v1.scaffolds.fasta' file, each scaffold has one row to record scaffold ID information and the following rows to list the nucleotide sequence of the scaffold in single-letter code (see Appendix A for the specific format). To generate the scaffold size file, a Python script was used to count the nucleotide sequences and to record the length and ID of scaffolds.

The gene symbol GTF file (nine columns per line) stores gene positions and symbols. In the 'CHO-K1-v1.refseq_2014.annotation.gff3' file, there are nine columns per line that contains metadata for each gene/exon/CDs (coding sequence). Therefore,

a Python script was used to generate the gene symbol GTF file by selecting and recording only ‘gene’ features with their gene symbol.

2.2.3 DNA microarray dataset

For DNA microarray datasets, a GTF file and bigWig files were generated using Python scripts (for details see Chapter 3 Results, section 3.1.2.2).

Each DNA microarray dataset has two files that are essential to process its data, the ‘raw table text file’ and the ‘target gene ID/Symbol/Name text file’. Generally, DNA microarray raw data are stored in a text format file that is referred to as the ‘raw table text file’. This file contains probe IDs used in the dataset and all genes’ expression levels from all samples. A supplementary file, referred to as the ‘target gene ID/Symbol/Name text file’, includes a table, which contains the probe IDs used in the dataset and the gene symbols. These two files can be downloaded from the GEO database.

To process the data more efficiently, the ‘raw table text file’ was separated into a series of sample text files (using the sample ID recorded in the ‘raw table text file’ as the new file name). All sample text files were saved in a single dataset folder (users can name the folder by setting the ‘-d’ parameter). As the ‘CHO-K1-v1.refseq_2014.annotation.gff3’ file’ contains gene positions and gene symbols, Python scripts were used to generate the folder and the text file for each sample (for details see Chapter 3 Results, section 3.1.2.2.1). This was accomplished by combining the gene symbols from the ‘target gene ID/Symbol/Name text file’, gene positions from the ‘CHO-K1-v1.refseq_2014.annotation.gff3’ file’ and gene expression values from the ‘raw table text file’. The output sample text file consists of five columns per line: gene symbol, scaffold ID, start position, end position, and gene expression value.

BigWig files were chosen to visualize the gene expression information for each dataset. If the dataset has small number of samples (e.g. ≤20), then a bigWig file is generated for each sample individually. In contrast, if the dataset has a large number of samples (e.g. >20), two options are provided for generating the bigWig files based on the sample culture conditions and genetic engineering modifications. If it is possible to group samples by similar conditions or modifications, one bigWig file for each group is generated which stores the average gene expression level among the samples belonging to the respective group. On the contrary, if it is not possible to group the samples, the dataset will be treated as one group. Thus, one bigWig file is generated for the entire dataset.

The bigWig file format was chosen because it can directly display two key values: the gene location and the gene expression level. Visualizing the data of bigWig files on JBrowse are shown as blocks, where each block represents an expressed gene (in Figure 2.1, the track in red box is the reference gene track). The position of the block indicates the location of the gene, and the height of the block represents the gene expression level. The average expression value was chosen because there are various conditions of samples and it provides a good summary of how a gene was expressed for a given condition/dataset.

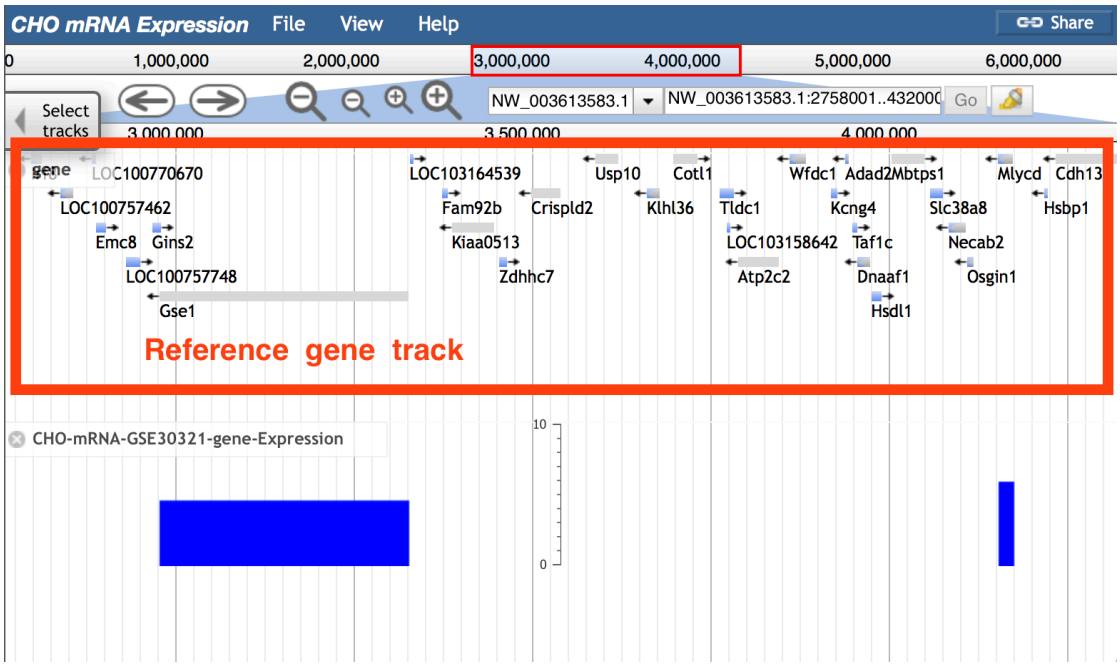

Figure 2.1 The bigWig file of DNA microarray data shown on JBrowse.

Generating a bedGraph file is a prerequisite to obtaining a bigWig file. The bedGraph format, which stores the same information as the bigWig file, is a human readable file format for the binary bigWig file. The bedGraph/bigWig file consists of four columns per line, which records the gene's location and expression value. The four columns are scaffold ID, start site, end site, and gene expression value. To convert the bedGraph format to the bigWig format, 'bedGraphToBigWig', was download from ENCODE [https://www.encodeproject.org/software/bedgraphtobigwig/], and used.

While a bigWig file provides the position and expression value of genes, it can not directly display the gene symbol/ID. Accordingly, a GTF file for each DNA microarray dataset was generated to indicate the gene symbol. In the dataset GTF file, each line records the metadata for an expressed gene that contains the gene symbol/ID

(consistent with the gene symbol/ID in reference file) and the PMID (PubMed ID) of the related paper. The data of the GTF file shown on JBrowse consists of colored blocks that represent expressed genes (Figure 2.2). The block position is the gene position, and the displayed gene symbol/ID indicate the gene symbol. Python scripts (for details see Chapter 3 Results, section 3.1.2.2.2) and two input files were used to generate this GTF file: the assistant gene symbol GTF file and one of the bedGraph files of the dataset.

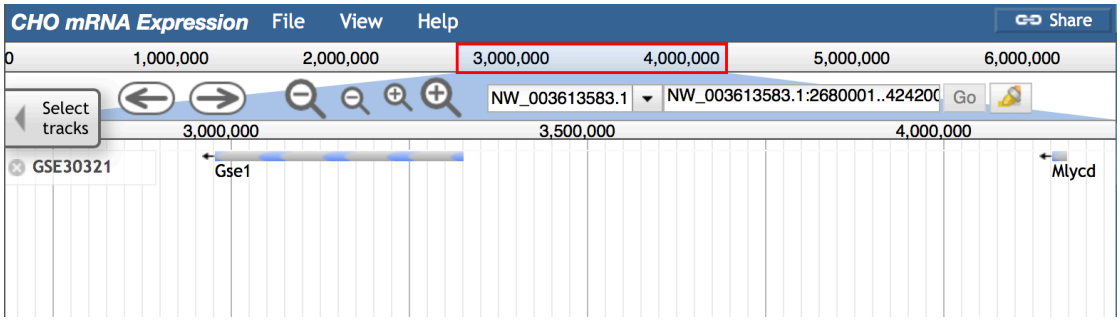

Figure 2.2 The GTF file of DNA microarray data shown on JBrowse.

2.2.4 RNA-Seq dataset

For RNA-Seq data, there are five file types: a BAM file, an annotation GFF3 file, and a FPKM (Fragments Per Kilobase per Million mapped reads) bigWig file for each sample, a GTF file and an annotation GFF3 file for each dataset. To generate all of these files, the Tuxedo protocol [Trapnell et al., 2012] and Python scripts were used to process the data.

In each RNA-Seq dataset, there are files for visualizing samples individually. RNA-Seq raw data (reads information) is usually stored in a FASTQ (for details see Appendix A) file. Many tools included in the Tuxedo protocol, Tophat, Cufflinks,

Cuffmerge, and Cuffdiff (this is also the order for running these tools) were chosen and run. Tophat maps reads to the genome. Cufflinks assembles reads to exons and transcripts. Cuffmerge and Cuffdiff compare gene expression conditions between any paired groups. Some of the output files were selected and used in further steps, including: BAM files from Tophat, assembly GTF files and text format files (with suffix 'fpkm_tracking') from Cufflinks, the merged GTF files from Cuffmerge, and the text format files (with suffix 'diff') from Cuffdiff.

The Tophat output BAM file stores mapping information, including the read mapping location and mapping quality. The BAM file is compatible with JBrowse. Two versions of the BAM file are shown on JBrowse: the alignment BAM track and the coverage BAM track (Figure 2.3). The alignment BAM track indicates the position, direction, and match/mismatch information of the mapped reads. It uses colored dots to represent reads. The position of each dot represents the read mapping location, where red usually represents a positive strand, blue represents a negative strand, and black represents a mismatched sequence inside a read. The coverage BAM track displays the density of reads aligned to each position. The appearance and usage of the coverage BAM track are similar to the bigWig track. The whole track consists of several blocks. The height of each block represents the density of reads in that position with a density value.

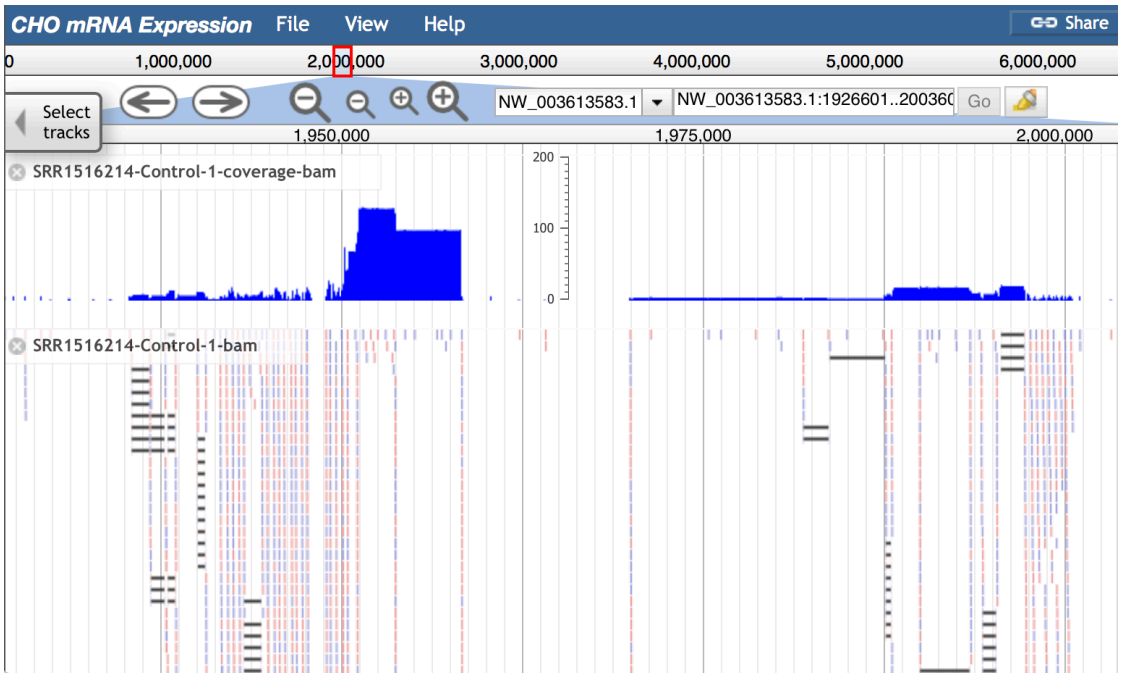

Figure 2.3 The BAM file of RNA-Seq data shown on JBrowse.

Cufflinks generates the assembled GTF file and the FPKM tracking file. The GTF file contains assembled transcripts information, including location of exons and transcripts, and the gene expression value. Cufflinks generates the assembly based on the reference 'CHO-K1-v1.refseq_2014.annotation.gff3' file. Accordingly, the assembled GTF file records all possible transcripts, which includes both expressed and unexpressed transcripts. Because the expressed and unexpressed gene cannot be distinguished from the assembled GTF file when it shown on JBrowse. Custom Python scripts were used to generate a new annotation GFF3 file to replace the assembly GTF file (for details see Chapter 3 Results, section 3.1.2.3.2). Two input files are required to initiate the scripts: the assembled GTF file and the assistant gene

symbol GTF file. The new annotation GFF3 file contains annotation information for the only expressed exons and transcripts (Figure 2.4).

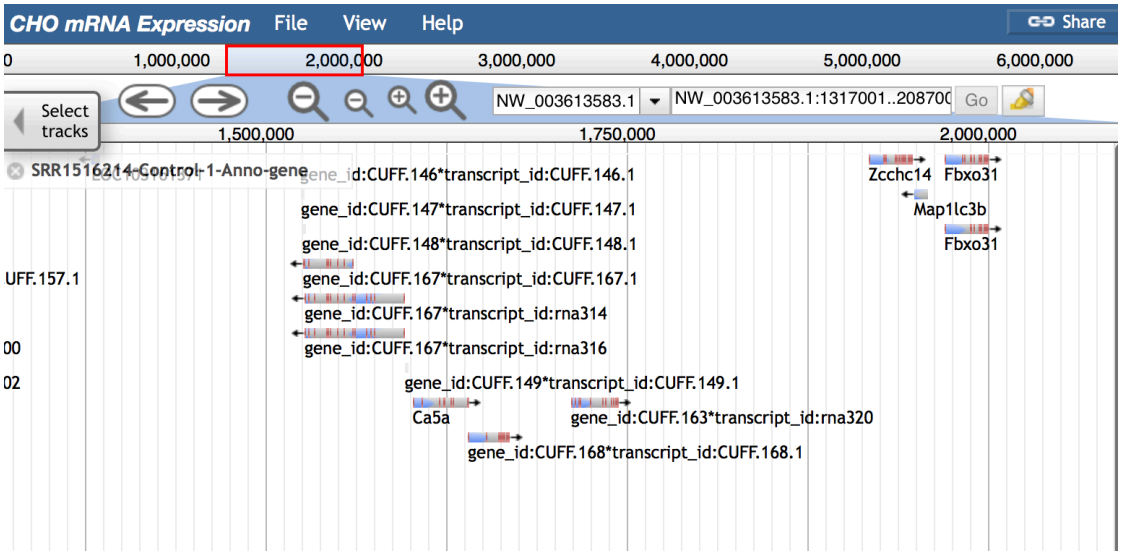

Figure 2.4 The annotation GFF3 file of RNA-Seq data shown on JBrowse.

The FPKM tracking file is a text format file that contains sequence expression values. Cufflinks generates two FPKM tracking files: a transcript-level (isoform) FPKM tracking file (contains all transcripts and its expression level) and a gene-level FPKM tracking file (contains all genes and its expression level). The gene expression level in the gene-level FPKM tracking file is the summation of expression level for all transcripts of this gene. As all transcripts and genes are derived from reads, the data in gene-level FPKM tracking file contains all reads information for that gene, so the gene-level FPKM tracking file was selected for further use. To visualize the gene expression information in the FPKM tracking file (which is incompatible with JBrowse), Python scripts were used to build a new FPKM bigWig file (for details see

Chapter 3 Results, section 3.1.2.3.3). To generate the bigWig file, or the human readable bedGraph file, two input files are required: a gene-level FPKM tracking file and the assistant scaffold size text file. The new FPKM bigWig file displays the measurement value of the gene expression level when users hover their cursor over the gene (In Figure 2.5, the track in red box is the reference gene track).

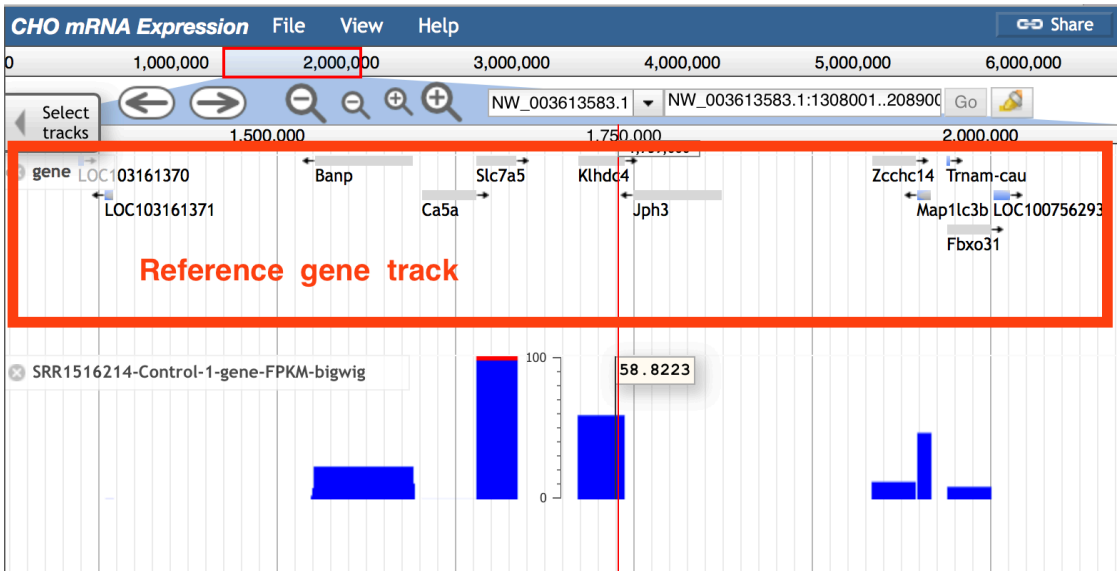

Figure 2.5 The FPKM bigWig file of RNA-Seq data shown on JBrowse.

Each sample has a BAM file, the annotation GFF3 file, and the FPKM bigWig file. To compare gene expression among different conditions, Cuffmerge and Cuffdiff were used. Cuffmerge is a preparation step for running Cuffdiff and its function is to merge the assembled GTF files that come from each sample into a single merged GTF file. This merged GTF file contains all transcripts and exons information and works as a reference file in Cuffdiff. The function of Cuffdiff is to compare the gene expression level between paired conditions. It also can make comparisons among several

conditions by comparing gene expression between each paired conditions separately. In addition to the merged GTF from Cuffmerge, BAM files that come from Tophat for each sample are required. Cuffdiff maps all reads into the merged GTF file and then calculates a FPKM value for each gene. Afterwards, Cuffdiff calculates the expression differences of each gene between paired conditions. Finally, Cuffdiff generates an output text format file that contains all comparison information between paired conditions, which includes each condition's gene expression value, and fold change as well as the p-value between two conditions. Because the comparison output text file is incompatible with JBrowse, a differential GTF file that records all the genes that show a significant difference between paired conditions was generated by custom scripts. There are two options to generate the differential GTF file: select by p-value and select by fold change (for details see Chapter 3 Results, section 3.1.2.3.4). If the selection by p-value option is chosen, the Cuffdiff provides a p-value for each comparison that is compared against an alpha value of 0.05. If the selection by fold change option is chosen, a parameter 'threshold' is used to do the selection. The differential GTF file shown on JBrowse has normal GTF file appearance, but each visible block represents that the gene in that position is expressed differently between two conditions (Figure 2.6). When the user clicks the blocks, detailed information, such as gene expression level under each condition and groups name/ID, are displayed in a pop-up window.

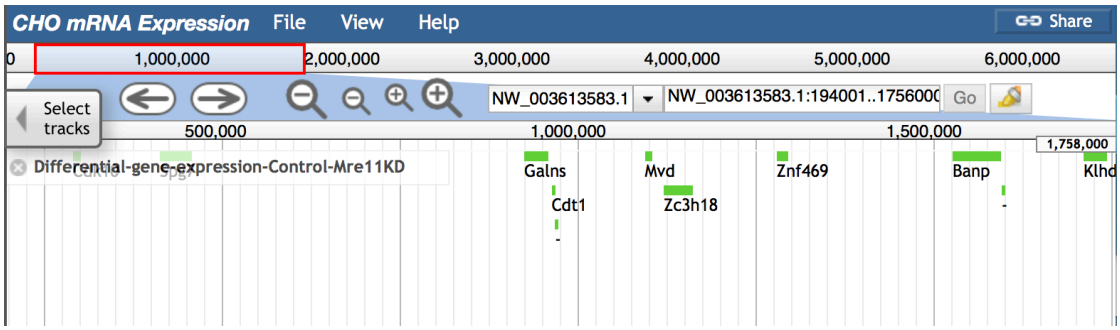

Figure 2.6 The differential GTF file of RNA-Seq data shown on JBrowse.

The GFF3 file for the entire RNA-Seq dataset was generated by merging all the annotation GFF3 files that from each sample. Since the GFF3 file is similar to the GTF file in content, the appearance and usage of the GFF3 file on JBrowse are similar to that of the GTF file. On JBrowse, the GFF3 file also consists of colored blocks, though here the blocks in the GFF3 file have two colors (Figure 2.7). The different colored blocks are used to distinguish the sources of the data. Green block represents the sequence found to be expressed in all samples, whereas yellow block represents the sequence found to be unexpressed in at least one sample. When users click the block, the pop-up window provides metadata of the gene, containing gene symbol/ID (a ID generated by scripts), the data source, and other information. Python scripts (for details see Chapter 3 Results, section 3.1.2.3.5) and two assistant files (the scaffold size text file and the gene symbol GTF file) were used to generate this GFF3 file.

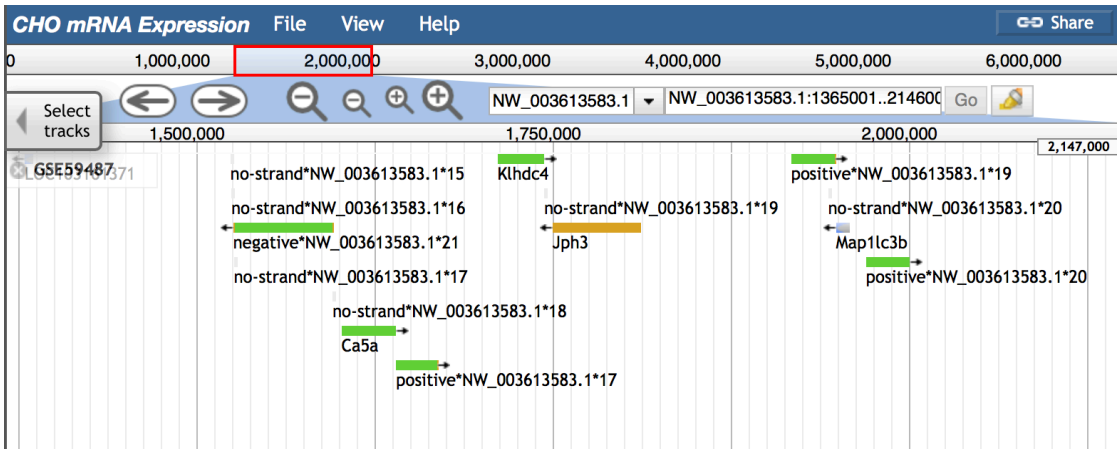

Figure 2.7 The dataset GFF3 file of RNA-Seq data shown on JBrowse.

2.2.5 Data type track and overall track

The GFF3 file for each data type (DNA microarray or RNA-Seq) is similar to the GFF3 file for the RNA-Seq dataset (Figure 2.8). The difference is that the source of data in data type track is the dataset ID instead of the sample ID in the dataset track. The data type track provides a convenient method for users to go through their data type of interest. For users who do not have data type of interest, another annotation GFF3 file referred as 'Overall track' was generated and displayed on JBrowse (Figure 2.9). This track contains information from all datasets from both DNA microarray and RNA-Seq data types. Two assistant files (the scaffold size text file and the gene symbol GTF file) and Python scripts are required to generate the data type GFF3 file and 'Overall track' (for details see Chapter 3 Results, section 3.1.2.4).

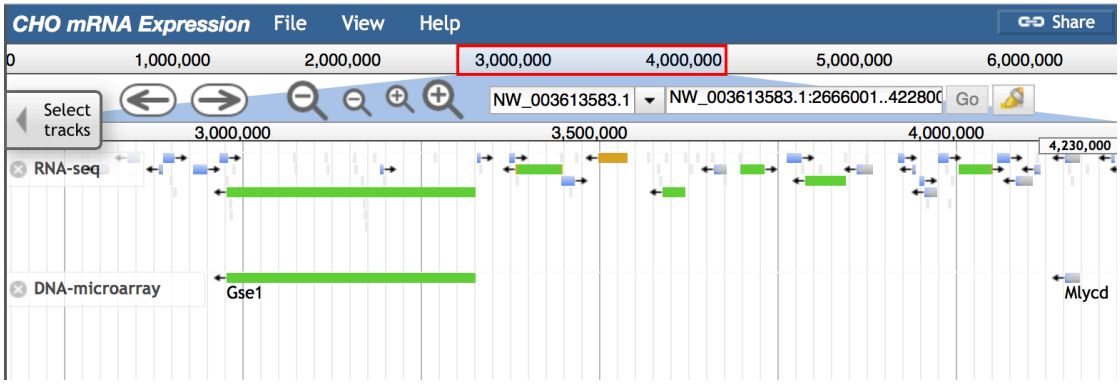

Figure 2.8 The data type GFF3 files shown on JBrowse.

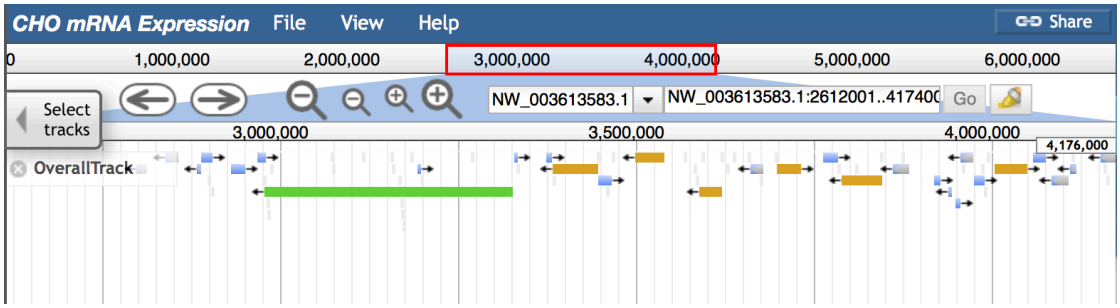

Figure 2.9 The overall GFF3 file shown on JBrowse.

2.2.6 Optional multiple bigWig XY track

The multiple bigWig XY track can display bigWig tracks from all samples within the same dataset in one track by selecting a unique color for the bigWig track of each sample (Figure 2.10). For either a DNA microarray or a RNA-Seq dataset, a multiple bigWig XY track can be generated if the bigWig file that contains gene expression values for each sample is provided.

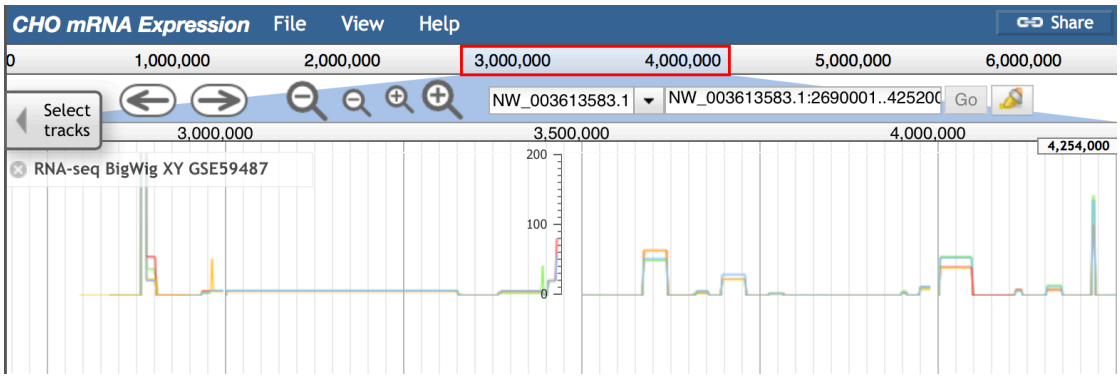

Figure 2.10 The multiple bigWig XY track shown on JBrowse.

Chapter 3

RESULTS

3.1 Files

To generate files, run ExpressGENiE from the command line as follows:

```
Python ExpressGENiE.py [options] <file>
```

The following is a detailed description of each option used to control ExpressGENiE.

3.1.1 ExpressGENiE general options

`-h/-help`

Prints the help message and exits.

`-i input file`

This option is used to provide name of the input file. E.g. `-i CHO_reference.gff3`.

`-r raw_data_file.txt`

This option is used when processing DNA microarray data. This file contains all the raw data: the probe IDs and gene expression levels.

`-t target_gene_file.txt`

This option is used when processing DNA microarray data. This file contains the target gene symbols and the probe IDs.

`-o output file`

This option is used to provide the name of the output file. E.g. `-o merged.gtf`.

`-d input/output directory`

This option tells ExpressGENiE to process the files that are inside a certain folder, or it can name an output folder.

`--list list.txt`

This option tells ExpressGENiE to process the files that have been recorded in the text format file. The text format file contains a list of files. All files will have the path: path/FileName.suffix

`-gn gene_symbol.gtf`

Is a parameter that lets ExpressGENiE use the assistant GTF file that contains the gene symbol information. This gene symbol GTF file provides the gene symbol for many functions.

`-sf scaffold_size.txt`

Is a parameter that lets ExpressGENiE use the assistant text file that contains the scaffold size information. This scaffold size text format file provides the scaffold size or scaffold ID for many functions.

`-pcn <int> probe ID column number`

This option is used to record the column number that contains probe ID in the target gene ID/Symbol/Name.txt file.

`-gcn <int> gene symbol column number`

This option is used to record the column number that contains gene symbol in the target gene ID/Symbol/Name.txt file.

`-cR <int> Choose RNA-Seq`

When processing RNA-Seq data, this option tells ExpressGENiE the data type is RNA-Seq and selects the correct function to run. This option has only one possible value: 1.

`-cD <int> Choose DNA microarray`

When processing DNA microarray data, this option tells ExpressGENiE the data type is DNA microarray data and selects the correct function to run. This option has only one possible value: 1.

`-c <int> Choose step`

This option is used to guide users through the selection of certain steps when processing RNA-Seq or/and DNA microarray data. It provides three integers: 1, 2, and 3. Each integer will lead ExpressGENiE to run a specific function. Default: 0.

`-s Source ID`

This option is used to assign a source ID to the dataset. The source ID is usually the PubMed ID and the PMC ID (if available). The star sign is used to connect these two IDs. E.g. `-s PMID:#####*PMCID:#####`. In the function of merged data from two data types, the source ID should be ExpressGENiE name.

`-si Sample ID`

This option is used to record the sample ID for each sample. The sample ID can be found from the GSE database and is used to distinguish files that are from the same/different datasets. E.g. `-si SRR#####`.

`-di Dataset ID`

This option is used to record the dataset ID for each dataset. The dataset ID can be found from the GSE database and is used to distinguish datasets. E.g. `-di GSE#####`.

`-cg Category of data`

This option is used to name the expected sequence (usually the longest sequence) that will be selected. The selected category will be merged into a longer sequence if there are any overlapping regions. E.g. `-cg transcript`.

`-Ncg New category of data`

This option is used to distinguish the source of the sequence when preparing to make an annotation GFF3 file. If the sequence/region can be found in all samples/sources, then the sequence/region is a new category. E.g. `-Ncg all_transcripts`.

`-pa Parent category of data`

This option is used when preparing to make an annotation GFF3 file. It is used to name the parent region, which keeps child regions that are from different sources. E.g. `-pa sequence`.

`-a <int> Amount of merged items`

The merged item can be samples, datasets or data types. This option is the number of merged items, which can used to distinguish the features (or the categories) set in parameter '`-cg`' and '`-Ncg`'. E.g. `-a 4`.

`-th Threshold`

This option gives a threshold for selection of the differential gene expression level. If the threshold is based on log2(fold_change), the gene will be selected as having differential expression among different conditions only when the value of calculated log2(fold_change) is larger than the threshold. E.g. `-th 2`.

`-tp Data type`

This option is used to distinguish DNA microarray data and RNA-Seq data. E.g. `-tp DNA microarray`.

3.1.2 Arguments

In the ExpressGENiE scripts, there are several different functions. By using positional arguments, the user can select which function to use.

3.1.2.1 Assistant files

The two assistant files are generated from reference 'CHO-K1-v1.refseq_2014.annotation.gff3' file and 'CHO-K1-v1.scaffolds.fasta' file, respectively. These files are used to help generate other files that can be shown on JBrowse.

3.1.2.1.1 Scaffold size text file

The scaffold size text file is a text format file that contains the scaffold ID and its scaffold size. From the command line, run the script as follows to generate the scaffold size text file:

```
python ExpressGENiE.py Scaffoldsize -i <input.fasta> -o <output.txt>
```

Input files:

`-i <input.fasta>`. The reference genome FASTA format file is the input file.

Output file:

`-o <outputfile>`. Writes the scaffold sizes into the text format file.

The 'Scaffoldsize' function produces a text file, named whatever you use in the '`-o`' option. The scaffold size text format file contains two columns per line, one for scaffold ID and another for scaffold size. The columns are defined as follows in Table 3.1.

Table 3.1 Columns in the Scaffold size text file

Column number	Column name	Description	Example
1	*Scaffold ID*	*The Scaffold ID recorded in the reference FASTA file*	*NW_003613580.1*
2	*Scaffold size*	*The length of the scaffold*	*8779783*

An example is shown below:

```
NW_003613580.1   8779783
NW_003613581.1   8081566
NW_003613582.1   6666273
…
```

3.1.2.1.2 Gene symbol GTF file

The gene symbol GTF file is a GTF format file that contains gene symbol information. In the GTF file, all gene symbols are recorded in the last column. The prior columns contain the location information for each gene. From the command line, run the script as follows to generate gene symbol GTF file:

`python ExpressGENiE.py Genesymbol -i <input.gff3/gff> -o <output.gtf>`

Input files:

`-i <input.fasta>`. Provide the reference genome annotation GFF3 file as the input file.

Output files:

`-o <outputfile>`. Writes the gene symbol into the GTF format file.

The 'Genesymbol' produces a GTF file, named whatever you give in the '`-o`' option. The gene symbol GTF format file contains nine columns per line. The columns are defined as follows in Table 3.2.

Table 3.2 Columns in the gene symbol GTF file

Column number	Column name	Description	Example
1	*Scaffold ID*	*The Scaffold ID recorded in the reference GFF3 file*	*NW_003613580.1*
2	*Source*	*The name recorded in the input file*	*Gnomon*
3	*Feature*	*All features are 'gene'*	*gene*
4	*Start*	*The leftmost coordinate of this gene*	*387941*
5	*End*	*The rightmost coordinate of this gene*	*388591*
6	*Score*	*All scores are '.'*	*.*
7	*Strand*	*The strand of the gene. Always one of '+', '-', '.'*	*+*
8	*Phase*	*All phases are '.'*	*.*
9	*Attributes*	*The gene symbol GTF file only contains the gene ID and gene symbol information in the attributes column*	*ID=gene0;Name=LOC100754456;*

The gene symbol GTF file keeps only 'gene' features and indicates the gene IDs and symbols (the item 'Name' is the gene symbol). An example is shown below:

```
NW_003613580.1  Gnomon  gene  387941  388591  .  +
    .  ID=gene0;Name=LOC100754456;
NW_003613580.1  Gnomon  gene  690691  691138  .  +
    .  ID=gene1;Name=LOC100757647;
NW_003613580.1  Gnomon  gene  1834571  1855181  .  -
    .  ID=gene2;Name=Gata3;
…
```

3.1.2.2 DNA microarray dataset

There are two different types of files for each DNA microarray dataset. All examples used in the following files and figures are derived from DNA microarray dataset GSE30321.

3.1.2.2.1 DNA microarray bedGraph/bigWig file and bigWig track

DNA microarray data can represent relative gene expression levels for the genes that are hybridized with probes. BigWig files were used to display continuous quantitative data and to visualize the gene expression level in JBrowse. It keeps concise information for each gene: the location and the expression level.

The number of bedGraph/bigWig files created for each DNA microarray experiment is varied. It is based on the classification of the samples' groups. However, each DNA microarray dataset has at least one bedGraph/bigWig file for the dataset.

The bigWig/bedGraph format file contains the gene expression value for each gene. For each DNA microarray dataset, if the dataset has a small number of samples (E.g. ≤20), then a bigWig file is generated for each sample individually. In contrast, if the dataset has a large number of samples (E.g. >20), two options are provided for generating the bigWig files based on the sample culture conditions and genetic engineering modifications. If it is possible to group samples by similar conditions or genetic modifications, one bigWig file for each group is generated which stores the average gene expression level among the samples belonging to the respective group. On the contrary, if it is not possible to group the samples, the dataset will be treated as one group. Thus, one bigWig file is generated for the entire dataset.

To generate the binary bigWig format file, users need to generate a readable bedGraph file first, and then convert the format by using the 'bedGraphToBigwig' software (https://www.encodeproject.org/software/bedgraphtobigwig/). There are three steps to generate the bedGraph format file as shown in Figure 3.1:

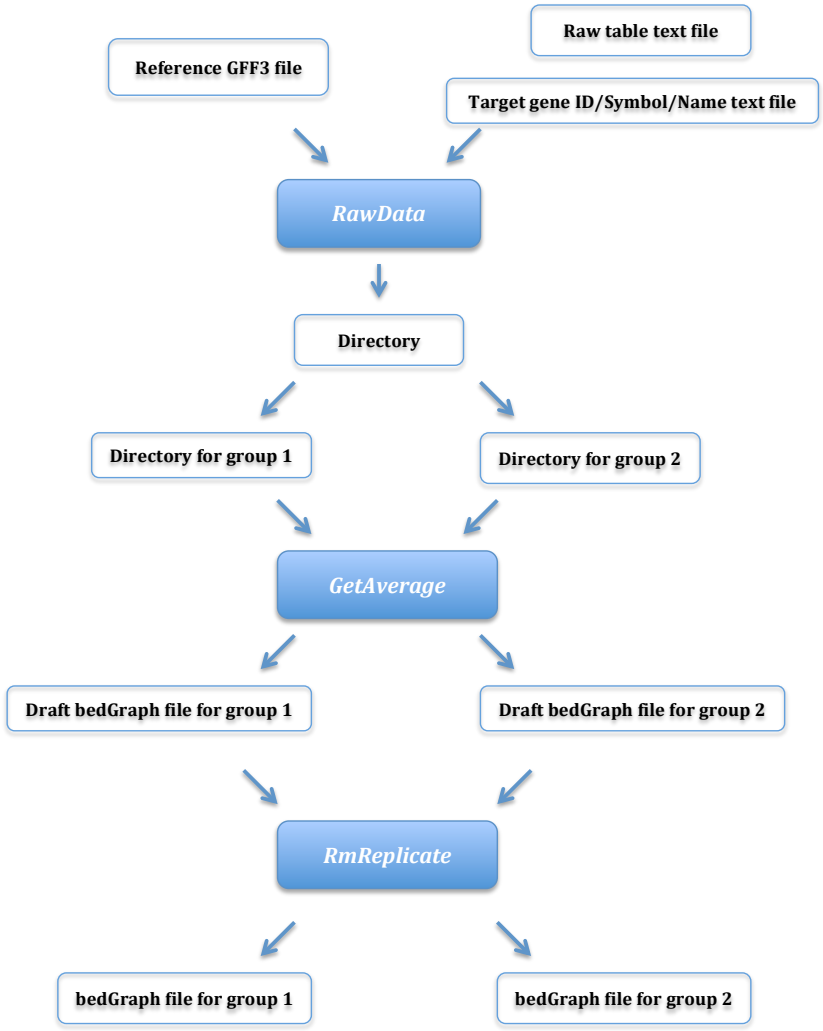

Figure 3.1 The workflow for generating bedGraph file (DNA microarray).

Step 1. Generating a text file for each sample

The raw DNA microarray data file contains the probe IDs and gene expression values. To process the data, a text format file for each sample was generated containing the genes' positions and expression values. The argument to do this is

'RawData'. Follow the command line to generate a directory that contains an output file for each sample:

```
python ExpressGENiE.py RawData -i <input.gff3> -r <raw table.txt> -t <target gene ID/Symbol/Name.txt> -pcn <column number of probe ID> -gcn <column number of gene symbol> -d <output directory>
```

Input files:

`-i/<input.gff3>`. The input file is the GFF3 reference gene file that contains gene position, direction, and name/ID/symbol information.

`-r/<raw table.txt>`. This file is the text format file that contains the DNA microarray raw data (the microarray specific probes' ID and genes' expression values).

`-t/<target gene ID/Symbol/Name.txt>`. This file is a supplementary file for the raw dataset. It contains all probe IDs and target gene symbols.

Output file:

`-d/<output directory>`. This option is used to name the directory that holds all output files.

Each output file (one output file per sample) is a text format file that consists of five columns per line. These columns are defined as follows in Table 3.3.

Table 3.3 Columns in the gene expression level text file for each sample

Column number	**Column name**	**Description**	**Example**
1	*Gene symbol*	*The gene symbol of the target gene*	*Gpaa1*
2	*Scaffold ID*	*The Scaffold ID indicates where the target gene is located*	*NW_003614411.1*
3	*Start*	*The leftmost coordinate of this gene*	*426982*
4	*Stop*	*The rightmost coordinate of this gene*	*431123*
5	*Value*	*The gene expression value*	*6.453162979*

Step 2: Generating a draft bedGraph file for each group

Put all text files (gene position and expression value text file) that belong to the same group into a directory, and use the argument: `GetAverage`. From the command line, run the script as follows to generate the primary bedGraph format file:

```
python ExpressGENiE.py GetAverage –d <input directory> -o <output.bedgraph>
```

Input files:

`-d/<input directory>`. The input directory containing text files that are composed of gene positions and expression values. The text files are the files generated in step 1.

Output file:

`-o/<output.bedgraph>`. The output file is the draft bedGraph file.

Step 3: Removing replicate values for the same gene

In the bigWig/bedGraph file, only one value is kept for each gene. In this step, use the argument '`RmReplicate`' to keep only one average expression value for the gene. From the command line, run the script as follows to generate the final bedGraph format file:

`python ExpressGENiE.py RmReplicate -i <input.bedgraph> -o <output.bedgraph>`

Input file:

`-i/<input.bedgraph>`. The input file is the draft bedGraph file produced in the prior step (step 2). For each group, there is only one bedGraph file.

Output file (DNA microarray bigWig/bedGraph):

`-o/<output.bedgraph>`. Writes the genes' position and expression value into the bedGraph format file.

The output file contains four columns, these columns are defined as follows in Table 3.4.

Table 3.4 Columns in the bedGraph file for each group of DNA microarray

Column number	**Column name**	**Description**	**Example**
1	*Scaffold ID*	*The Scaffold ID indicates where the target gene is located*	*NW_003613582.1*
2	*Start*	*The leftmost coordinate of this gene*	*1513795*
3	*End*	*The rightmost coordinate of this gene*	*1521630*
4	*Expression value*	*The gene expression value*	*6.40219641124*

In GSE30321, an example of the bedGraph/bigWig file for GSE30321 is shown below:

```
NW_003613582.1   1513795    1521630    6.40219641124
NW_003613582.1   5516369    5601341    4.85232332011
NW_003613583.1   2978150    3327154    4.58852018019
…
```

When the data is displayed with JBrowse, each visible block represents an expressed gene, and the height of the column indicates the gene expression level. A

bigWig track is shown in Figure 3.1.2. These are the reference gene track (boxed in red) and the bigWig track (boxed in green). All visible blue blocks in bigWig track are genes with measurable expression values. To use the bigWig track, the first step is to use the reference gene track to find the location of the gene of interest. If there is a visible blue block in the bigWig track, it means the gene of interest was detected as expressed in the condition. By hovering the cursor over the blocks in the bigWig track, the average expression level value of the gene is shown. Figure 3.2 gives an example with the gene of interest *Adam10*. The average expression level for this gene is 4.85232.

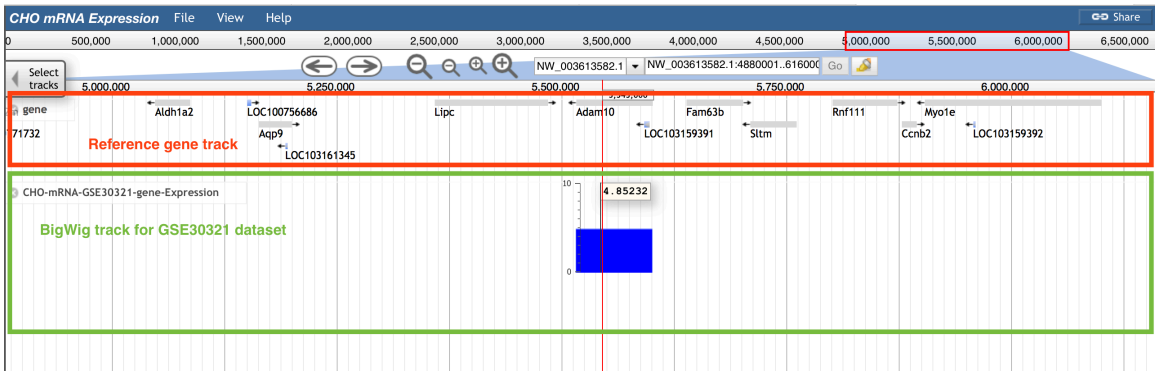

Figure 3.2 BigWig track for DNA microarray dataset GSE30321.

3.1.2.2.2 DNA microarray GTF file and GTF track

The bigWig files are used to visualize gene location and gene expression level, while the GTF file for each DNA microarray dataset is used to visualize the gene symbols for expressed genes.

The workflow for generating GTF file is shown in Figure 3.3:

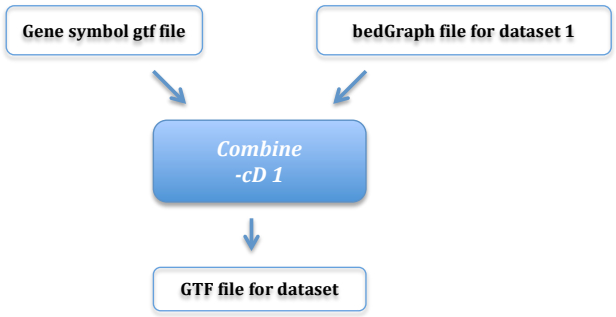

Figure 3.3 The workflow for generating GTF file for each DNA microarray dataset.

By using the argument '`Combine`' followed by an optional argument '`-cD`' (specifically for DNA microarray data), for which the default value is '0', users can use the command line to generate the GTF format file:

```
python ExpressGENiE.py Combine -cD 1 -i <input.bedgraph>
-gn <genesymbol.gtf> -o <output.gtf> -di <dataset id> -s
<source id> -cg <category>
```

Input files:

`-i/<input.bedgraph>`. The input file is the bedGraph file for DNA microarray dataset.

`-gn/<genesymbol.gtf>`. This option defines the assistant gene symbol GTF file (to generate the gene symbol GTF file, please check chapter 3.1.2.1 'Assistant files').

Output file (DNA microarray GTF):

`-o/<output.gtf>`. Writes the gene position information into the GTF format file for each DNA microarray dataset.

The GTF file has nine columns per line. These columns are defined as follows in Table 3.5.

Table 3.5 Columns in the GTF file for each DNA microarray dataset

Column number	**Column name**	**Description**	**Example**
1	*Scaffold ID*	*The scaffold ID indicates where the target gene is located*	*NW_003613582.1*
2	*Source*	*The source of this GTF file will be the PubMed ID and PMC ID (if available) of the related publication*	*PMID:21801763*
3	*Feature*	*The DNA microarray GTF file only keeps the feature that is named by parameter 'category'. E.g. gene*	*gene*
4	*Start*	*The leftmost coordinate of this gene*	*1513795*
5	*End*	*The rightmost coordinate of this gene*	*1521630*
6	*Score*	*In this GTF file, all scores are '.'*	*.*
7	*Strand*	*The strand of the gene. Always one of '+', '-', '.'*	*-*
8	*Phase*	*In this GTF file, all phases are '.'*	*.*
9	*Attributes*	*The attributes contain the gene symbol and the dataset ID. The dataset ID indicates where the data comes from, usually the GEO database*	*Name=Arpp19;Derived_from=GSE30321*

An example of the GTF file for GSE30321 is shown below:

```
NW_003613582.1  PMID:21801763   gene  1513795   1521630   .
     -    .    Name=Arpp19;Derived_from=GSE30321
NW_003613582.1  PMID:21801763   gene  5516369   5601341   .
     -    .    Name=Adam10;Derived_from=GSE30321
NW_003613583.1  PMID:21801763   gene  2978150   3327154   .
     -    .    Name=Gse1;Derived_from=GSE30321
…
```

Each shown green block in the GTF track represents a gene and its location (as shown in Figure 3.4). Figure 3.4 displays the reference gene track (boxed in red) and

the GTF track for dataset GSE30321 (boxed in green). It indicates that gene *Adam10* was identified in dataset GSE30321.

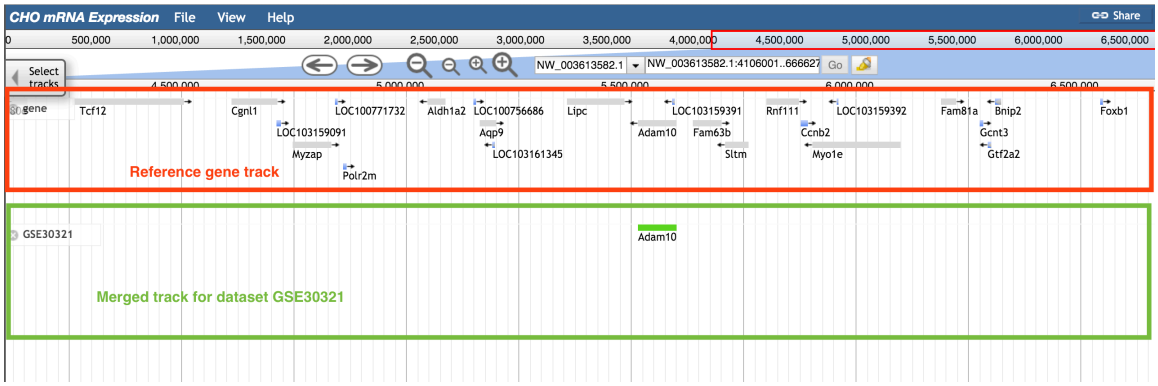

Figure 3.4 The GTF track for dataset GSE30321.

In Figure 3.5, when the user clicks the block, a pop-up window will give detailed information about the gene, such as dataset ID and PubMed ID. These IDs can be used to obtain raw data and the related paper for background knowledge.

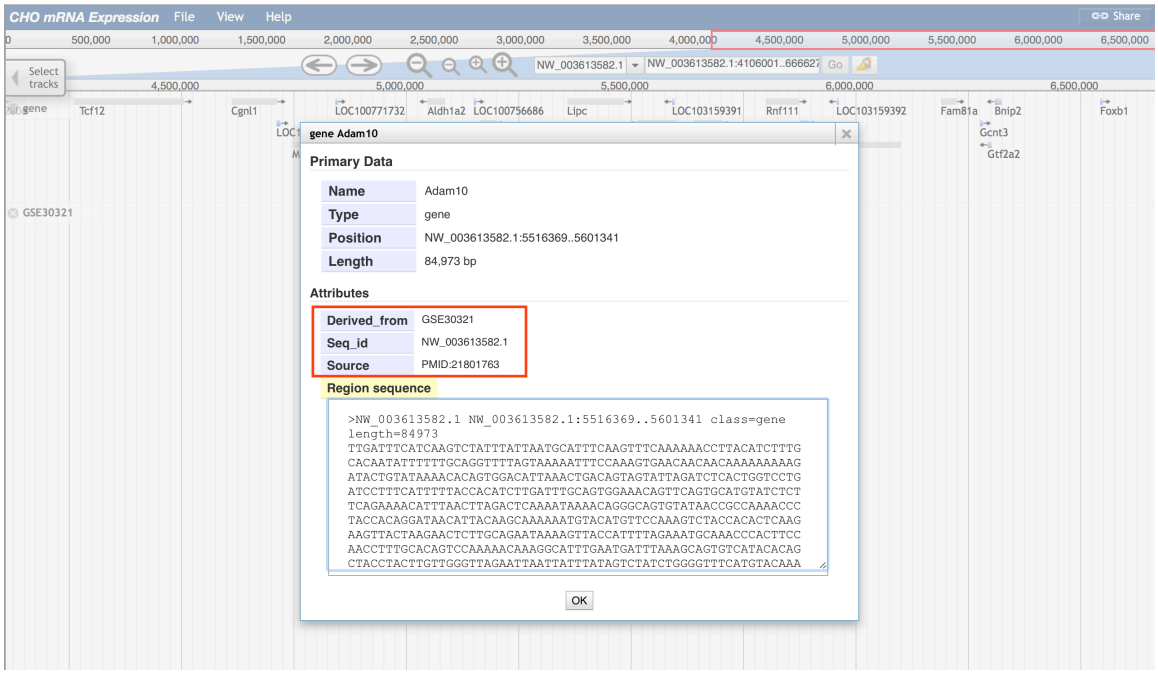

Figure 3.5 The pop-up window of the GTF track.

3.1.2.3 RNA-Seq dataset

There are five different types of file for each RNA-Seq dataset. All examples used in the following files and figures are derived from RNA-Seq dataset GSE75094.

3.1.2.3.1 The BAM file and BAM tracks for each sample

The BAM/SAM files contain read mapping information.

An example of a BAM/SAM file for one sample, SRR2922597, in GSE75094 is shown below:

```
SRR2922597.6699695      272     NW_003613580.1  1036    3
151M    *       0       0
AAGGGAATCAGATATATGTTGTATTTTTCTGTATCTATACAATAGTATTACTACTTAGGTCAT
TAGATATCTGATTTGTTTCATTGCTGAGTAATAGCCATGAAGCAGAGTGCTCCTTTGTTTCTT
TCCTATAGTTGCATGTAATGTGTCT
;?DFBDD??4DFFCCC;FFFDFAHHFFC.FHGDC@F@.FHGF=DBGDGCA.A8?HEAFFF=FH
HIGFDC-
```

```
HFGDHFFFGCFA=HGHFFEBFGDGFBHGEGGGDHHHHHFE;;FEFBFF=DFBHFFE>CAFFFF
?FBBBBBBBBBB?<??, AS:i:0  XN:i:0  XM:i:0  XO:i:0  XG:i:0
NM:i:0  MD:Z:151        YT:Z:UU NH:i:2  CC:Z:NW_003613992.1
CP:i:46434      HI:i:0
SRR2922597.675445       16      NW_003613580.1  33272   50
149M    *       0       0
AGGTATTGTAATTACTAGTATGTTTATATTAAATCTGTTGTGTTTCTCAGCATGGAAAAAATC
CAGTAAATTGCAGATTTCCACATATCCCAAAATTCCCAGGCCCCTAGTTCCCCCTATTAACAC
CCACCGTGAGTGTGGTACATATG
GGGGEGGHGGGGGEGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGHHHHHHHHHHHHFH
HHHHHHHHHIHIIIIHHHHGHHIIHHHHIIIIIIIIIIHHHIIIIIIHHHHIIIIIIHFFECH
HFAGGGGDEDEDDDDBBBAAA?A   AS:i:0  XN:i:0  XM:i:0  XO:i:0
XG:i:0  NM:i:0  MD:Z:149        YT:Z:UU NH:i:1
SRR2922597.2511706      272     NW_003613580.1  322974  0
151M    *       0       0
GCGATGAGTCCACGGAAAGGGAAGGAAGAGTGTCCTGAACCAAGCCAGATTCAGGAAGCTCCA
CCCTGCCTGGGCGTGTCTGTTTCAAGCAGGGACGAGCCTGGAGGGATCTGGGTGGATGGCGCT
TAGGACAAGAATTGGGTAAAAGCAG GGGGEECGGD?C9GGEECA-
GGGGGGEEGGGEGGGGGGGEGBEDGGGGGGGGGGGGHHHHFHFCHHHHHFHHHHCDFFFBCHGI
IIIIIHIHHHHDHHHHFGIHIHIIIIIHIHIIIIIHHHGHHHIHIIGGGGGGEDDDDDDDBBB
AA??? AS:i:0  XN:i:0  XM:i:0  XO:i:0  XG:i:0  NM:i:0  MD:Z:151
YT:Z:UU NH:i:20 CC:Z:NW_003613752.1     CP:i:2129241    HI:i:0
…
```

The BAM file is shown as the alignment feature and coverage feature tracks on JBrowse (these two bam file tracks are shown in Figure 3.6). The alignment BAM track indicates reads alignment information and the coverage BAM track provides a quick coverage plot. In Figure 3.6, it displays the reference gene track (boxed in red), the coverage BAM track (boxed in green) and the alignment BAM track (boxed in blue). The alignment BAM track shows the position, direction and match/mismatch information of the reads. A red dot represents a read oriented in a positive direction, a blue dot represents a read in the negative direction, and a black bar within a read represents a mismatch region. The coverage BAM track shows the read density in each position. By hovering the cursor over a position, the user can obtain the value of the density.

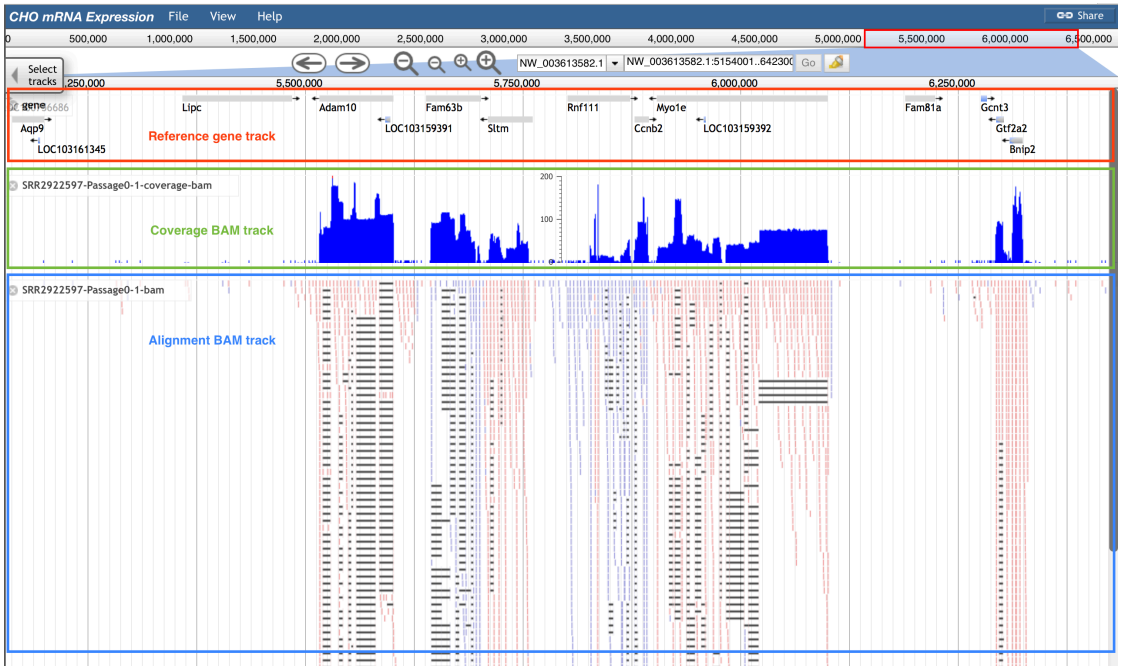

Figure 3.6 The BAM tracks for RNA-Seq sample SRR2922597.

3.1.2.3.2 The annotation GFF3 file and GFF3 track for each sample

The annotation GFF3 file for each sample contains annotated expressed exons and transcripts. The workflow for generating the annotation GFF3 file is shown in Figure 3.7:

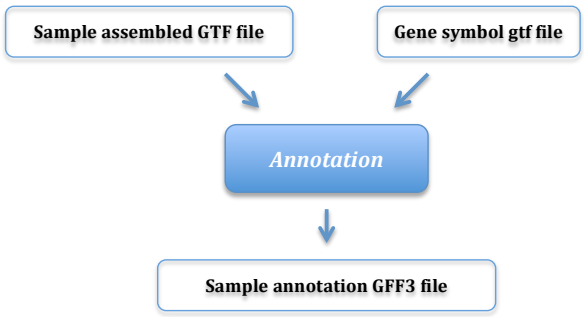

Figure 3.7 The workflow for generating the annotation GFF3 file for each RNA-Seq sample.

To generate this GFF3 file, use the argument '`Annotation`'. Follow the command:

```
python ExpressGENiE.py Annotation -i <assembled.gtf> -gn <genesymbol.gtf> -o <output.gff3> -si <sample id> -s <source id>
```

Input files:

`-i/<input.gtf>`. The input file is an assembled GTF file for each sample. Users can obtain the assembled GTF file from Cufflinks (part of the Tuxedo pipeline).

`-gn/<genesymbol.gtf>`. This option defines the assistant gene symbol GTF file (to generate the gene symbol GTF file, please check chapter 3.1.2.1 'Assistant files').

Output file (RNA-Seq annotation GFF3):

`-o/<output.gff3>`. Writes the annotated transcripts information into GFF3 format file for each sample.

The annotation GFF3 file has nine columns per line. These columns are defined as follows in Table 3.6.

Table 3.6 Columns in the annotation GFF3 file for each RNA-Seq sample

Column number	Column name	Description	Example
1	*Scaffold ID*	*The Scaffold ID indicates where the sequence is located*	*NW_003613580.1*
2	*Source*	*The source in this GTF file will be the PubMed ID and PMC ID (if available) of the related publication*	*PMID:26854539*
3	*Feature*	*The feature is inherited from the GTF file*	*transcript*
4	*Start*	*The leftmost coordinate of this gene*	*745285*
5	*End*	*The rightmost coordinate of this gene*	*746407*
6	*Score*	*The score is inherited from the GTF file*	*1000*
7	*Strand*	*The strand is inherited from the GTF file*	*.*
8	*Phase*	*The phase is inherited from the GTF file*	*.*
9	*Attributes*	*The attributes contain the ID for gene ID/symbol and transcript ID (if available). The transcript ID is inherited from the assembled GTF file. If the gene can be found in the gene symbol GTF file, the gene symbol will be inherited from the gene symbol GTF file. 'Derived_from' indicates the sample ID/symbol*	*ID=gene_id:CUFF.3*transcript_id:CUFF.3.1;Derived_from=SRR2922597*

An example of the annotation GFF3 file for one of sample, SRR2922597, in GSE75094 is shown below:

```
NW_003613580.1  PMID:26854539   transcript      745285  746407
1000    .       .
ID=gene_id:CUFF.3*transcript_id:CUFF.3.1;Derived_from=SRR292259
7
NW_003613580.1  PMID:26854539   exon    745285  746407  1000
```

```
.       .
ID=gene_id:CUFF.3*transcript_id:CUFF.3.1*1;Parent=gene_id:CUFF.
3*transcript_id:CUFF.3.1;Derived_from=SRR2922597
NW_003613580.1  PMID:26854539   transcript      739188  739857
1000    .       .
ID=gene_id:CUFF.1*transcript_id:CUFF.1.1;Derived_from=SRR292259
7
NW_003613580.1  PMID:26854539   exon    739188  739857  1000
.       .
ID=gene_id:CUFF.1*transcript_id:CUFF.1.1*1;Parent=gene_id:CUFF.
1*transcript_id:CUFF.1.1;Derived_from=SRR2922597
…
```

In the annotation GFF3 track, the region with blue and grey represents the transcript, while the red region represents the exon. In Figure 3.8, the reference gene track (boxed in red), the annotation GFF3 track (boxed in green), and the assembled GTF track (boxed in blue) are displayed. The GTF file contains expressed exons and transcripts as well as unexpressed exons and transcripts. While in the annotation GFF3 track, unexpressed exons and transcripts have been removed and only annotated expressed exons and transcripts are kept.

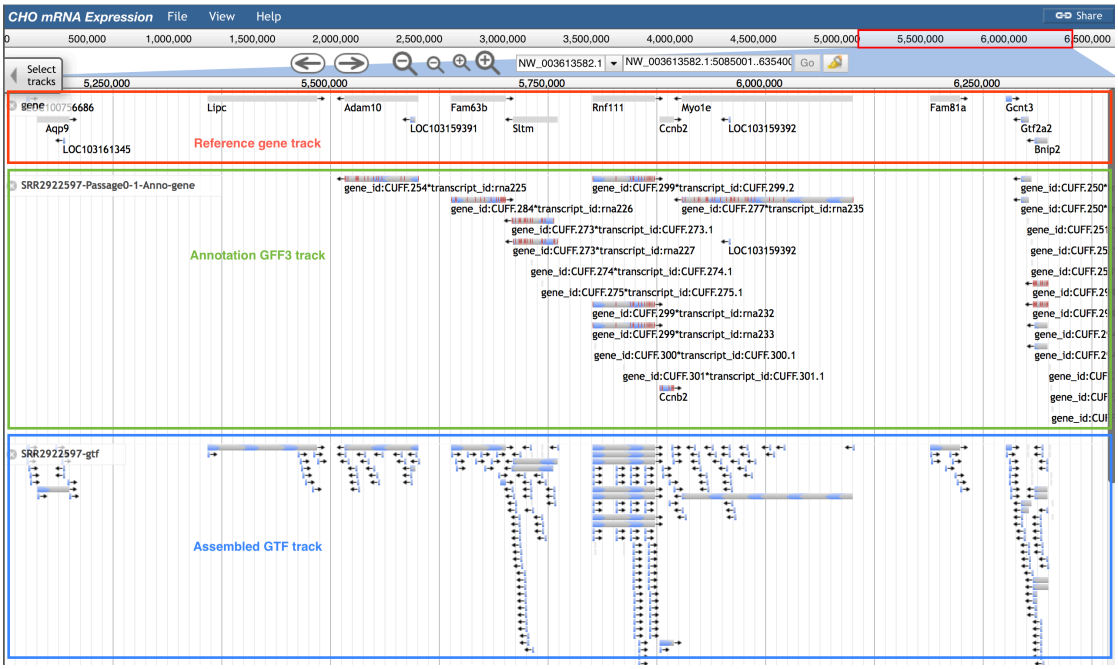

Figure 3.8 The GFF3 annotation track for RNA-Seq sample SRR2922597.

3.1.2.3.3 The bigWig file and bigWig track for each sample

In a RNA-Seq dataset, each sample has a bigWig track that is used to display the gene position and its expression level. The workflow for generating the FPKM bigWig file is shown in Figure 3.9:

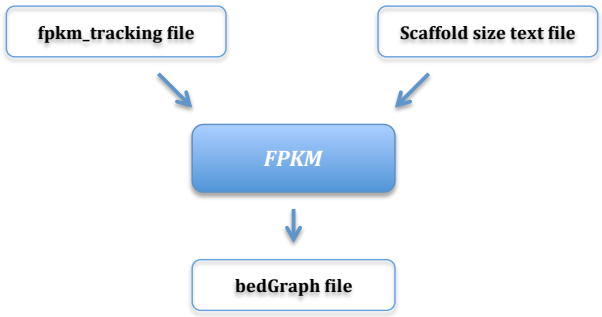

Figure 3.9 The workflow for generating the FPKM bigWig file for each RNA-Seq sample.

Use the argument '`FPKM`' to generate the bigWig/bedGraph file and follow the command:

```
python ExpressGENiE.py FPKM -i <input.fpkm_tracking> -sf <scaffold size.txt> -o <output.bedgraph>
```

Input files:

`-i/<input.fpkm_tracking>`. The input file is a text format file with suffix 'fpkm_tracking', which is the gene-level FPKM output file from Cufflinks. It contains all gene expression values and other detailed information.

`-sf/<scaffold size.txt>`. The option indicates the assistant scaffold size text file (to generate the scaffold size text file, please check chapter 3.1.2.1 'Assistant files').

Output file (RNA-Seq FPKM bigWig/bedGraph):

`-o/<output.bedgraph>`. Writes the genes' positions and expression values into the bedGraph format file.

The file contains four columns, which record the expression value and position for each gene. These columns are defined as follows in Table 3.7.

Table 3.7 Columns in the FPKM bigWig file for each RNA-Seq sample

Column number	**Column name**	**Description**	**Example**
1	*Scaffold ID*	*The Scaffold ID indicates where the target gene is located*	*NW_003613580.1*
2	*Start*	*The start of the gene*	*736556*
3	*End*	*The end of the gene*	*737350*
4	*Expression value*	*The gene expression value*	*1.7958*

Because all genes or transcripts are predicted by Cufflinks, some of the predicted genes overlap each other in the FPKM tracking file. Additionally, there are

some genes totally covered by another gene that has a longer sequence. To prevent the same region have two different values or one value covered by another, the gene expression level of the gene that has the shorter sequence has been inserted into the expression level of longer sequence gene. The example of FPKM bigWig/bedGraph file for one of sample, SRR2922597, in GSE75094 is shown below:

```
NW_003613580.1  736556  737350  1.7958
NW_003613580.1  739187  739857  1.51459
NW_003613580.1  745284  746407  0.897637
…
```

As the bigWig track does not display the gene symbols, it is recommended to use the annotation GFF3 track with the bigWig track. Figure 3.10 displays the reference gene track (boxed in red), the annotation GFF3 track (boxed in green), and the FPKM bigWig track (boxed in blue). The FPKM bigWig file indicates the gene expression level, which is same as the usage of the bigWig file in DNA microarray dataset. When users zoom in, the expression level for each region becomes more clear and distinguishable. By using the annotation GFF3 track and reference gene track, users can match the expressed gene with its expression level. For example, the gene *Adam10* expression level is 39.2699.

Figure 3.10 The FPKM bigWig track for one sample of RNA-Seq.

3.1.2.3.4 The differential GTF file and GTF track for each dataset

The differential GTF file contains differentially expressed genes and the expression values between any paired groups within a dataset. The workflow for generating the differential GTF file is shown in Figure 3.11:

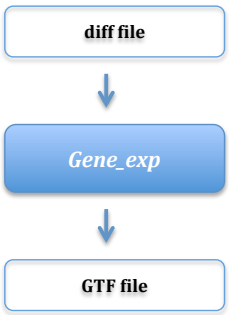

Figure 3.11 The workflow for generation of the differential GTF file for each RNA-Seq dataset.

The argument is '`Gene_exp`', which provides two options to generate the differential GTF file.

Option 1. To select by p-value, follow the command to generate the differential GTF file:

```
python ExpressGENiE.py Gene_exp -i <input.diff> -o <output.gtf>
```

Option 2. To select by log2(fold_change) threshold, follow the command to generate the differential GTF file:

```
python ExpressGENiE.py Gene_exp -i <input.diff> -o <output.gtf> -th <threshold>
```

Input file:

`-i/<input.diff>`. The input file is a text format file with suffix 'diff', which users can obtain by using Cuffmerge and Cuffdiff.

Output file (RNA-Seq differential GTF):

`-o/<output.gtf>`. Writes the position and expression value of the differently expressed genes into the GTF format file.

The differential GTF file contains nine columns per line. These columns are defined as follows in Table 3.8.

Table 3.8 Columns in the differential GTF file for one RNA-Seq dataset

Column number	**Column name**	**Description**	**Example**
1	*Scaffold ID*	*The Scaffold ID indicates where the sequence is located*	*NW_003613580.1*
2	*Source*	*The source in this GTF file will be 'cuffdiff', which is the tool that produces the gene differential information*	*cuffdiff*
3	*Feature*	*The feature used in this GTF file indicates the differential gene expression*	*gene_expression_diff*
4	*Start*	*The leftmost coordinate of this gene*	*1890885*
5	*End*	*The rightmost coordinate of this gene*	*2042119*
6	*Score*	*The score is '.'*	.
7	*Strand*	*The strand is '.'*	.
8	*Phase*	*The phase is '.'*	.
9	*Attributes*	*The attributes contain the gene ID/symbol. The 'Derived_from' value indicates which groups are expressed differently for this gene and the gene expression value for each group*	*Name=Taf3;Derived_from=Passage0:3.9628*Passage4:1.50716*

The example of the differential GTF file for GSE75094:

```
NW_003613580.1  cuffdiff  gene_expression_diff  1890885
     2042119    .    .    .
     Name=Taf3;Derived_from=Passage0:3.9628*Passage4:1.50716

NW_003613580.1  cuffdiff  gene_expression_diff  1890885
     2042119    .    .    .
     Name=Taf3;Derived_from=Passage2:2.01824*Passage6:5.19716

NW_003613580.1  cuffdiff  gene_expression_diff  1890885
     2042119    .    .    .
     Name=Taf3;Derived_from=Passage4:1.50716*Passage6:5.19716

…
```

In the differential GTF track, each visible block represents the gene that was differentially expressed in paired groups. Figure 3.12 displays the reference gene track (boxed in red) and the differential GTF track (boxed in green). The differential GTF file track is highlighted in green. If a green block exists in the track, it means in this position the gene is expressed differently between two conditions. The raw differential file with 'diff' as the suffix lists several parameters, like $\log_2$(fold_change) and the significance of the difference which is based on a calculated p-value and an alpha-value of 0.05. The selection condition is flexible and can be used to generate different differential GTF files.

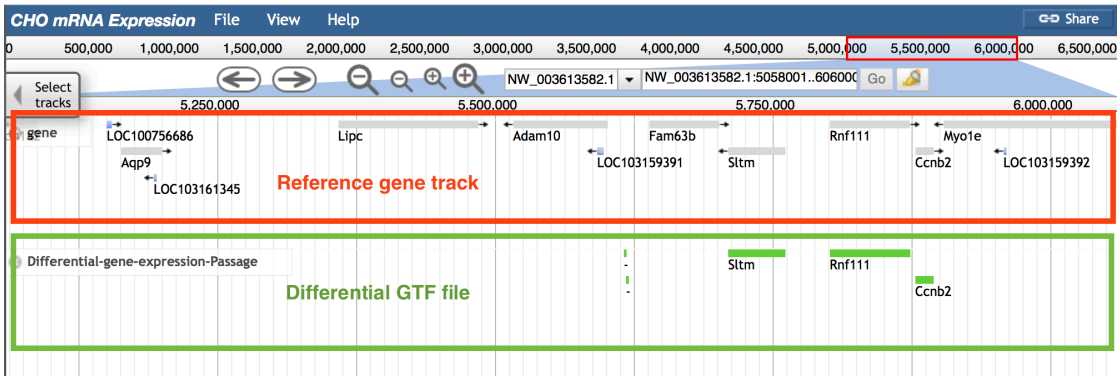

Figure 3.12 The differential GTF track for one sample of RNA-Seq.

In Figure 3.13, a pop-up window shows information of the gene *Sltm*, such as all paired groups of the gene expressed differently and the corresponding gene expression value.

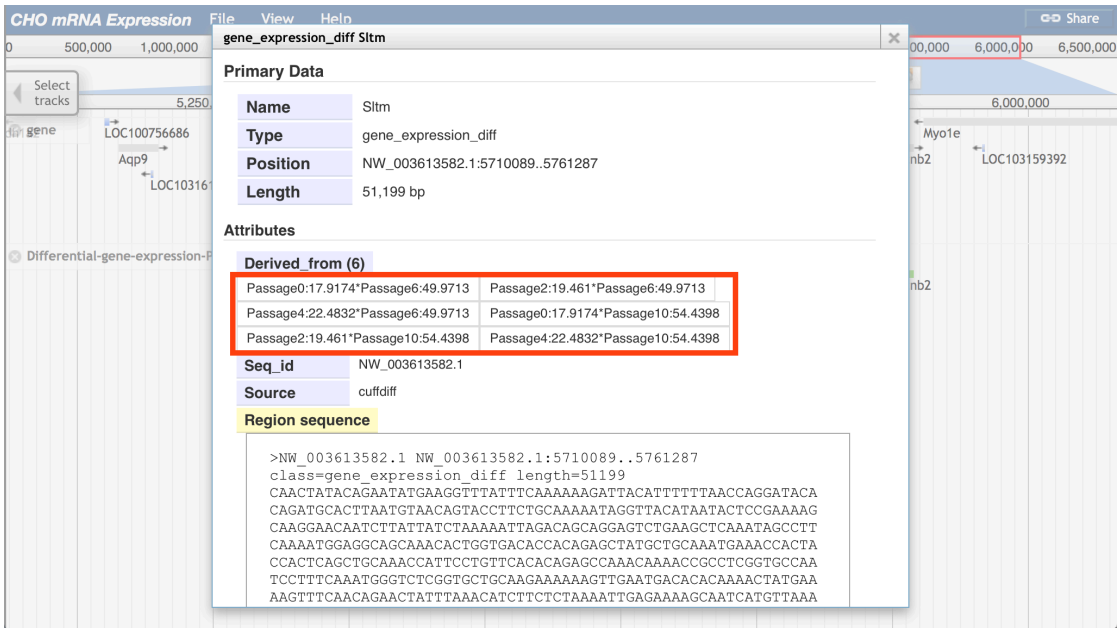

Figure 3.13 The pop-up window of the differential GTF track.

3.1.2.3.5 The RNA-Seq dataset GFF3 file and GFF3 track

The function of this GFF3 format file is to indicate the expressed genes/sequences and their sources (which sample expressed the genes/sequences) in each dataset. The argument is '`Combine`'. There are three steps to generate this file as shown in Figure 3.14:

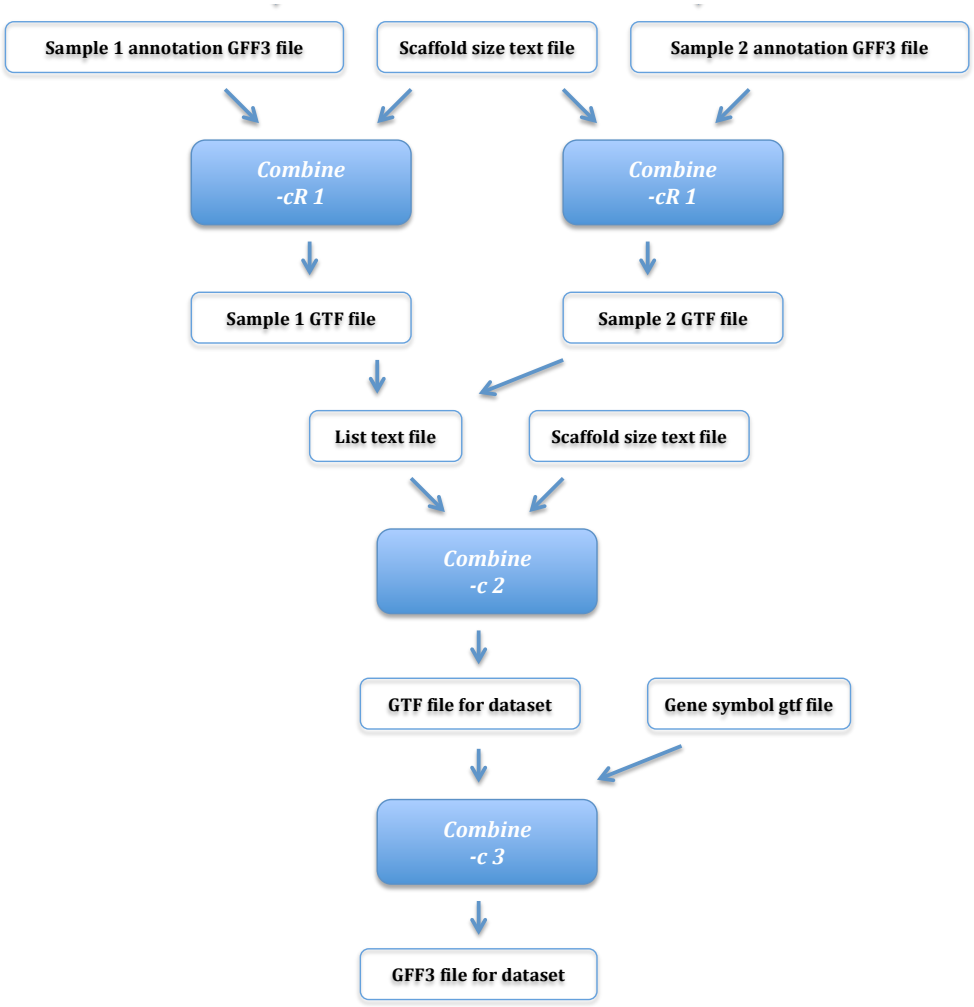

Figure 3.14 The workflow for generating the RNA-Seq dataset GFF3 file.

Step 1. Generating a GTF file for each sample

The parameter 'category' in this step should be the item in the 'feature' column that the user wants to select. Because the input file in this step is the GFF3 file for each sample (generated in section 3.1.2.3.2) and the 'feature' column in this file is 'transcript', the selected item should be 'transcript'. By using optional argument '`-cR`'

(specifically for RNA-Seq data), which has default value '0', follow the command to generate the GTF format file:

```
python ExpressGENiE.py Combine -cR 1 -i <input.gff3> -sf <scaffold size.txt> -o <output.gtf> -si <sample id> -s <source id> -cg <category>
```

Input files:

`-i/<input.gff>`. The input file is the annotation GFF3 for each sample.

`-sf/<scaffold size.txt>`. The option indicates the assistant scaffold size text file (to generate the scaffold size text file, please check chapter 3.1.2.1 'Assistant files').

Output file:

`-o/<output.gtf>`. The output GTF file only keeps records for the selected category.

Step 2. Generating a GTF file for the entire dataset

This GTF file contains gene information from all the samples. The parameter 'category' is same as the item that was selected in step 1. The parameter 'parent category' is a newly named category that contains the item 'category'. Follow the command:

```
python ExpressGENiE.py Combine -c 2 --list <file list.txt> -sf <scaffold size.txt> -o <output.gtf> -cg <category> -pa <parent>
```

Input files:

`--list/<file list.txt>`. The input text format file records the path of GTF files that were generated in step 1.

`-sf/<scaffold size.txt>`. This option indicates the assistant scaffold size text file (to generate the scaffold size text file, please check chapter 3.1.2.1 'Assistant files').

Output file:

`-o/<output.gtf>`. This GTF file keeps information records from all samples.

Step 3. Annotating the GTF file

This function annotates the GTF file to a GFF3 file. The parameter 'category' and 'parent category' are same as the items that were selected in step 2. The parameter 'new category' is a newly named category for distinguishing the source of data. Follow the command:

```
python ExpressGENiE.py Combine -c 3 -i <input.gtf> -gn
<genesymbol.gtf> -o <output.gff3> -a <amount of samples> -di
<dataset id> -cg <category> -Ncg <new category> -pa <parent>
```

Input files:

`-i/<input.gtf>`. This input file is the GTF file produced in the step 2.

`-gn/<genesymbol.gtf>`. This option indicates the assistant gene symbol GTF file (to generate the gene symbol GTF file, please check chapter 3.1.2.1 'Assistant files').

Output file (RNA-Seq dataset GFF3):

`-o/<output.gff3>`. Writes the information records from all samples into the GFF3 format file.

The dataset GFF3 file has nine columns per line. These columns are defined as follows in Table 3.9.

Table 3.9 Columns in the dataset GFF3 file of RNA-Seq

Column number	**Column name**	**Description**	**Example**
1	*Scaffold ID*	*The Scaffold ID indicates where the sequence is located*	*NW_003613580.1*
2	*Source*	*The source in this GFF3 file will be the PubMed ID and PMC ID (if available) of the related publication*	*PMID:26854539*
3	*Feature*	*The feature in this GFF3 file is based on the names that users assign to the parameters 'category', 'new category', and 'parent category'. The feature will be one of the three user-named options. E.g. transcript, all_transcripts, sequence*	*transcript*
4	*Start*	*The leftmost coordinate of this gene*	*736568*
5	*End*	*The rightmost coordinate of this gene*	*736649*
6	*Score*	*The score is '1'*	*1*
7	*Strand*	*The strand of the gene. Always one of '+', '-', '.'. It is inherited from the GTF file*	*+*
8	*Phase*	*The phase is '.'*	*.*
9	*Attributes*	*The attributes will give the 'ID', 'Parent', and 'derived_from' items. The 'ID' consists of the strand, scaffold ID, and dataset ID. 'Parent' is the ID of the parent region. 'Derived_from' indicates which sample contains this region*	*ID=positive*NW_003613580.1*0*GSE75094*1;Parent=positive*NW_003613580.1*0; Derived_from=SRR2922598*

The dataset GFF3 file stores source information for each sequence. The example of dataset GFF3 file for GSE75094 is shown below:

```
NW_003613580.1   PMID:26854539    sequence   736568     738820
     1    +     .     ID=positive*NW_003613580.1*0
```

```
NW_003613580.1   PMID:26854539    transcript 736568     736649
      1     +     .
      ID=positive*NW_003613580.1*0*GSE75094*1;Parent=positive*N
W_003613580.1*0;Derived_from=SRR2922598
NW_003613580.1   PMID:26854539    transcript 736650     736917
      1     +     .
      ID=positive*NW_003613580.1*0*GSE75094*2;Parent=positive*N
W_003613580.1*0;Derived_from=SRR2922598&SRR2922601
NW_003613580.1   PMID:26854539    transcript 736918     736929
      1     +     .
      ID=positive*NW_003613580.1*0*GSE75094*3;Parent=positive*N
W_003613580.1*0;Derived_from=SRR2922598&SRR2922601&SRR2922604

NW_003613580.1   PMID:26854539    transcript 736930     738580
      1     +     .
      ID=positive*NW_003613580.1*0*GSE75094*4;Parent=positive*N
W_003613580.1*0;Derived_from=SRR2922598&SRR2922601&SRR2922603&S
RR2922604
NW_003613580.1   PMID:26854539    transcript 738581     738667
      1     +     .
      ID=positive*NW_003613580.1*0*GSE75094*5;Parent=positive*N
W_003613580.1*0;Derived_from=SRR2922598&SRR2922601&SRR2922604

NW_003613580.1   PMID:26854539    transcript 738668     738797
      1     +     .
      ID=positive*NW_003613580.1*0*GSE75094*6;Parent=positive*N
W_003613580.1*0;Derived_from=SRR2922598&SRR2922601
NW_003613580.1   PMID:26854539    transcript 738798     738820
      1     +     .
      ID=positive*NW_003613580.1*0*GSE75094*7;Parent=positive*N
W_003613580.1*0;Derived_from=SRR2922598
…
```

The dataset GFF3 track uses two different colors (Figure 3.15). Figure 3.15 displays the reference gene track (boxed in red) and the dataset GFF3 track (boxed in green). The green region represents all samples that are expressed this region. The yellow region represents the gene that was not expressed in at least one sample. In Figure 3.16, the pop-up window shows the source of each region and other metadata.

Figure 3.15 The RNA-Seq dataset GFF3 track for GSE75094.

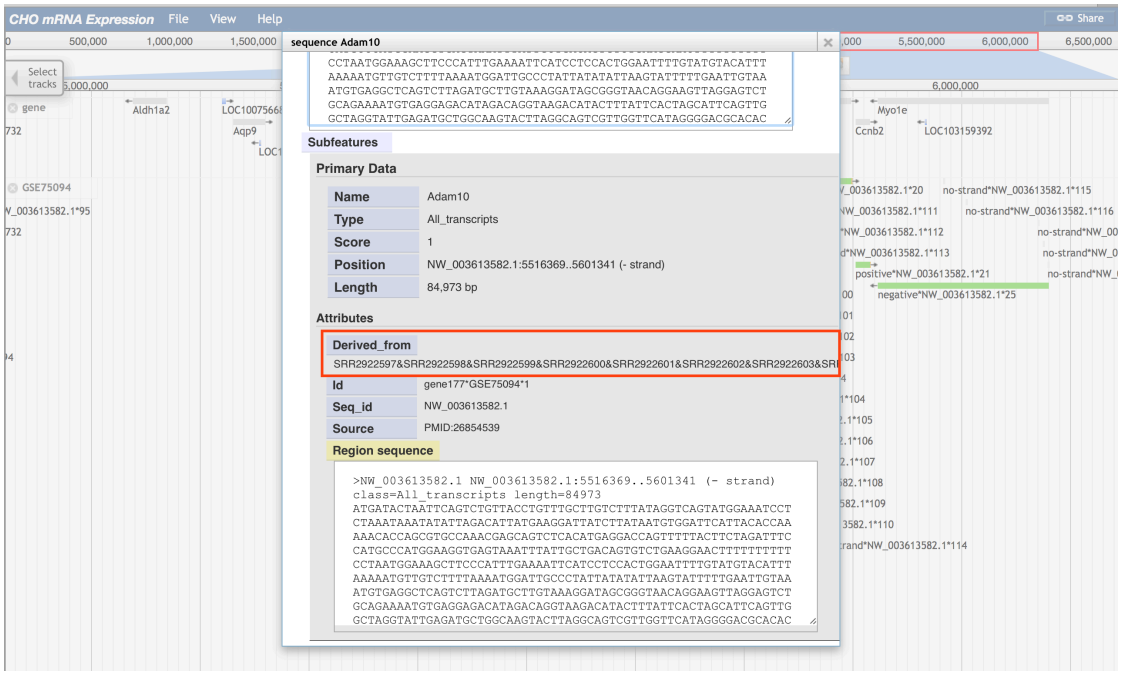

Figure 3.16 The pop-up window of dataset RNA-Seq.

3.1.2.4 The data type GFF3 file and overall GFF3 file

The profile and usage of data type and overall GFF3 files are similar to the RNA-Seq dataset GFF3 file. The only difference is the source of data. The data source in RNA-Seq dataset GFF3 file is the sample ID, in the data type GFF3 file, it is the dataset ID, and in overall GFF3 file, it is the data type.

3.1.2.4.1 The data type GFF3 file and GFF3 track

The GFF3 file for each data type provides a general view for all expressed genes/sequences that are derived from the same data type. To obtain the data type GFF3 file, there are three steps that are similar to generating a GFF3 file for each RNA-Seq dataset (each DNA microarray dataset already has a GTF file, so skip step 1) as shown in Figure 3.17:

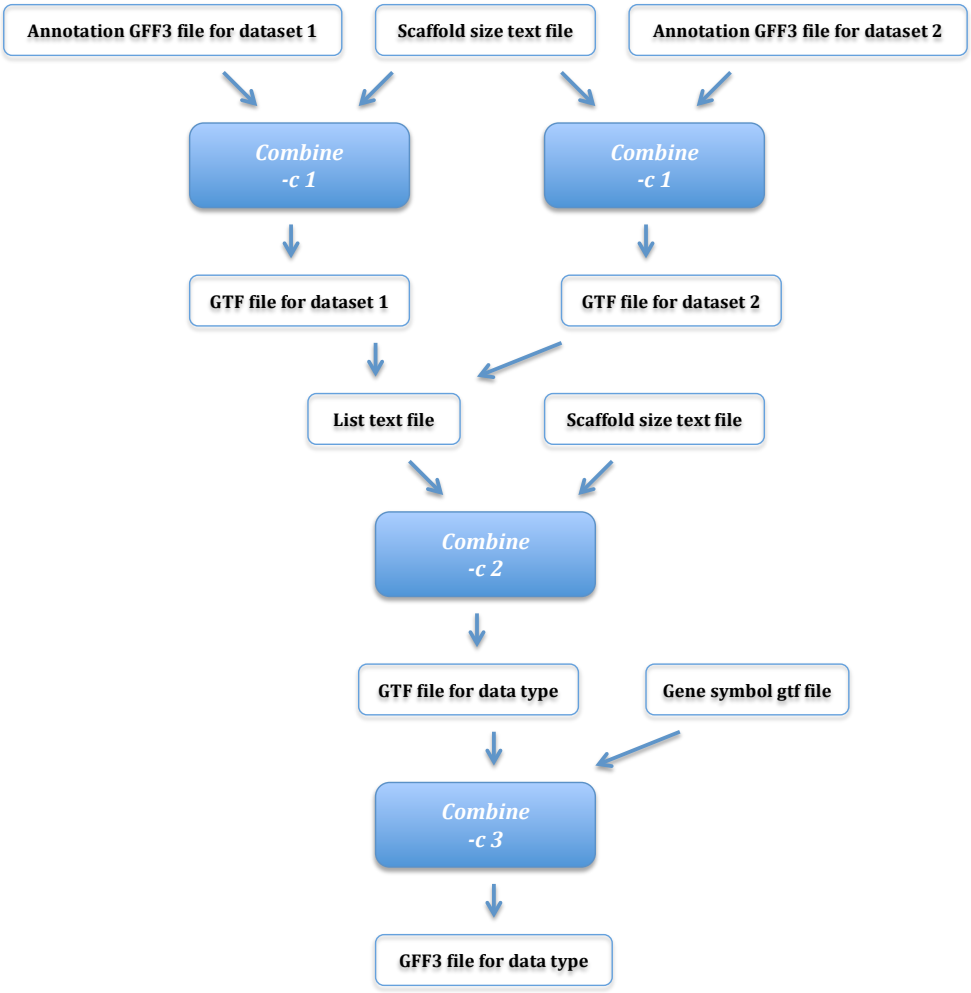

Figure 3.17 The workflow for generating the data type GFF3 file.

Step 1. Generating a GTF file for each dataset

The parameter ‘category’ in this step should be the item in the ‘feature’ column that the user wants to select. Based on the input file with the dataset GFF3 file in this step, the selected item should be the ‘parent category’ used in dataset GFF3 file. Follow the command:

```
python ExpressGENiE.py Combine -c 1 -i <input.gff3> -sf <scaffold size.txt> -o <output.gtf> -di <dataset id> -s <source id> -cg <category>
```

Input files:

`-i/<input.gff>`. The dataset GFF3 file is the input file.

`-sf/<scaffold size.txt>`. This option indicates the assistant scaffold size text file (to generate the scaffold size text file, please check chapter 3.1.2.1 'Assistant files').

Output file:

`-o/<output.gtf>`. The GTF file only consists of the selected category records.

Step 2. Generating a GTF file for the entire data type

The parameter 'category' is same as the item that was selected in step 1. And the parameter 'parent category' is a newly named category that contains the item 'category'. Follow the command:

```
python ExpressGENiE.py Combine -c 2 --list <file list.txt> -sf <scaffold size.txt> -o <output.gtf> -cg <category> -pa <parent>
```

Input files:

`--list/<file list.txt>`. The input text format file records the path of GTF files that were generated in step 1.

`-sf/<scaffold size.txt>`. The option indicates the assistant scaffold size text file (to generate the scaffold size text file please, check chapter 3.1.2.1 'Assistant files').

Output file:

`-o/<output.gtf>`. The output file keeps informational records from all datasets within the same data type.

Step 3. Annotating the GTF file

This function annotates the GTF file to a GFF3 file. The parameter 'category' and 'parent category' are same as the items that were selected in step 2. The parameter 'new category' is a newly named category for distinguishing the source of data. Follow the command:

```
python ExpressGENiE.py Combine -c 3 -i <input.gtf> -gn <genesymbol.gtf> -o <output.gff3> -a <amount of datasets or datatype> -tp <data type> -cg <category> -Ncg <new category> -pa <parent>
```

Input files:

`-i/<input.gtf>`. The GTF file produced in step 2 is the input file.

`-gn/<genesymbol.gtf>`. This option indicates the assistant gene symbol GTF file (to generate the gene symbol GTF file, please check chapter 3.1.2.1 'Assistant files').

Output file (Data type GFF3):

`-o/<output.gff3>`. Writes the information records from all datasets within the same data type into the GFF3 format file.

The data type GFF3 file has nine columns per line. These columns are defined as follows in Table 3.10.

Table 3.10 Columns in the data type GFF3 file

Column number	Column name	Description	Example
1	*Scaffold ID*	*The Scaffold ID indicates where the sequence is located*	*NW_003613580.1*
2	*Source*	*The source is 'ExpressGENiE'*	*ExpressGENiE*
3	*Feature*	*The feature in this GFF3 file is based on the names that users assign to the parameters 'category', 'new category', and 'parent category'. The feature will be one of the three user-named options. E.g: sequence, all_sequence, region*	*sequence*
4	*Start*	*The leftmost coordinate of this gene*	*690691*
5	*End*	*The rightmost coordinate of this gene*	*691138*
6	*Score*	*The score is '1'*	*1*
7	*Strand*	*The strand of the gene. Always one of '+', '-', '.'. It is inherited from the GTF3 file*	+
8	*Phase*	*The phase is '.'*	.
9	*Attributes*	*The attributes will give the 'ID', 'Parent', and 'derived_from' items. The 'ID' consists of strand, scaffold ID, and dataset ID. 'Parent' is the ID of parent region. 'Derived_from' indicates which dataset contains this region*	*ID=positive*NW_003613580.1*0*All_Datasets*1;Parent=positive*NW_003613580.1*0;Derived_from=GSE59487*

An example of the data type GFF3 file for the RNA-Seq data type is shown below:

```
NW_003613580.1  ExpressGENiE    region    690691    691138
     1    +    .
     ID=gene1;Name=LOC100757647;Derived_from=RNA-Seq
NW_003613580.1  ExpressGENiE    sequence  690691    691138
     1    +    .    ID=gene1*RNA-
Seq*1;Name=LOC100757647;Parent=gene1;Derived_from=GSE59487
```

```
NW_003613580.1   ExpressGENiE     region      736568      738820
      1     +      .     ID=positive*NW_003613580.1*1
NW_003613580.1   ExpressGENiE     sequence    736568      738820
      1     +      .     ID=positive*NW_003613580.1*1*RNA-
Seq*1;Parent=positive*NW_003613580.1*1;Derived_from=GSE75094
…
```

The dataset GFF3 track uses two different colors (Figure 3.18). The green region represents all samples that are expressed this region. The yellow region represents at least one sample didn't express this region. In Figure 3.18, it displays the reference gene track (boxed in red), the RNA-Seq data type GFF3 track (boxed in green), the GSE75094 dataset GFF3 track (boxed in blue), and the GSE59487 dataset track (boxed in yellow). Using the gene *Adam10* as an example, the gene *Adam10* has been detected expressed in both datasets. Thus, in the RNA-Seq data type GFF3 track, the color of region of *Adam10* is green. In Figure 3.19, when the user clicks the region in the RNA-Seq data type GFF3 track, a pop-up window shows the data source.

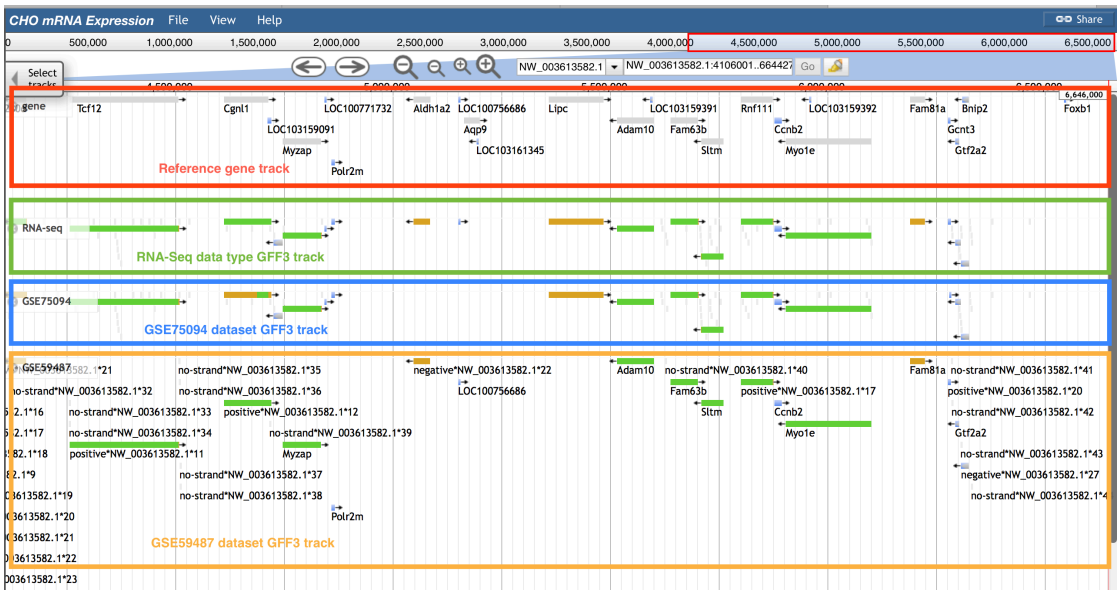

Figure 3.18 The data type GFF3 track.

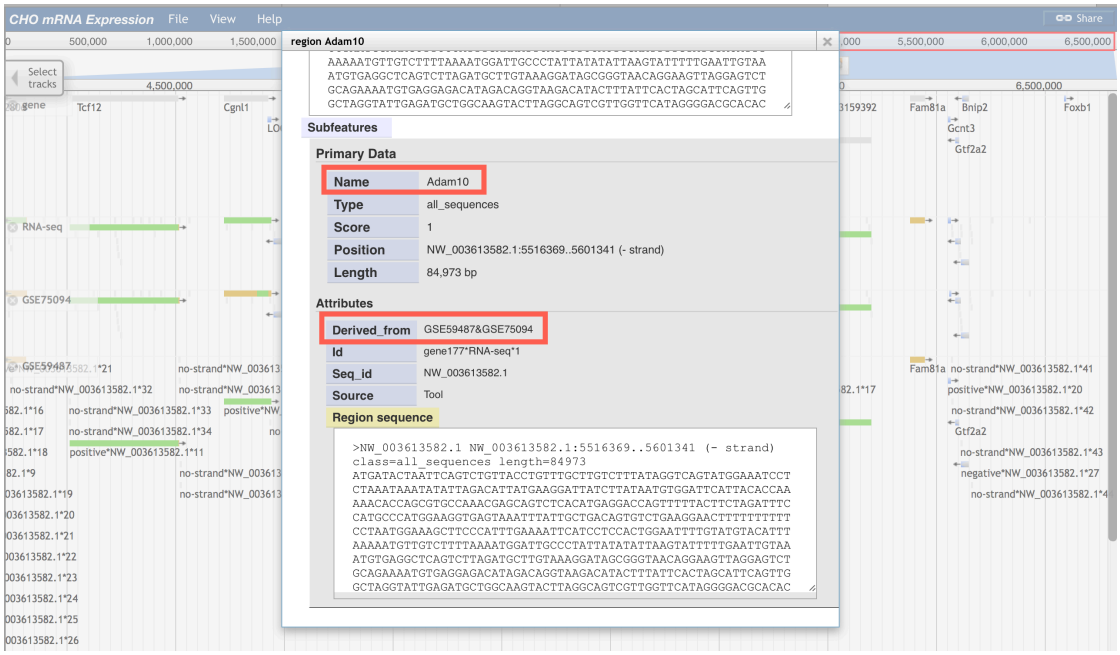

Figure 3.19 The pop-up window of data type GFF3 track.

3.1.2.4.2 The overall GFF3 file and GFF3 track

The overall GFF3 file for both data types provides an overall view for all datasets. To generate this GFF3 file, three steps are similar to generating a GTF file for each data type as shown in Figure 3.20:

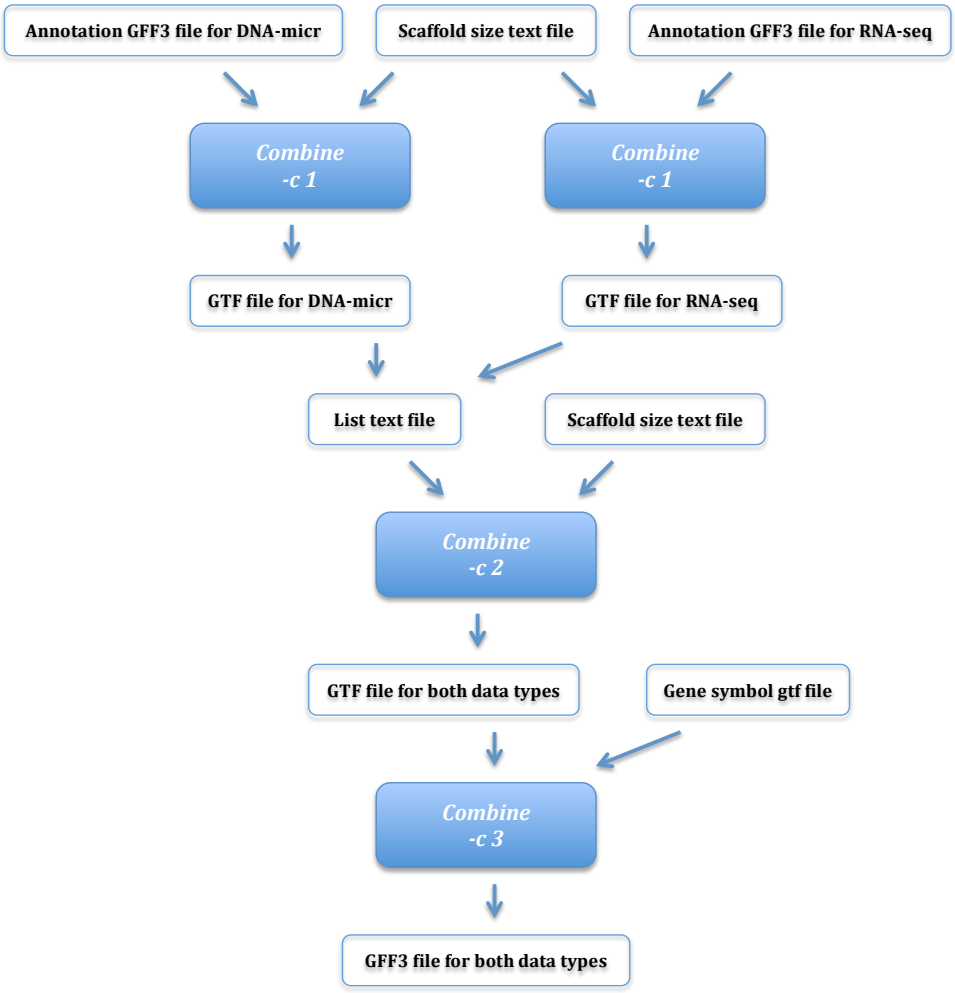

Figure 3.20 The workflow for generating the overall GFF3 file.

Step 1. Generating a GTF file for each data type

The parameter ‘category’ in this step should be the item in ‘feature’ column that the user wants to select. Based on the input file with data type GFF3 file in this step, the selected item should be the ‘parent category’ used in data type GFF3 file. Follow the command:

```
python ExpressGENiE.py Combine -c 1 -i <input.gff3> -sf <scaffold size.txt> -o <output.gtf> -tp <data type> -s <tool name> -cg <category>
```

Input files:

`-i/<input.gff>`. The data type GFF3 file is the input file.

`-sf/<scaffold size.txt>`. This option indicates the assistant scaffold size text file (to generate the scaffold size text file, please check chapter 3.1.2.1 'Assistant files').

Output file:

`-o/<output.gtf>`. The output GTF file only keeps records for the selected category.

Step 2. Generating a GTF file for the both data types

The parameter 'category' is same as the item that was selected in step 1. The parameter 'parent category' is a new named category that contains the item 'category'. Follow the command:

```
python ExpressGENiE.py Combine -c 2 --list <file list.txt> -sf <scaffold size.txt> -o <output.gtf> -cg <category> -pa <parent>
```

Input files:

`--list/<file list.txt>`. The input text format file records the paths of the GTF files that were generated in step 1.

`-sf/<scaffold size.txt>`. This option indicates the assistant scaffold size text file (to generate the scaffold size text file, please check chapter 3.1.2.1 'Assistant files').

Output file:

`-o/<output.gtf>`. The output GTF file contains records from both data types.

Step 3. Annotating the GTF file

This function annotates the GTF file to a GFF3 file. The parameter 'category' and 'parent category' are same as the items that were selected in step 2. The parameter 'new category' is a newly named category for distinguishing the source of the data. Follow the command:

```
python ExpressGENiE.py Combine -c 3 -i <input.gtf> -gn <genesymbol.gtf> -o <output.gff3> -a <amount of datasets or datatype> -tp <tool name> -cg <category> -Ncg <new category> -pa <parent>
```

Input files:

`-i/<input.gtf>`. The GTF file produced in step 2 is the input file.

`-gn/<genesymbol.gtf>`. This option defines the assistant gene symbol GTF file (to generate the gene symbol GTF file, please check chapter 3.1.2.1 'Assistant files').

Output file (Overall GFF3 file):

`-o/<output.gff3>`. Writes the information records from both data types into the GFF3 format file.

The overall GFF3 file has nine columns per line. These columns are defined as follows in Table 3.11.

Table 3.11 Columns in the overall GFF3 file

Column number	Column name	Description	Example
1	*Scaffold ID*	*The Scaffold ID indicates where the sequence is located*	*NW_003613580.1*
2	*Source*	*The source is 'ExpressGENiE'*	*ExpressGENiE*
3	*Feature*	*The feature in this GFF3 file is based on the names that users assign to the parameters 'category', 'new category', and 'parent category'. The feature will be one of the three user-named options. E.g. region, all_regions, expressed_region*	*region*
4	*Start*	*The leftmost coordinate of this gene*	*690691*
5	*End*	*The rightmost coordinate of this gene*	*691138*
6	*Score*	*The score is '1'*	*1*
7	*Strand*	*The strand of the gene. Always one of '+', '-', '.'. It is inherited from the GTF3 file*	*+*
8	*Phase*	*The phase is '.'*	*.*
9	*Attributes*	*In this GFF3 file, the attributes will give the 'ID', 'Parent', and 'Derived_from' items. The 'ID' is consists of strand, scaffold ID, and dataset ID. 'Parent' is the ID of parent region. 'Derived_from' indicates which data type contains this region*	*ID=positive*NW_003613580.1*0*ExpressGENiE*1;Parent=positive*NW_003613580.1*0;Derived_from=RNA-Seq*

An example of the overall GFF3 file is shown below:

```
NW_003613580.1  ExpressGENiE    Expressed_region 690691
     691138     1    +    .
     ID=gene1;Name=LOC100757647;Derived_from=ExpressGENiE
NW_003613580.1  ExpressGENiE    region     690691     691138
     1    +    .
     ID=gene1*ExpressGENiE*1;Name=LOC100757647;Parent=gene1;De
rived_from=RNA-Seq
```

```
NW_003613580.1   ExpressGENiE     Expressed_region 736568
      738820     1     +     .     ID=positive*NW_003613580.1*1

NW_003613580.1   ExpressGENiE     region     736568     738820
      1     +     .
      ID=positive*NW_003613580.1*1*ExpressGENiE*1;Parent=positi
ve*NW_003613580.1*1;Derived_from=RNA-Seq
…
```

An example of the overall GFF3 track and two data type GFF3 tracks are shown in Figure 3.21. Figure 3.21 displays the reference gene track (boxed in red), the overall GFF3 track (boxed in green), the RNA-Seq data type GFF3 track (boxed in blue), and the DNA microarray data type track (boxed in yellow). Using the gene *Adam10* in the DNA microarray data type track as an example, the gene *Adam10* has been detected expressed in both data types. Thus, in the overall GFF3 track, the color of the region of *Adam10* is green.

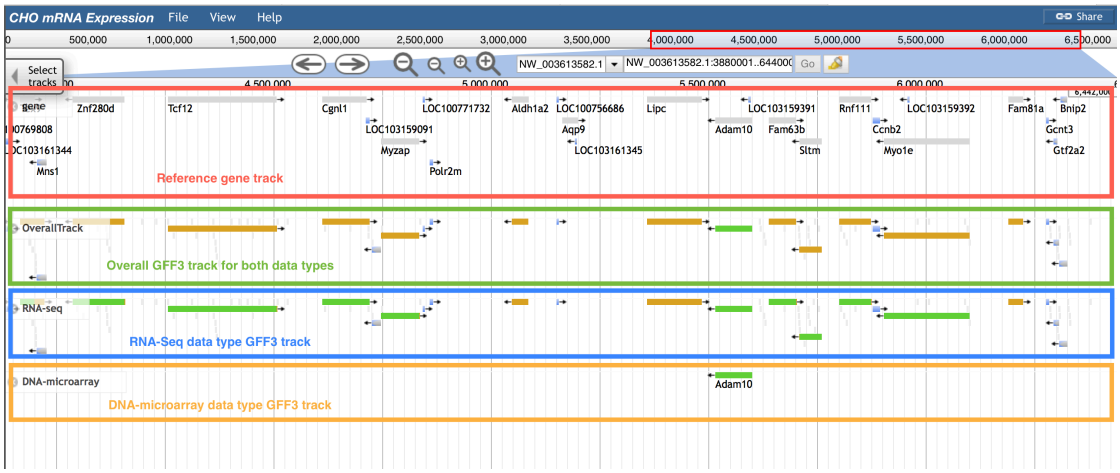

Figure 3.21 The overall GFF3 track.

In Figure 3.22, when the user clicks the region in the overall GFF3 track, the pop-up window indicates the data source.

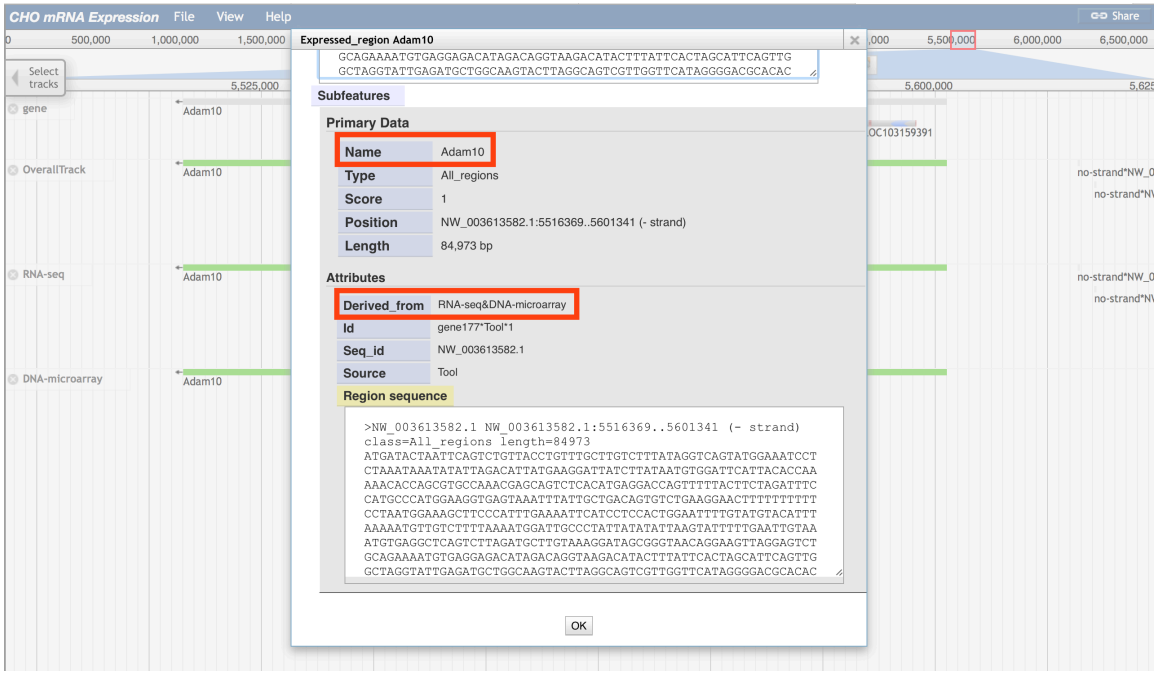

Figure 3.22 The pop-up window of overall GFF3 track.

3.1.2.5 The optional multiple bigWig XY track

The multiple bigWig XY track for each dataset is a track that is based on the bigWig files of samples. In a single track, it contains a merged set of bigWig tracks where each colored line represents the gene expression level of a sample.

The example of multiple bigWig XY track is shown in Figure 3.23, including the reference gene track (boxed in red), the multiple bigWig XY track of RNA-Seq dataset GSE75094 (boxed in green), and the four following tracks are sample bigWig tracks of GSE59487. Using the gene *Ipo5* in the multiple bigWig XY track as an

example, the four colors show distinctly different heights. Thus, the gene *Ipo5* has been detected as differentially expressed in four samples and under certain conditions, the gene expression level is much higher than others.

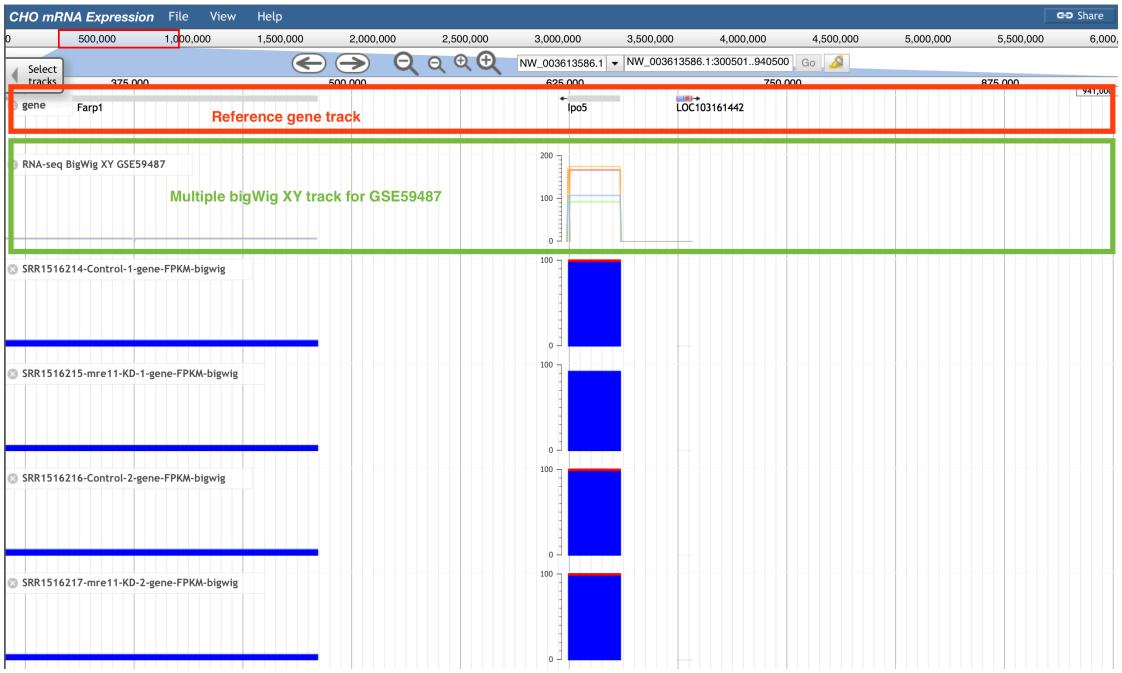

Figure 3.23 The multiple bigWig XY track of GSE58476.

Figure 3.24 displays the reference gene track (boxed in red), the multiple bigWig XY track of RNA-Seq dataset GSE75094 (boxed in green), and the multiple bigWig XY track of RNA-Seq dataset GSE59487 (boxed in blue). The dataset GSE75094 has ten samples and the dataset GSE59487 has four samples.

Figure 3.24 The multiple bigWig XY track.

For users who want to generate this multiple bigWig XY track for a dataset, they have to choose a unique color for each sample. In the multiple bigWig XY track of GSE75094, ten different colors are a challenge for users to distinguish. Thus, the multiple bigWig XY track is an optional track for users.

3.2 Usage of Tracks

3.2.1 Track selector

To choose which tracks to view, click the ‘Select tracks’ box (Figure 3.25, the ‘Select tracks’ is boxed in red). A menu listing all tracks will appear. In Figure 3.26, the track menu is boxed in red and the display window is boxed in green.

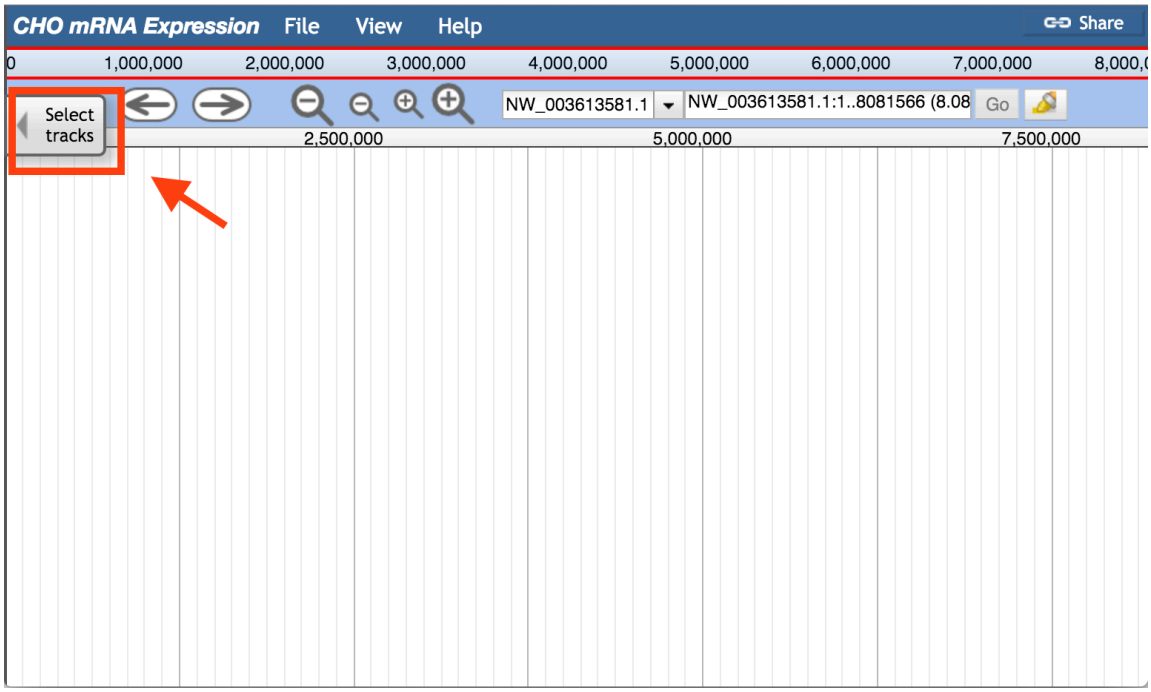

Figure 3.25 The 'Select tracks' box.

Figure 3.26 The track menu.

The track menu lists four headings:

1. ‘My Tracks’ contains the ‘Currently active’ and ‘Recently used’ options. ‘Currently active’ lists all the selected tracks in the display window. ‘Recently used’ lists all the recently selected tracks in the display window.

2. ‘Description’ contains the four categories: ‘DNA microarray’, ‘RNA-Seq’, ‘Reference Track’, and ‘Overall track’, which are classified by data types. The ‘Reference track’ contains three tracks, a DNA sequence reference track, a gene reference track, and a mRNA reference track. The ‘DNA microarray’ and ‘RNA-Seq’ are the two different data types that were derived from DNA microarray and RNA-Seq studies respectively. The ‘Overall track’ displays the DNA microarray and RNA-Seq data in one merged track.

3. ‘Data provider’ lists the raw dataset ID archived in the GEO database (http://www.ncbi.nlm.nih.gov/geo/). Users can search through the GEO database by using the dataset ID to obtain the raw data and a link to the NCBI or PubMed IDs, which can be used to get related documents.

4. ‘Category’ displays various categories of tracks. It classifies tracks into different groups and sub groups based on the data type and dataset of the uploaded track. For ‘DNA microarray’ or ‘RNA-Seq’ data types, its category has the format: data type/dataset/track group/track sub group (if needed).

In addition to the three categories shown on the track menu, the display window contains ‘Name’, ‘Species’, and ‘Organism’ columns that display more detailed descriptions for each track.

By selecting any option in the track menu, the display window displays all tracks that belong to the selected category. In the track menu, select options from top to bottom to go from a general group of tracks to a more specific set.

For new users, the ‘My Tracks’ section will be empty. For returning users, ‘My Tracks’ provides a shortcut to review the currently selected or recently used tracks.

The ‘Description’ section lists ‘DNA microarray’, ‘RNA-Seq’, ‘Reference Track’, and ‘Overall track’ options. Users need to set a reference track before selecting either ‘DNA microarray’ or ‘RNA-Seq’ data type, because the reference track can provide a good indicator of what exists (genes or exons or transcripts or nucleotides) at a certain location in the genome. Three reference tracks are available: ‘DNA’, ‘gene’, and ‘mRNA’ tracks.

In Figure 3.27, the reference DNA sequence track (boxed in red), reference gene track (boxed in green), and reference mRNA track (boxed in blue) are displayed. The reference DNA sequence track displays nucleotide sequences, shown as color blocks. The reference gene track displays gene symbol/ID and location. The reference mRNA track indicates possible exons, introns, and coding sequences (CDs) for each gene (can click the track for details).

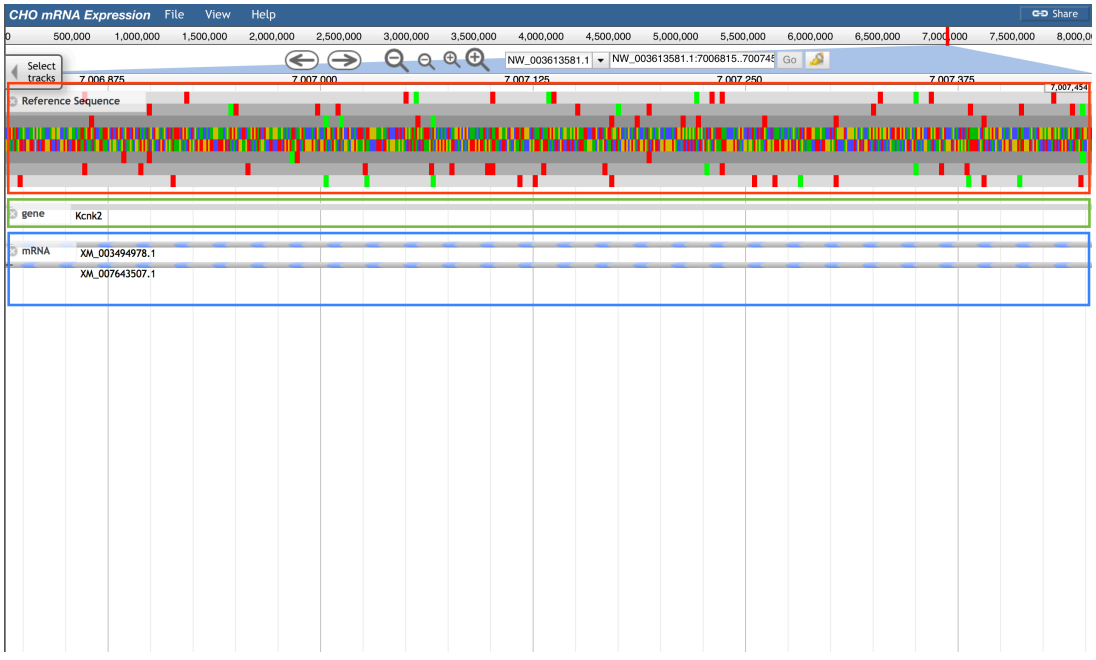

Figure 3.27 Reference tracks.

After selecting a reference track, a user can find the specific locations of the genes of interest. If users already have a data type of interest and want to focus on it, they can select one data type to start searching. For other users who want to get more data type information for their genes of interest, they can go to ‘Overall track’.

The ‘Overall track’ in description option only contains one track, named ‘Overall track’ that is a GFF3 track containing the mRNA expression information from both data types. Thus, the regions shown in this track as blocks are regions that have been expressed in a certain experiment (either DNA microarray or RNA-Seq) or several experiments (DNA microarray and RNA-Seq).

In Figure 3.28, when users select ‘Overall track’, there is only one selection in the next two options (‘Data provider’ and ‘Category’).

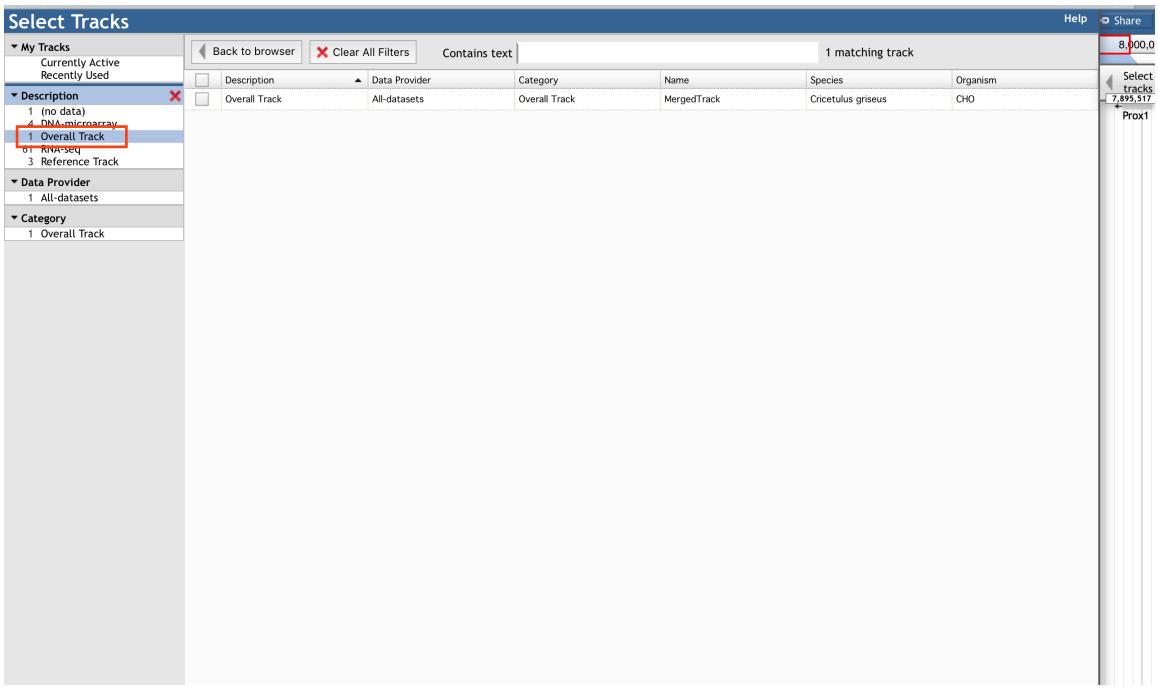

Figure 3.28 The ‘Overall track’ in track menu.

In Figure 3.29, the reference gene track (boxed in red) and overall track (boxed in green) are displayed. When zoomed in, users can see that the overall track has two colors. The two different colors are used to distinguish the sources of data. If the sequence can be found expressed in every source, the sequence is shown as a green block. If at least one source did not express that sequence, the sequence is shown as a yellow block.

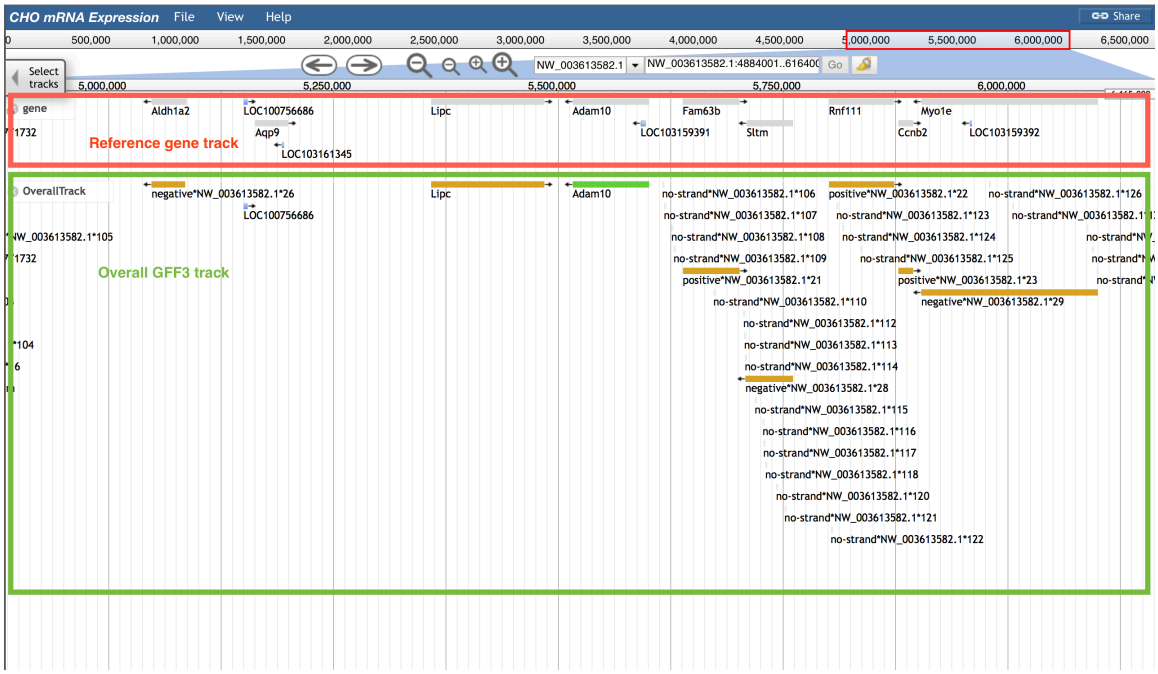

Figure 3.29 The display of ‘Overall track’.

In Figure 3.30, clicking on any part of the blocks (a block may consists of several continuous sequences/parts), a pop-up window will provides more details about the continuous sequence. The ‘Overall track’ is a GFF3 format file. For example, in a pop-up window, it will contain basic sequence information such as its

name and position. In the ‘Subfeatures’ section, it provides information for each part, including which data type contains this region.

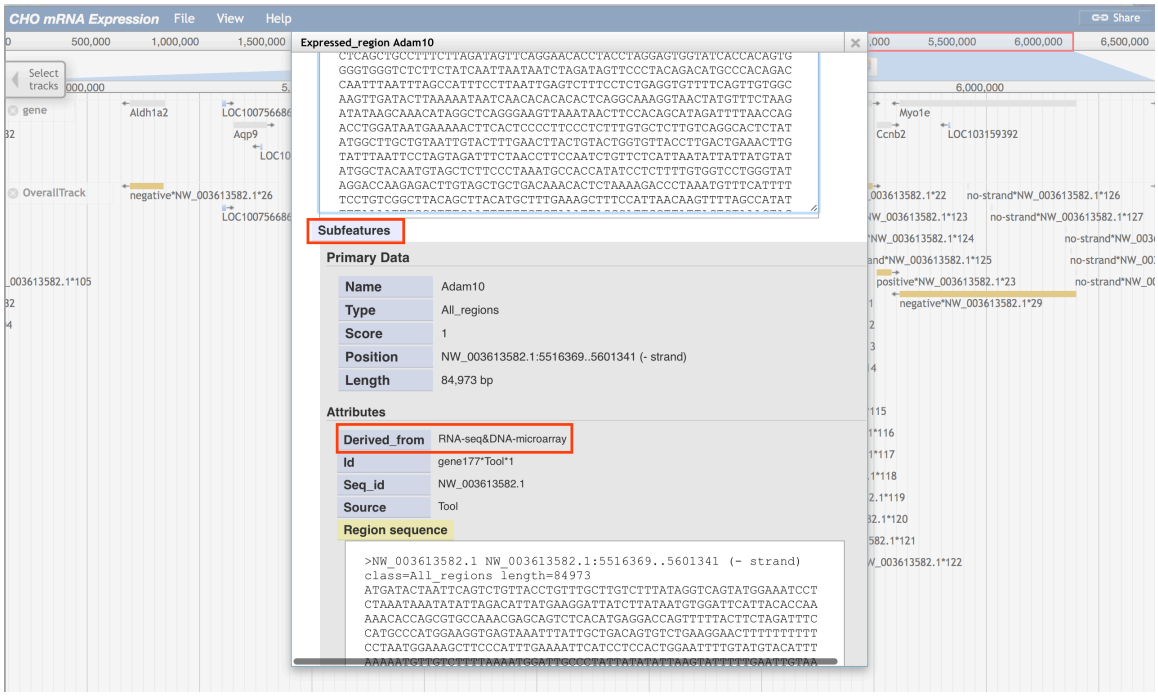

Figure 3.30 Pop-up window of ‘Overall track’.

In this tutorial, there is one DNA microarray dataset, GSE30321 [Clarke et al., 2011], and two RNA-Seq datasets, GSE59487 [Kondratova et al., 2015] and GSE75094 [Lee et al., 2016].

To gain more details about the gene expression information, select ‘Description’ and ‘Data Provider’. The RNA-Seq data type was presented as an example in Figure 3.31. If ‘RNA-Seq’ is selected in the ‘Description’ option, the ‘Data provider’ lists only the RNA-Seq dataset IDs and the display window lists the RNA-Seq data type tracks.

Figure 3.31 The ‘RNA-Seq’ data type track in track menu.

For either DNA microarray data or RNA-Seq data, there is a track for the entire data type. The ‘Data provider’ lists two different types of data provider, one is the dataset ID, another is the ‘All-RNAseq(data type)-datasets’. The data type track is within the ‘All-data type-datasets’ provider and use ‘data type’ (e.g. RNA-Seq) as track name.

The data type track is a GFF3 track containing gene expression information from all datasets that belong to the same data type. The data type track can be viewed in a similar way to the ‘Overall track’. The blocks in this track represent mRNAs expressed in a certain dataset.

In Figure 3.32, the ‘RNA-Seq’ is under the sub category ‘RNA-Seq/Merged Track’. There is only one track in this sub category.

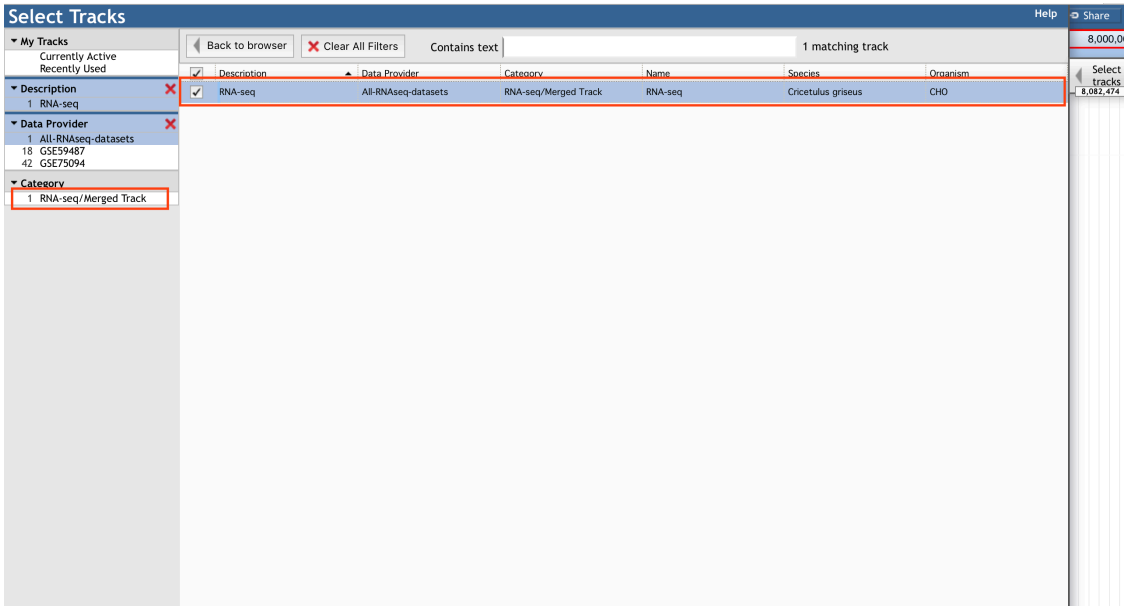

Figure 3.32 The RNA-Seq data type track in track menu.

In Figure 3.33, the display of the merged track for a data type is similar to the overall track. In the attributes section, the ‘Derived_from’ value indicates the dataset ID of the selected block (Figure 3.34).

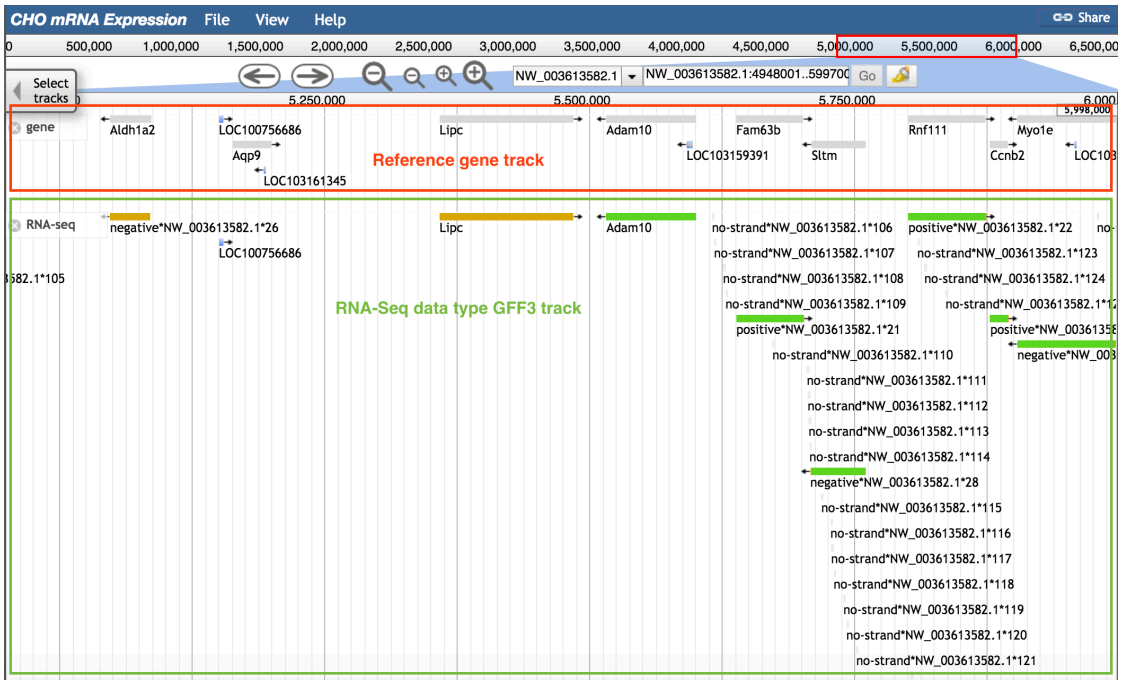

Figure 3.33 The display of RNA-Seq data type track.

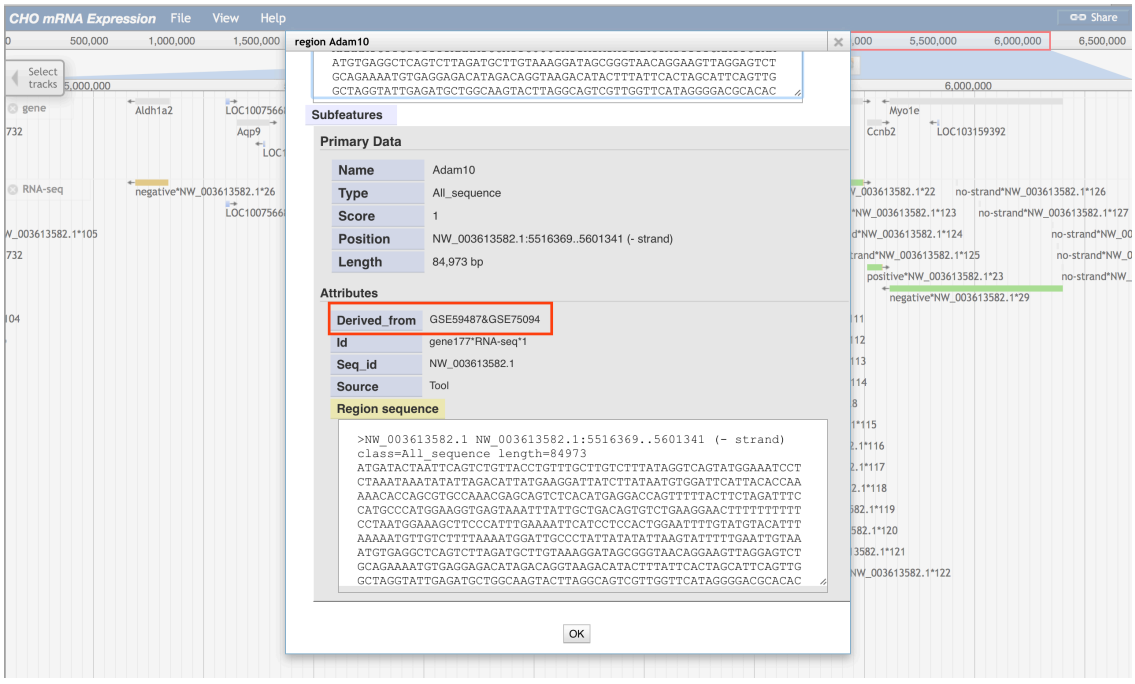

Figure 3.34 Pop-up window of RNA-Seq data type track.

The data type can be used to obtain the general gene expression information from previous experiment. For instance, from the RNA-Seq data type track, users can get which dataset contains the expressed genes of interest. By selecting a list of datasets that contain a gene or region of interest, users can find specific information by going through the datasets one by one.

In addition to the GFF3 file for all datasets (Overall track) for each data type (data type track), there is another GFF3 file for each dataset. Here, we use the RNA-Seq dataset GSE75094 as an example.

In Figure 3.35, The GFF3 track for each RNA-Seq dataset is under the sub category ‘RNA-Seq/dataset ID’.

Figure 3.35 The dataset track in track menu.

In Figure 3.36, the sub category contains two tracks, the ‘dataset track’ (the track name is the dataset ID), and the ‘differential track’.

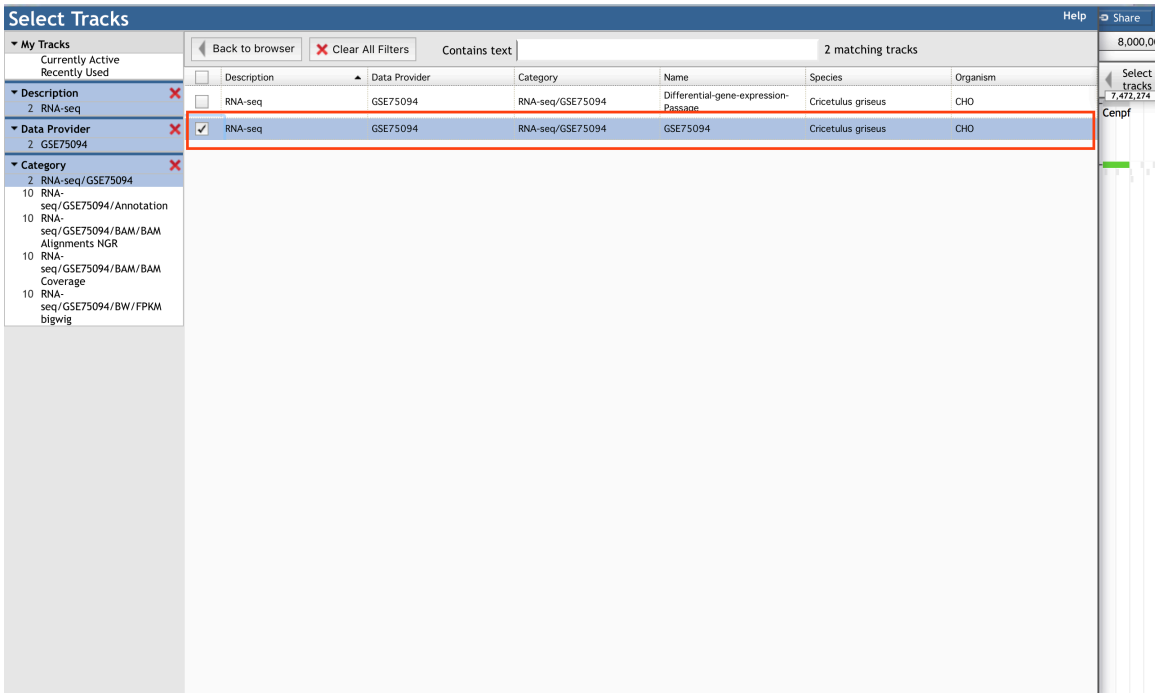

Figure 3.36 The GSE75094 dataset track in track menu.

In Figure 3.37, the usage of GFF3 track for dataset is similar as the GFF3 track for data type. In the dataset track, the source/dataset ID is replaced with the sample ID (Figure 3.38).

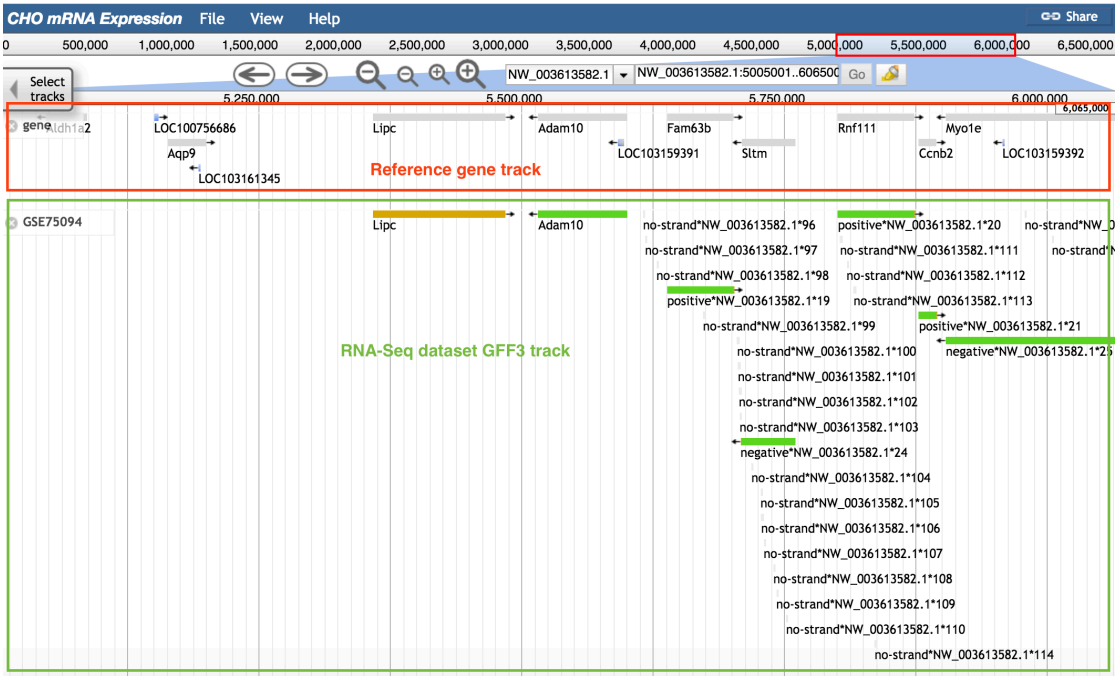

Figure 3.37 The display of GSE75094 dataset track.

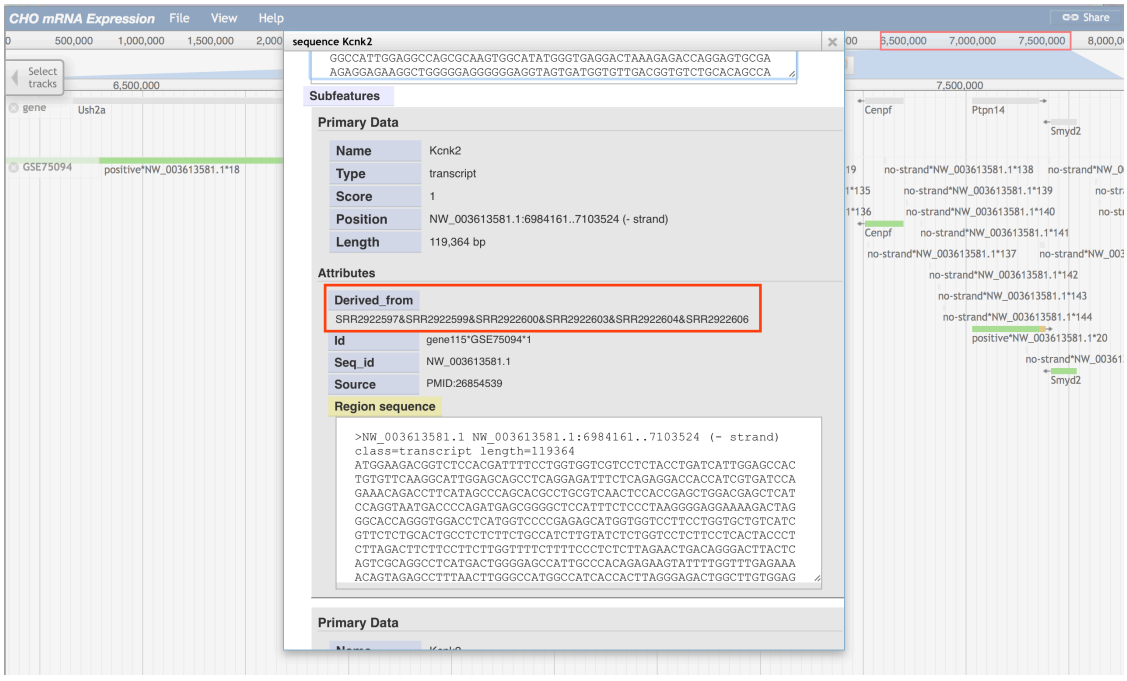

Figure 3.38 The pop-up window of GSE75094 dataset track.

From the dataset track, a user can compare gene or mRNA expression situation among all samples or examine if a certain gene is expressed in different samples or conditions.

3.2.2 DNA microarray data

DNA microarray datasets, except the GTF track for the dataset, have only one type of track, a bigWig track. In a bigWig track, the position of each block represents the position of an expressed gene. The height of the block represents the gene expression level. There is a scale bar in the track for a reference. A red line will be used to highlight the top layer of the block when a gene expression value is higher than the maximum value on the scale. In addition, hovering the cursor over the blocks in a bigWig track will display the gene expression value measured for the condition.

After going through the 'Description' and the 'Data provider' sections, users usually have to keep the reference gene track selected. Users can go to the scaffold and by using the reference gene track to find the gene of interest. In addition, selecting the dataset GTF track displays the expressed gene symbol/ID, as shown in the Figure 3.39.

In Figure 3.39, the reference gene track (boxed in red), dataset GTF track (boxed in green) and the bigWig track (boxed in blue) are displayed. For example, the gene of interest is *Adam10*. In the dataset GTF track, users can identify the name of the expressed gene *Adam10*. Hovering the cursor over the block in the bigWig track shows the value '4.85232', which is the average gene expression value of *Adam10* in the dataset GSE30321. When users are interested in further information, they can click the block in the dataset GFF3 track (Figure 3.40). From a pop-up window, users can gain the dataset ID and source ID for raw data and related paper.

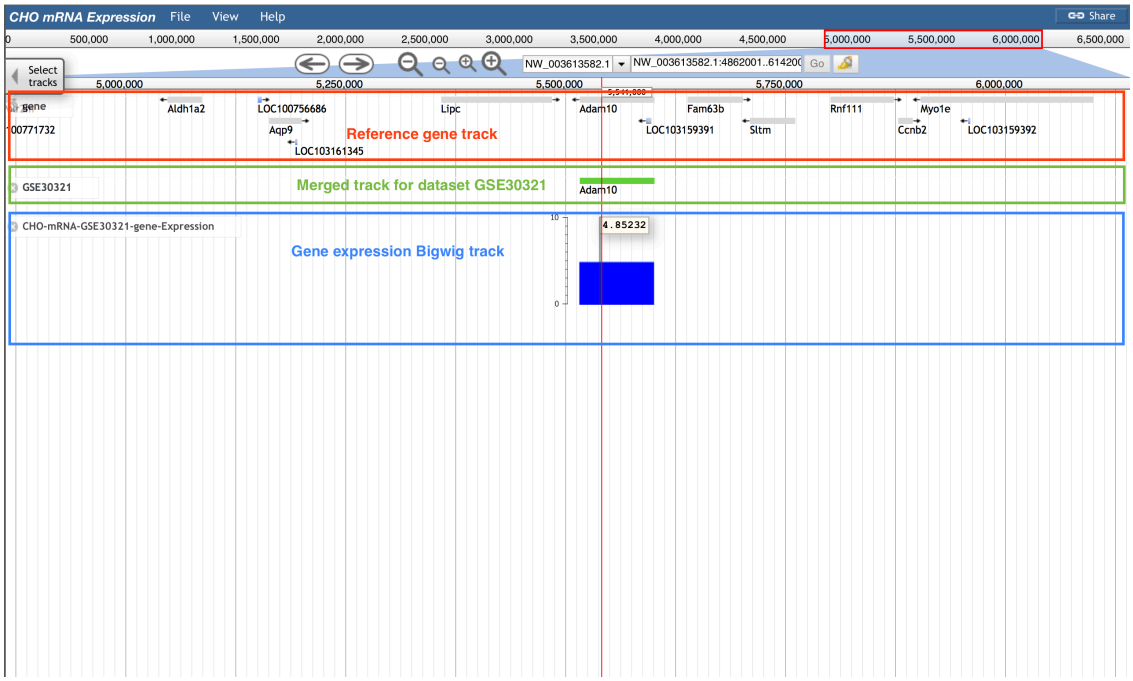

Figure 3.39 The display of GSE30321 dataset track and bigWig track.

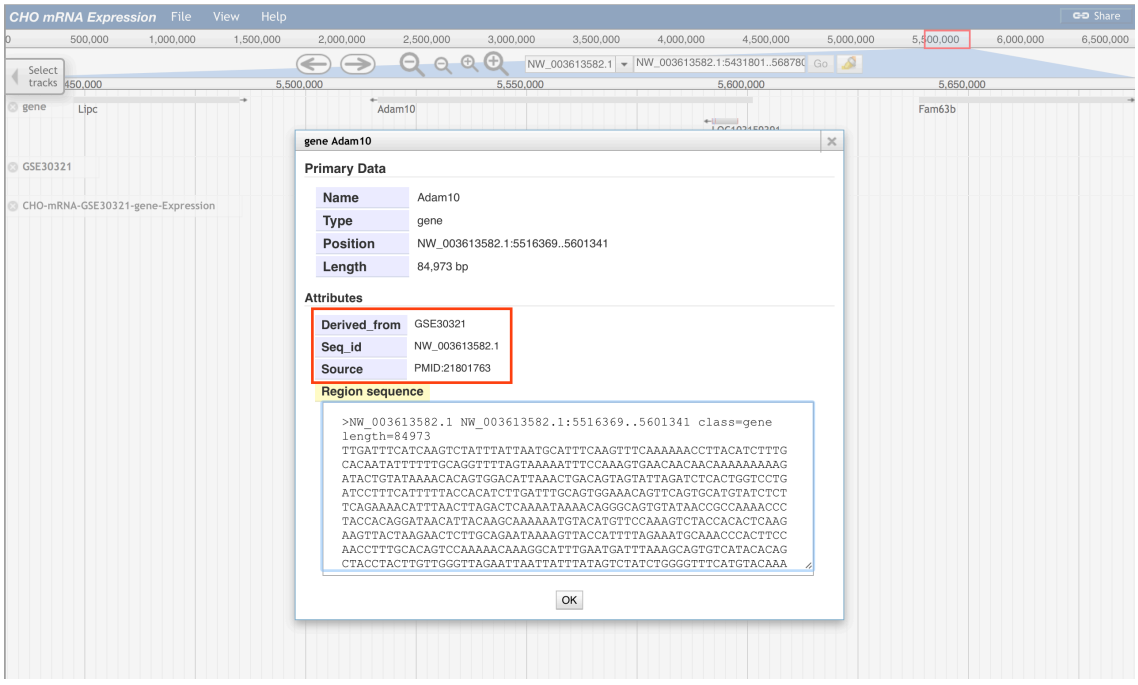

Figure 3.40 The pop-up window of GSE30321 dataset track.

In the example GSE30321, only one bigWig track was generated because the dataset is complex and has various sample conditions. For other DNA microarray datasets, group can be classified based on samples' condition. Then users can compare the expression value of a gene of interest among different groups by using the bigWig tracks for each group.

3.2.3 RNA-Seq data

For each RNA-Seq dataset, several types of tracks are provided for visualizing the gene expression information.

For each sample, the basic reads mapping information has been kept in two 'BAM' file categories: the BAM coverage feature track and the BAM alignment feature track. The 'BAM coverage feature track' is similar to a bigWig track. It is composed of histograms to represent the coverage of RNA-Seq reads in the given position. It also has a scale bar, and uses a red top to indicate values that are beyond the scale.

The 'BAM alignment feature track' shows read alignment for the condition/sample. Each dot represents a read and its position indicates where the read was mapped to the scaffold. There are two colors of the dots to distinguish the direction of the read. Red indicates a read mapping to the positive strand, and the blue indicates the read mapped to the negative strand. Additionally, a black bar within a read indicates that the read has been separated into several pieces in order to map to the scaffold (Figure 3.41).

In Figure 3.41, the reference gene track (boxed in red), the coverage BAM track (boxed in green) and alignment BAM track (boxed in blue) are displayed. *Adam10*, an example of gene of interest, the alignment BAM track displays many

reads and how these reads were mapped to the genome. By hovering the cursor over the coverage BAM track, users can identify the density of reads in the given position.

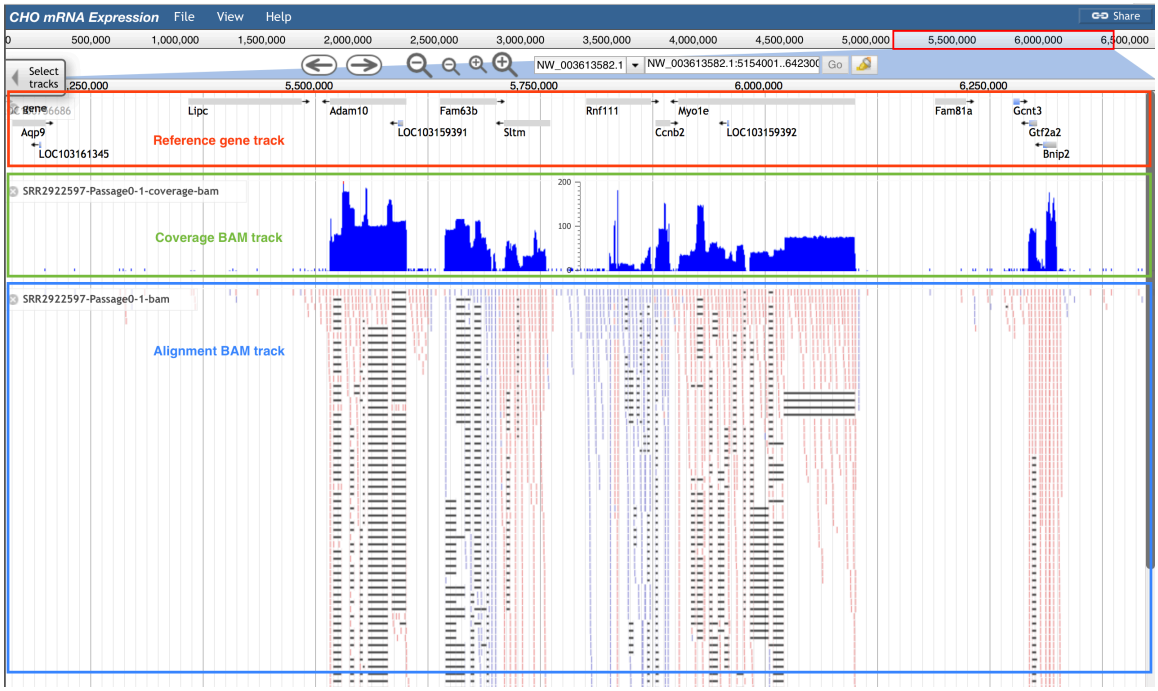

Figure 3.41 The display of sample BAM tracks.

In addition to basic reads information, the assembly of these reads into a transcript provides prediction for exons and transcripts. The ‘Annotation track’, which includes annotation of expressed transcripts for each gene, can be used to review the assembly. When using the ‘Annotation track’, it is recommended to also use the reference gene track and reference mRNA track. By comparing the reference gene and mRNA tracks with the ‘Annotation track’, users can obtain a general idea of the sample expression condition.

In Figure 3.42, the reference gene track (boxed in red), reference mRNA track (boxed in green), and annotation GFF3 track (boxed in blue) are displayed. Here we use gene *Adam10* and gene *Ccnb2* as examples. For gene *Adam10*, both the reference mRNA track and the annotation GFF3 track have one mRNA, but the gene ID of mRNA in the annotation GFF3 track (gene id: CUFF.254) is different with the gene ID in the reference gene track. For gene *Ccnb2*, the gene ID of mRNA in the annotation GFF3 track (*Ccnb2*) is same as the gene ID in the reference gene track. Same gene ID means that the sequence of this gene in the annotation GFF3 track is entirely the same as the sequence of this gene in the reference gene track. Therefore, the gene *Ccnb2* was more consistent with the reference tracks during the experiment. Compared to the gene *Adam10* (gene IDs are different in reference gene track and annotation GFF3 track), we have higher confidence in saying *Ccnb2* was expressed in the sample.

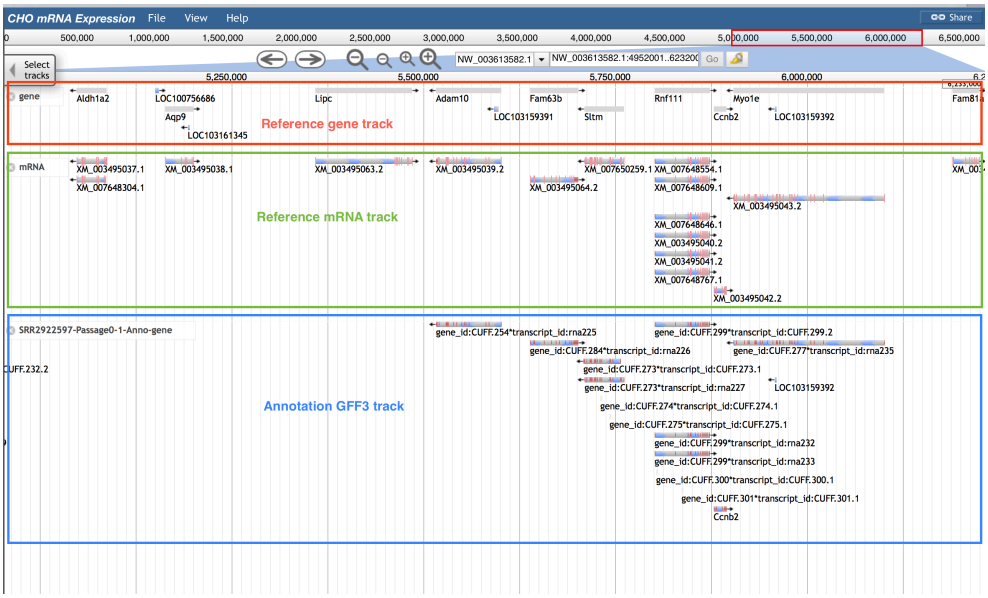

Figure 3.42 The display of sample annotation GFF3 track.

In Figure 3.43, users can click each part of the block in the annotation GFF3 track to obtain annotation information about the selected mRNA (pop-up window).

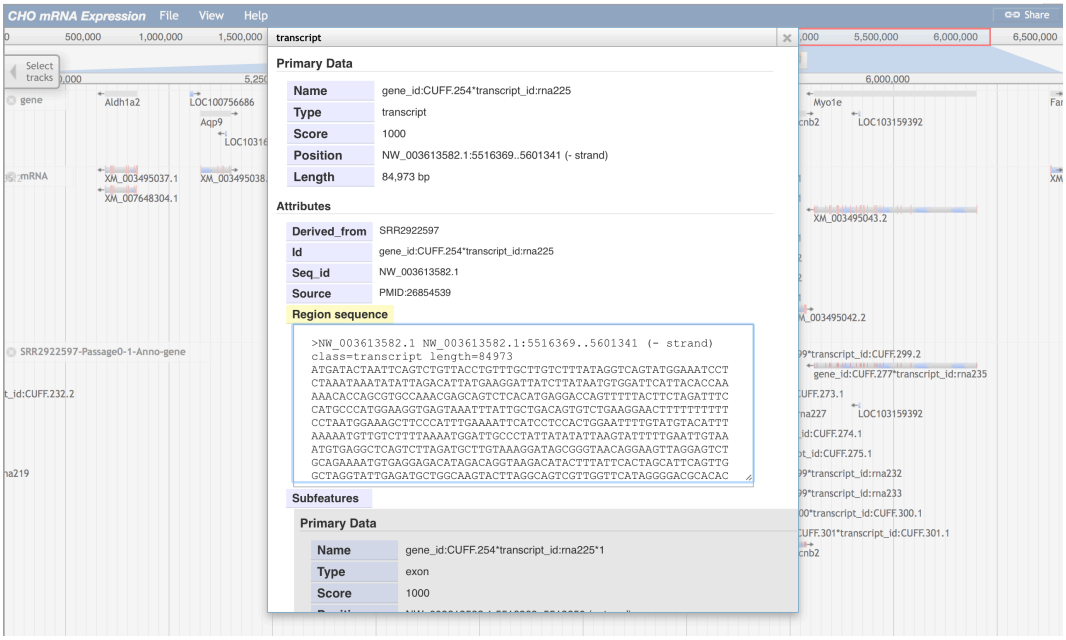

Figure 3.43 The pop-up window of sample annotation GFF3 track.

The ‘FPKM bigWig track’ displays gene level FPKM values and has bigWig track features. The height of each column and its value in this bigWig track represent the gene expression FPKM value. By combining the annotation track with the FPKM bigWig track, users can identify the expression level for a given gene (Figure 3.44).

In Figure 3.44, the reference gene track (boxed in red), annotation GFF3 track (boxed in green), and FPKM bigWig track (boxed in blue) are displayed. The gene expression level of *Adam10* is 39.2699.

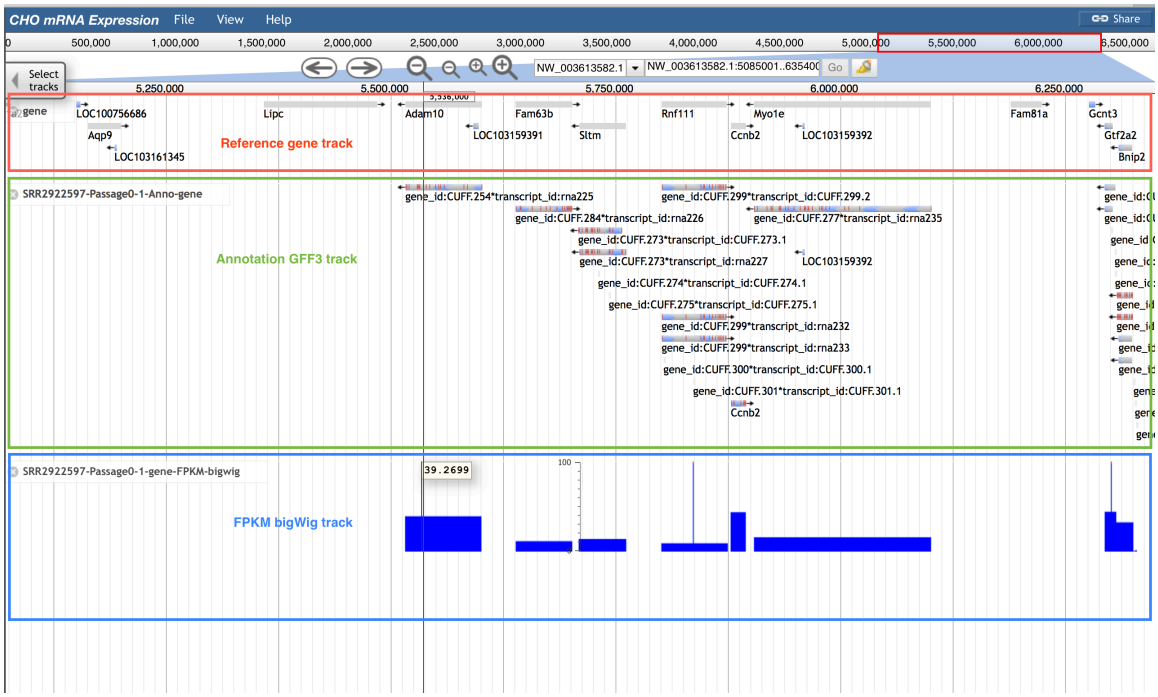

Figure 3.44 The display of sample annotation GFF3 track and FPKM bigWig track.

The four tracks explained above are used to display reads mapping, assembly and gene expression level for each sample. To make a comparison of gene expression level among different samples or conditions, users can select a 'Differential gene expression track'.

In the differential gene expression track, a green block means that the gene in the selected position is differently expressed between paired conditions.

In Figure 3.45, the reference gene track (boxed in red), dataset GFF3 track (boxed in green), and the differential GTF track (boxed in blue) are displayed. For instance, gene *Adam10* was expressed (shown block in dataset GFF3 track), but it was not differentially expressed between any paired conditions (shown as no block in the

differential GTF track). On the contrary, the gene *Sltm* has one green block, which means the gene *Sltm* was expressed differentially in paired conditions.

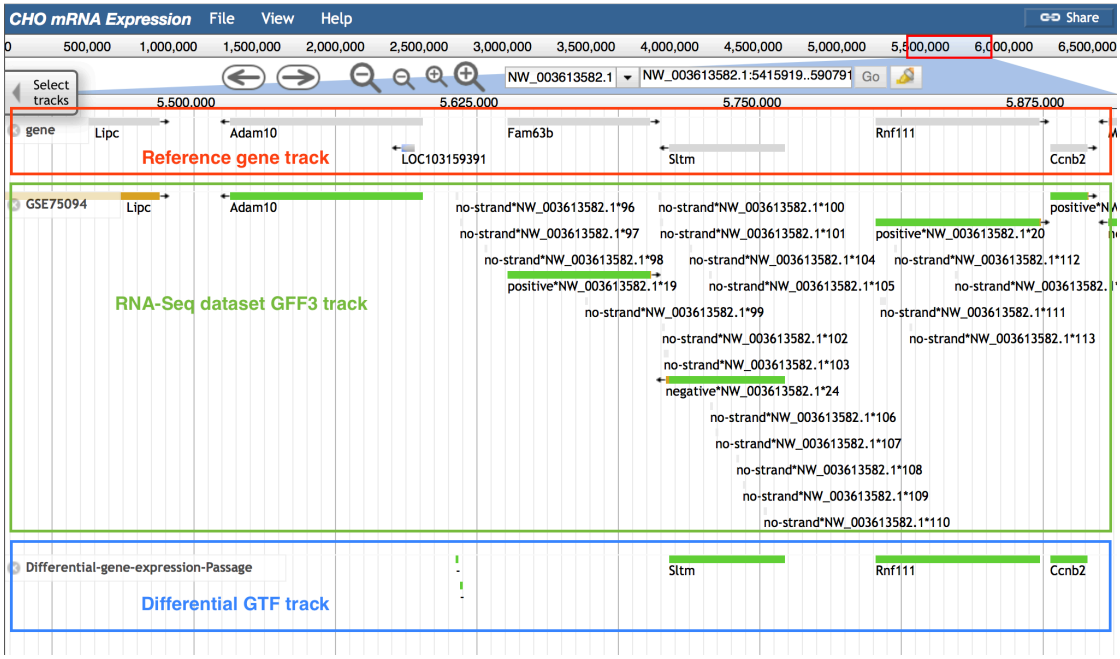

Figure 3.45 The displaying of GSE75094 dataset track and differential GTF track.

When users click on a block, a pop-up window provides more information about which two groups expressed the gene differently. In Figure 3.46, click the block of gene *Sltm*. The 'Derived_from' attribute displays the experimental condition and the gene expression level under the condition. Condition one is 'Passage0', and the expression level is '17.9174'. Condition two is 'Passage 6', and the expression level is '49.9713'.

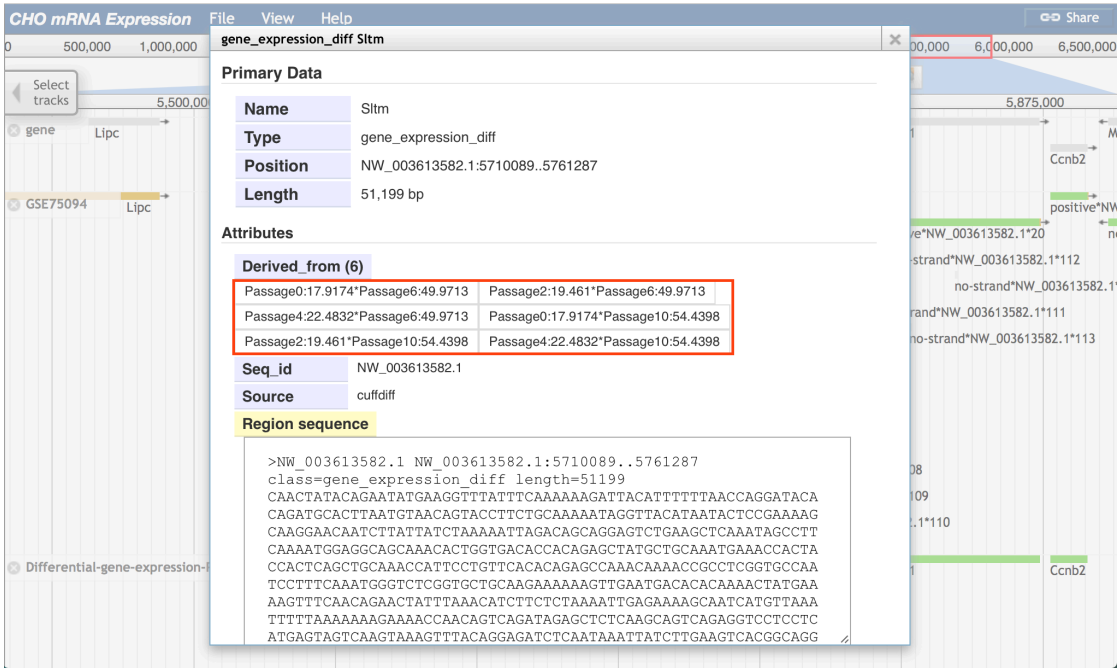

Figure 3.46 The pop-up window of differential GTF track.

Chapter 4

DISCUSSION

When studying CHO cells, mRNA expression data from DNA microarrays or RNA-Seq is essential for scientists to understand how genes are expressed under various conditions. Accordingly, various tools have been generated for processing processed DNA microarray and RNA-Seq data. There are tools used to measure or calculate a numerical value to represent the gene expression level, and tools used for comparing differential gene expression within a dataset for both DNA microarray and RNA-Seq data. However, files that contain gene expression data are not easy to visualize because the file formats restrict the visualization of expression level for each gene.

Currently, visualizations of the comparison result for gene expression include viewing differentially expressed genes, such as the output from edgeR. This visualization enables users to easily make a comparison for the expression level of a group of genes among different conditions, but it is not easy for users to view the expression level of a certain gene. Some genome browsers have also been used to attempt to visualize gene expression data, such as JBrowse, UCSC genome browser, and Integrative genome viewer (IGV). These genome browsers more or less can visualize part of the gene expression information. For example, reads mapping information can be visualized by using BAM file. However, users are not able to obtain transcript or gene expression level from this limited visualization information. In addition, there is no source available for the CHO community to check the

expression level of certain genes from past DNA microarray or RNA-Seq experiments.

This study is aim to provide a better way to visualize mRNA/gene expression data by creating new format files that contain data of interest and are compatible with genome viewers. For each DNA microarray experiment, a normalized gene expression level is provided. For each RNA-Seq experiment, further analysis tools, such as Tuxedo protocol, are required to calculate gene expression levels. Based on these available data and tools, ExpressGENiE has been developed to process data from either normalized DNA microarray data or RNA-Seq data after processed by Tuxedo protocol.

The format of data from DNA microarray and RNA-Seq experiments cannot be visualized directly to represent mRNA/gene expression levels. The formats of the newly generated files, such as bigWig/bedGraph, GTF, and GFF3, are commonly used and can be visualized on JBrowse as well as other genome browsers. These newly generated files help to reveal different aspects of the mRNA/gene expression data. For both DNA microarray and RNA-Seq data, there are files to indicate the symbols of expressed genes and display their numerical gene expression levels. For users who have genes of interest, the visualization of gene expression level files can be used to search for numerical expression levels from past experiments. In addition, for users who want to make a gene expression comparison for a certain gene, there is a file generated for each RNA-Seq dataset to better display differentially expressed genes.

There are also multiple files generated that contain gene expression information for CHO cells from both DNA microarray and RNA-Seq data. Due to the fact that when a new experiment is added, the increasing number of files will make it

more difficult to search for information on genes of interest, there are files generated that indicate the source of each expressed sequence. These files enable users to find whether or not their gene of interest is expressed in a certain data type/dataset/sample. The usage of the source indicator files simplifies the process to obtain information for genes of interest.

However, currently, the tool is unable to compare the gene expression level between two datasets. Make a comparison between two datasets is challenging to implement because there is no process to normalize the gene expression level for a certain gene between two different RNA-Seq or DNA microarray datasets. In addition, there is no existing file format that allows visualization of differential gene expression for DNA microarray data in JBrowse. Even though there are tools available to compare gene differential expression for DNA microarray data, the outputs are unfriendly to visualize on JBrowse.

The main purpose of this study is to directly visualize the gene expression level for CHO cells. The chosen browser, JBrowse, has been used to display CHO cell DNA microarray and RNA-Seq data. Thus, for the CHO community, all visualized files from past experiments are put together and setup as a database so that people do not need to collect these information individually. The visualization of gene expression data enables users to directly observe the gene expression level for their genes of interest. The gene expression level from past experiments can provide a reference and basic information of how gene is expressed under certain culture condition, and thus users can better plan their experiments. Additionally, the visualization of data enables users to compare the gene expression level from various conditions. For RNA-Seq

data, there are files containing differentially expressed genes, which enable users to directly identify differentially expressed genes.

Using this tool to generate new files provides a convenient way to visualize gene expression levels. To enrich the function of the tool, there are still several aspects that need to be improved. The future work includes the following points:

1. ExpressGENiE is based on running multiple custom Python scripts to generate new files. For existing functions, improving the efficiency of these scripts can decrease running time of each procedure.

2. There are currently only a few datasets available to view in JBrowse. As the number of studies of CHO cells using DNA microarray or RNA-Seq methods increase, adding more datasets into JBrowse would expand the CHO data in CHOgenome.org.

3. Generating files to visualize the differential gene expression for DNA microarray data would enable users to compare gene expressions among different conditions within a DNA microarray experiment. Fold-change and p-value can be used as the standard to select whether or not the gene was expressed differentially.

4. Generating files to visualize the differential gene expression for either multiple DNA microarray datasets or multiple RNA-Seq datasets would enable users to easily make comparisons for gene expressions among different datasets. Gene expression levels should be normalized before making the comparison among different datasets. DNA microarray data may have been normalized differently among experiments due to multiple normalization methods. In addition, control groups of various RNA-Seq experiments may have different gene expression levels for the same gene.

5. In addition, ExpressGENiE currently works only for CHO-K1 data. ExpressGENiE is based on reference gene sequence and reference annotation files to process data. Therefore, with the completion of reference files for other CHO cell lines, ExpressGENiE can be used to process data for different CHO cell lines.

REFERENCES

Anders, S., & Huber, W. (2010). Differential expression analysis for sequence count data. *Genome Biology*, *11*(10), R106.

Baik, J. Y., Lee, M. S., An, S. R., Yoon, S. K., Joo, E. J., Kim, Y. H., ... & Lee, G. M. (2006). Initial transcriptome and proteome analyses of low culture temperature-induced expression in CHO cells producing erythropoietin. *Biotechnology and Bioengineering*, *93*(2), 361-371.

Baik, J. Y., Ha, T. K., Kim, Y. H., & Lee, G. M. (2011). Proteomic understanding of intracellular responses of recombinant chinese hamster ovary cells adapted to grow in serum-free suspension culture. *Biotechnology Progress*, *27*(6), 1680-1688.

Becker, J., Hackl, M., Rupp, O., Jakobi, T., Schneider, J., Szczepanowski, R., ... & Kaltschmidt, C. (2011). Unraveling the Chinese hamster ovary cell line transcriptome by next-generation sequencing. *Journal of biotechnology*, *156*(3), 227-235.

bedGraphToBigWig. https://www.encodeproject.org/software/bedgraphtobigwig/

Benjamin, A. M., Nichols, M., Burke, T. W., Ginsburg, G. S., & Lucas, J. E. (2014). Comparing reference-based RNA-Seq mapping methods for non-human primate data. *BMC Genomics*, *15*(1), 570.

Berlec, A., & Štrukelj, B. (2013). Current state and recent advances in biopharmaceutical production in Escherichia coli, yeasts and mammalian cells. *Journal of industrial microbiology & biotechnology*, *40*(3-4), 257-274.

Birzele, F., Schaub, J., Rust, W., Clemens, C., Baum, P., Kaufmann, H., ... & Hildebrandt, T. (2010). Into the unknown: expression profiling without genome sequence information in CHO by next generation sequencing. *Nucleic acids research*, *38*(12), 3999-4010.

Brown TA. Genomes. 2nd edition. Oxford: Wiley-Liss; 2002. Chapter 3, Transcriptomes and Proteomes. Available from: http://www.ncbi.nlm.nih.gov/books/NBK21121/

Buels, R., Yao, E., Diesh, C. M., Hayes, R. D., Munoz-Torres, M., Helt, G., … Holmes, I. H. (2016). JBrowse: a dynamic web platform for genome visualization and analysis. *Genome Biology*, *17*, 66.

Bumgarner, R. (2013). DNA microarrays: Types, Applications and their future. *Current Protocols in Molecular Biology / Edited by Frederick M. Ausubel ... [et Al.]*, *0 22*, Unit–22.1.

CHO genome. http://www.chogenome.org.

Dingermann, T. (2008). Recombinant therapeutic proteins: production platforms and challenges. *Biotechnology journal*, *3*(1), 90-97.

Gene expression omnibus (GEO) database. http://www.ncbi.nlm.nih.gov/geo/

Goeddel, D. V., Kleid, D. G., Bolivar, F., Heyneker, H. L., Yansura, D. G., Crea, R., ... & Riggs, A. D. (1979). Expression in Escherichia coli of chemically synthesized genes for human insulin. *Proceedings of the National Academy of Sciences*, *76*(1), 106-110.

Grillberger, L., Kreil, T. R., Nasr, S., & Reiter, M. (2009). Emerging trends in plasma-free manufacturing of recombinant protein therapeutics expressed in mammalian cells. *Biotechnology journal*, *4*(2), 186-201.

Hammond, S., Kaplarevic, M., Borth, N., Betenbaugh, M. J., & Lee, K. H. (2012). Chinese hamster genome database: an online resource for the CHO community at www. CHOgenome. org. *Biotechnology and bioengineering*, *109*(6), 1353-1356.

Harrington, C. A., Rosenow, C., & Retief, J. (2000). Monitoring gene expression using DNA microarrays. *Current opinion in Microbiology*, *3*(3), 285-291.

Integrative genomics viewer (IGV). http://software.broadinstitute.org/software/igv/

Jayapal, K. P., Wlaschin, K. F., Hu, W., & Yap, M. G. (2007). Recombinant protein therapeutics from CHO cells-20 years and counting. *Chemical Engineering Progress*, *103*(10), 40.

Kantardjieff, A., & Zhou, W. (2014). Mammalian cell cultures for biologics manufacturing. *Advances in Biochemical Engineering/Biotechnology*, 139,1-9.

Kim, J. Y., Kim, Y. G., & Lee, G. M. (2012). CHO cells in biotechnology for production of recombinant proteins: current state and further potential. *Applied microbiology and biotechnology*, *93*(3), 917-930.

Kondratova, A., Watanabe, T., Marotta, M., Cannon, M., Segall, A. M., Serre, D., & Tanaka, H. (2015). Replication fork integrity and intra-S phase checkpoint suppress gene amplification. *Nucleic acids research*, *43*(5), 2678-2690.

Korke, R., de Leon Gatti, M., Lau, A. L. Y., Lim, J. W. E., Seow, T. K., Chung, M. C. M., & Hu, W. S. (2004). Large scale gene expression profiling of metabolic shift of mammalian cells in culture. *Journal of biotechnology*, *107*(1), 1-17.

Kremkow, B. G., Baik, J. Y., MacDonald, M. L., & Lee, K. H. (2015). CHOgenome. org 2.0: Genome resources and website updates. *Biotechnology journal*, *10*(7), 931-938.

Lee, N., Shin, J., Park, J. H., Lee, G. M., Cho, S., & Cho, B. K. (2016). Targeted Gene Deletion Using DNA-Free RNA-Guided Cas9 Nuclease Accelerates Adaptation of CHO Cells to Suspension Culture. *ACS synthetic biology*.

Lens, J., & Evertzen, A. (1952). The difference between insulin from cattle and from pigs. *Biochimica et biophysica acta*, *8*, 332-338.

Lewis, N. E., Liu, X., Li, Y., Nagarajan, H., Yerganian, G., O'Brien, E., ... & Xie, M. (2013). Genomic landscapes of Chinese hamster ovary cell lines as revealed by the Cricetulus griseus draft genome. *Nature biotechnology*, *31*(8), 759-765.

Rat genome database. http://rgd.mcw.edu

Robinson, M. D., McCarthy, D. J., & Smyth, G. K. (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, *26*(1), 139–140.

Rupp, O., Becker, J., Brinkrolf, K., Timmermann, C., Borth, N., Pühler, A., ... & Goesmann, A. (2014). Construction of a public CHO cell line transcript database using versatile bioinformatics analysis pipelines. *PloS one*, *9*(1), e85568.

Sanny, A., Kok, Y. J., Philip, R., Chuah, S. H., Ng, S. W., Tan, K. S., ... & Nissom, P. M. (2006). Identifying key signatures of highly productive CHO cells from transcriptome and proteome profiles. *Microbial Cell Factories*, *5*(1), P96.

Seth, G., Philp, R. J., Lau, A., Jiun, K. Y., Yap, M., & Hu, W. S. (2007). Molecular portrait of high productivity in recombinant NS0 cells. *Biotechnology and bioengineering*, *97*(4), 933-951.

Shen, D., & Sharfstein, S. T. (2006). Genome-wide analysis of the transcriptional response of murine hybridomas to osmotic shock. *Biotechnology and bioengineering*, *93*(1), 132-145.

Single-end reads and paired-end reads. http://www.yourgenome.org/facts/how-do-you-put-a-genome-back-together-after-sequencing

Skinner, M. E., Uzilov, A. V., Stein, L. D., Mungall, C. J., & Holmes, I. H. (2009). JBrowse: a next-generation genome browser. *Genome research*, *19*(9), 1630-1638.

Skinner, M. E., & Holmes, I. H. (2010). Setting Up the JBrowse Genome Browser. *Current Protocols in Bioinformatics / Editoral Board, Andreas D. Baxevanis ... [et Al.]*, *0 9*, Unit–9.13.

Trapnell, C., Roberts, A., Goff, L., Pertea, G., Kim, D., Kelley, D. R., … Pachter, L. (2012). Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nature Protocols*, *7*(3), 562–578.

Trevino, V., Falciani, F., & Barrera-Saldaña, H. A. (2007). DNA Microarrays: a Powerful Genomic Tool for Biomedical and Clinical Research. *Molecular Medicine*, *13*(9-10), 527–541.

UCSC genome browser. https://genome.ucsc.edu

Vishwanathan, N., Yongky, A., Johnson, K. C., Fu, H. Y., Jacob, N. M., Le, H., ... & Hu, W. S. (2015). Global insights into the Chinese hamster and CHO cell transcriptomes. *Biotechnology and bioengineering*, *112*(5), 965-976.

Walsh, G. (2006). Biopharmaceutical benchmarks. *Nature biotechnology*, *24*(7), 769-776.

Walsh, G. (2014). Biopharmaceutical benchmarks 2014. *Nature biotechnology*, *32*(10), 992-1000.

Wang, J., Kong, L., Gao, G., & Luo, J. (2013). A brief introduction to web-based genome browsers. *Briefings in Bioinformatics*, *14*(2), 131-143.

Wang, Z., Gerstein, M., & Snyder, M. (2009). RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews. Genetics*, *10*(1), 57–63.

Westesson, O., Skinner, M., & Holmes, I. (2013). Visualizing next-generation sequencing data with JBrowse. *Briefings in Bioinformatics*, *14*(2), 172–177.

WormBase. http://www.wormbase.org/#012-34-5

Wurm FM, Hacker D. (2011). First CHO genome. *Nature biotechnology*, *29*(8), 718-720.

Xu, X., Nagarajan, H., Lewis, N. E., Pan, S., Cai, Z., Liu, X., ... & Andersen, M. R. (2011). The genomic sequence of the Chinese hamster ovary (CHO)-K1 cell line. *Nature biotechnology*, *29*(8), 735-741.

Yee, J. C., de Leon Gatti, M., Philp, R. J., Yap, M., & Hu, W. S. (2008). Genomic and proteomic exploration of CHO and hybridoma cells under sodium butyrate treatment. *Biotechnology and bioengineering*, *99*(5), 1186-1204.

Appendix A

FILE FORMAT

A.1 File format introduction

Here are the introductions of each format of files.

A.1.1 FASTA format file

The FASTA format is a text-based format file. In bioinformatics, these files are usually used to record either nucleotide codes or amino acid codes. Each sequence consists of two parts, a title line, and data lines. The title line is a description of the sequence which starts with '>' character. It often contains the label of the sequence and an optional comment. The data lines are the sequence of the DNA or protein [http://www.ncbi.nlm.nih.gov/BLAST/blastcgihelp.shtml, https://en.wikipedia.org/wiki/FASTA_format]. An example sequence in FASTA format is shown below:

```
>gi|12345|
ACTGCTGCAAGTCTACTACGCTGCGTAGACGTTAGCGTATAGCGGGCCTCT
ATCTGCGTAGCCAGATACTGCTGTCGATGTACGTAGAGCTGTCGTAGTAGT
CGTGCTCGTAGCTGACGT
```

A.1.2 FASTQ format file

The FASTQ format is a text-based format file. It normally uses four lines to save information for each sequence. The first line is the sequence identifier and optional comment (same as the FASTA title line) which starts with a '@' character. The second line is the sequence code. The third line starts with a '+' character and

contains an optional sequence identifier. The fourth line is the quality values of the sequence in the second line [http://maq.sourceforge.net/fastq.shtml, https://en.wikipedia.org/wiki/FASTQ_format]. An example sequence in FASTQ format is shown below:

@Sequence_id
CGTATCGACTGCATGCTGAC
+Sequence_id
>IB=I>#B)&6IB?!>>8*%

A.1.3 GTF/GFF format file

GFF (General Feature Format) format is a text-based format file. Each line consists of nine tab-separated columns, and each line contains information for one feature, plus optional track definition lines. The GTF (General Transfer Format) is identical to GFF version 2 [GFF. http://gmod.org/wiki/GFF].

The nine columns are sequence name, source, feature, start, end, score, strand, phase, and attribute. They are described as below [http://useast.ensembl.org/info/website/upload/gff.html]:

1. 'Sequence name' is the name/ID of the chromosome/scaffold where the sequence is located. E.g. NW_0012345.1.
2. 'Source' is the name of the program/algorithm that generates this file. E.g. ExpressGENiE.
3. 'Feature' is the feature type of the sequence. E.g. gene.
4. 'Start' is the start position of the sequence. E.g. 1234.
5. 'End' is the end position of the sequence. E.g. 2345.
6. 'Score' is a floating point value. E.g. 1.
7. 'Strand' is the direction of the sequence. It is one of '+', '-', or '.'.

8. ‘Phase’ is one of ‘0’, ‘1’, ‘2’, and ‘.’. The phase is a description for the ‘CDS’, which is used to indicate the position of the first base of the next codon in this region. The ‘0’ represents that the codon begins at the first base of the region, the ‘1’ represents the second base of the region, the ‘2’ represents the third base of the region, and the ‘.’ represent no phase.

9. ‘Attribute’ is a semicolon separated list of additional information for each sequence.

An example sequence in GTF/GFF format is shown below:

NW_0012345.1 ExpressGENiE gene 1234 2345 1 + . gene_id’abcd’;

A.1.4 GFF3 format file

GFF3 (Generic Feature Format Version 3) is an extension of GFF format. It keeps basically the same structure as GFF file, with nine columns per line, but contains more information [Generic Feature Format version 3. http://www.sequenceontology.org/gff3.shtml]:

1. Can contain features and subfeatures that are in different hierarchical levels.
2. Feature name/ID and the group membership have been separated.
3. Feature type field should be taken from a controlled vocabulary.
4. Each single feature can belong to several groups at one time.
5. Features that occupy disjunct regions have an explicit convention.

A.1.5 BedGraph and Bigwig format file

The bedGraph format makes it practical to view the results of next-generation sequencing experiments as tracks and allows display of continuous-valued data. Bigwig is a binary file that is derived from text-formatted wiggle plot (wig) or

bedGraph files. It has the same information as a bedWig file [BedGraph format. https://genome.ucsc.edu/goldenpath/help/bedgraph.html, Kent et al., 2010]. Here is an example for a bedGrpah file:

```
Sequence_id   start    stop     value
NW_12345.1    1234     2345     6666
```

A.1.6 BAM and SAM format file

A BAM (Binary Alignment/Map) file is the compressed binary version of a SAM (Sequence Alignment/Map) file that is used to represent aligned sequences. The SAM format consists of one header section and one alignment section. The header section starts with a '@' character. Each alignment line has eleven mandatory fields and variable numbers of optional fields [Li et al., 2009].

Table A.1.6 Eleven mandatory fields in the alignment line of BAM/SAM file

Number	**Name**	**Description**
1	*QNAME*	*Query NAME of the read or the read pair*
2	*FLAG*	*Bitwise FLAG (pairing, strand, mate strand, etc.)*
3	*RNAME*	*Reference sequence NAME*
4	*POS*	*1-Based leftmost POSition of clipped alignment*
5	*MAPQ*	*MAPping Quality (Phred-scaled)*
6	*CIGAR*	*Extended CIGAR string (operations: MIDNSHP)*
7	*MRNM*	*Mate Reference NaMe ('=' if same as RNAME)*
8	*MPOS*	*1-Based leftmost Mate POSition*
9	*ISIZE*	*Inferred Insert SIZE*
10	*SEQ*	*Query SEQuence on the same strand as the reference*
11	*QUAL*	*Query QUALity (ASCII-33=Phred base quality)*

Here is an example:

```
@header
00r     123     scaffold/chr    start   0       151M  *       0       0
        ACGTGTACGTGTTACTGACGT;  AS:i:0
```

A.1.7 Appendix A's references

BedGraph format. https://genome.ucsc.edu/goldenpath/help/bedgraph.html

FASTA. https://en.wikipedia.org/wiki/FASTA_format

FASTA. http://www.ncbi.nlm.nih.gov/BLAST/blastcgihelp.shtml

FASTQ. https://en.wikipedia.org/wiki/FASTQ_format

FASTQ. http://maq.sourceforge.net/fastq.shtml

GFF. http://gmod.org/wiki/GFF

GFF. http://useast.ensembl.org/info/website/upload/gff.html

Generic Feature Format version 3. http://www.sequenceontology.org/gff3.shtml

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., … 1000 Genome Project Data Processing Subgroup. (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, *25*(16), 2078–2079.

Kent, W. J., Zweig, A. S., Barber, G., Hinrichs, A. S., & Karolchik, D. (2010). BigWig and BigBed: enabling browsing of large distributed datasets. *Bioinformatics*, *26*(17), 2204–2207.

Appendix B

INTERNSHIP

B.1 Internship

The summer internship at Millipore Sigma was mainly working on gene editing, which is related with CRISPR-Cas 9. Compared with zinc-finger nuclease (ZFNs), and transcription activator-like effector nucleases (TALENs), currently, the CRISPR-Cas 9 is the most preferred powerful gene editing tool [Gaj et al., 2013]. Companies that supply technical support for gene editing work on providing competitive CRISPR-Cas 9 skill for users. Bioinformatics plays an important role in gene editing, especially for guide RNA prediction and selection. My summer intern project was using bioinformatics to select better guide RNA candidates from all predicted possible guide RNAs based on the hit position and hit isoform of the guide RNA. All of the functions were realized by custom Python (Python 2.7) scripts. The guide RNA selection enables users to obtain the guide RNAs that only satisfy their experimental requirements, which can significantly improve the experiment efficiency and make it less money-consuming.

B.2 Appendix B's references

Gaj, T., Gersbach, C. A., & Barbas, C. F. (2013). ZFN, TALEN and CRISPR/Cas-based methods for genome engineering. *Trends in Biotechnology*, *31*(7), 397–405.