INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES

By

Cheah Zhao Qin

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Perak Campus)

Jan 2018

REPORT STATUS DECLARATION FORM

Title: _____

_____

_____

Academic Session: _____

I _____

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

_____ _____

(Author's signature) (Supervisor's signature)

Address:

_____

_____ _____

_____ Supervisor's name

Date: _____ **Date**: _____

.

INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES

By

Cheah Zhao Qin

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Perak Campus)

Jan 2018

DECLARATION OF ORIGINALITY

I declare that this report entitled “Interactive Online Tool For Methylation Studies” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : Cheah Zhao Qin

Date : 16/04/2018

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Dr. Ng Yen Kaow who has given me an opportunity to engage in bioinformatics. He gave a lot of helps to me throughout the project. Without his guidance, the project will not completed smoothly.

Thanks to ZiCheng, Zhao and HuiMin, Chai from City University of Hong Kong who have been always provided details of the analysis to me. Finally, I must say thanks to my parents and my family for their love, support and continuous encouragement throughout the course.

ABSTRACT

DNA methylation acts as a vital role in cancer detection. Lack of visualization tools wastes researchers' time when they are doing their projects and conducting research. They need a tool that is able to help them to analyze and visualize their data. None of the current visualization tool provides complete analysis and visualization for a bisulfite sequencing data. Thus, the project aims to develop a visualization tool for methylation. This project will save their time by generating publishable graphs. The objective of this project is to visualize overview of DNA methylation and analyze the quality of the input data. The visualization tools are written with the handling of large amount of data in mind since that is where they are most needed. The input file will be accepted in bgzip or gzip format. They will accept standard tables generated by BSMAP, or any input file with enough chromosome details. The tools are developed in the current standard practice in web development platforms: TypeScript, python, and d3.js.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

DMRs	Differential methylated regions
WGBS	Whole genome bisulfite sequencing
PCA	Principal Component analysis
API	Application Programming Interface

CHAPTER 1

INTRODUCTION

SECTION 1.1- PROBLEM STATEMENT AND MOTIVATION

DNA methylation is a process of adding methyl groups to DNA molecule and form 5-methlcytosine (5mC). DNA methylation has been broadly studied for its character which changes the DNA activity without changing the sequence. DNA methylation acts as a 'hat' to suppress DNA. DNA methylation is an important epigenetic mark that plays a vital role in genomic imprinting, X-chromosome inactivation, embryonic development, suppression of transposable components, aging, carcinogenesis and other biological process. These characteristic modifications have been linked to cancer and several chronic diseases. The increase in projects on DNA methylation has led to an increase in available genomic and epigenetic data.

However, lack of available tools to visualize huge genomic data and display interesting interfaces slows down the researcher's work and degrade the presentation of the researcher. Limitations that exist in currently available tools to visualize the outcome also degrade the presentation of the researcher. Researcher cannot make an interesting presentation with their results and discoveries to help others better understand their work. Besides, it is time-consuming for a researcher to interpret and visualize the data without the aid of tools. There is increasing data obtained in this field but no suitable visualization tool exists to help researchers visualize the results for public viewing. With that, researchers are having difficulty in explaining their results and discoveries to the authorities and the public that might be interested in this matter. The results and discoveries of their research will not be widely spread. The quality of the researcher's job might also be affected. Researchers need to waste more time to analyse the sequence of DNA methylation. They also need a tool that can be used to display and analyse their information that can also be directly used in their papers with publishable quality. In short, it is important to take the problem into consideration and develop a solution to solve it.

The project aims to develop an interactive online tool that helps in DNA methylation studies. The tool aims to provide meaningful information and interface

for the researchers. Static figures in DNA methylation results make them difficult to explain. Moreover, they can use the tool to generate interactive graph and chart for their research. Static graph or chart that cannot interact with user is difficult to show the details of the graph clearly. The interactive graph or chart captures details of the DNA methylation research results and shows in interesting way so that they can use the graph produced to give a clear elaboration for their research and use for publication.

SECTION 1.2- BACKGROUND INFORMATION

DNA methylation is an epigenetic system that transfers methyl to a specific base in cytosine. The process is carried out by DNA Methyltransferases (DNMT). DNMT1 maintains methylation and controls cell division. DNA methylation status has a strong inverse correlation with gene expression. DNA methylation normally happened at outside promoter region. Promoter region contain gene expression that helps in transcription of gene. Once methylation occur in promoter region, the merging of transcription factor with promoter will be damaged. It affect gene transcription of the cell. Some of the silencing of gene transcription may cause cancer. DNA methylation pattern changes in cancer cell. In normal cells, there will be an absence of methylated cytosine in the promoter region. While in the cancer cells, the cytosine in promoter region is methylated and results in no transcription of gene. Some of the transcription helps to repair mutation of the cell. Due to transcription of gene silencing in tumour gene promoter, mutation in cancer cell increased.

CpG site or CpG Island is one of the important concept that is going to be illustrated in the project. The CpG sites are regions of DNA where a cytosine nucleotide is connected to a guanine nucleotide like Figure 1.2.1. Many CpG sites form a CpG island. CpG islands arise near promoter region of the gene. CpG island methylation will result in control of imprinted gene and X-chromosome inactivation. Besides, methylation of CpG is important in control of gene expression.

Figure 1.2.1 CpG site/ CG site

CHAPTER 1: INTRODUCTION

Detection of Differentially Methylated Regions is one of the main figure that is going to visualize in the project. Differentially methylated regions (DMRs) are genomic regions with various methylation status among different samples (tissues, cells, individuals or others). These regions worked as functional region which is regulation of gene expression (Zhang, Y et. al, 2011, e58). DMRs show abnormal methylation status in cancer compare to normal cell.

There are various methods used to detect genome-wide DNA methylation. Whole genome bisulfite sequencing (WGBS) used to determine DNA methylation status in single cytosine. It is more powerful compared to others but at the same time associated with high cost. Table 1.2.1 compare the methods that are used to detect DNA methylation.

Table 1.2.1 various method used in detect genome wide DNA methylation.

	WGBS	450K infinium bead chip array	MethylationEPIC infinium bead chip array	ERRBS
Regions sequenced	Whole genome including intergenic and enhancer regions	Predesigned array-based	Predesigned array-based	Determined by Msp1 digestion to enrich for CpG fragments
Genome coverage	15-20 million CpG sites	485 000 methylation sites across the genome	850 000 methylation sites across the genome	3 million CpG sites approximately 85% of CpG islands, and 60% promoters
Assay details	Bisulfite conversion of genomic DNA, followed by next-generation sequencing	Bisulfite conversion of genomic DNA followed by annealing bead array	Bisulfite conversion of the genomic DNA, followed by annealing bead array	MspI digestion followed by bisulfite conversion and next-generation sequencing
Cost per sample	$$$	$	$	$$
Input DNA	50-100 ng	500 ng-1 μg	250 ng	10-300 ng
Additional coverage information	Comprehensively covers the entire genome including methylated and unmethylated regions	Most CpG islands, shores, flanking regions, non-CpG island methylation, and miRNA promoter regions	5'hydroxy-methyl-cytosine patterns and novel CpG regulatory sites	Enrichment in CpG islands, CpG shores, promoters, exons, introns, and intergenic regions

Abbreviations: ERRBS, enhanced reduced representation bisulphite sequencing; WGBS, whole-genome bisulfite sequencing.

Figure 1.2.2 shows the step in bisulfite library preparation. Fragmentation of DNA cut genomics DNA into many fragments. Some of the fragments might face difficulties to undergo treatment due to different lengths of fragments. Thus, end repair adds adaption ligation to the fragment before bisulfite conversion starts. All the unmethylated cytosine will be converted to thymine while methylated cytosine will remain as cytosine. Repair of DNA fragment can be identified. The adaption ligation added will not consider as effective cytosine in DNA methylation.

CHAPTER 1: INTRODUCTION

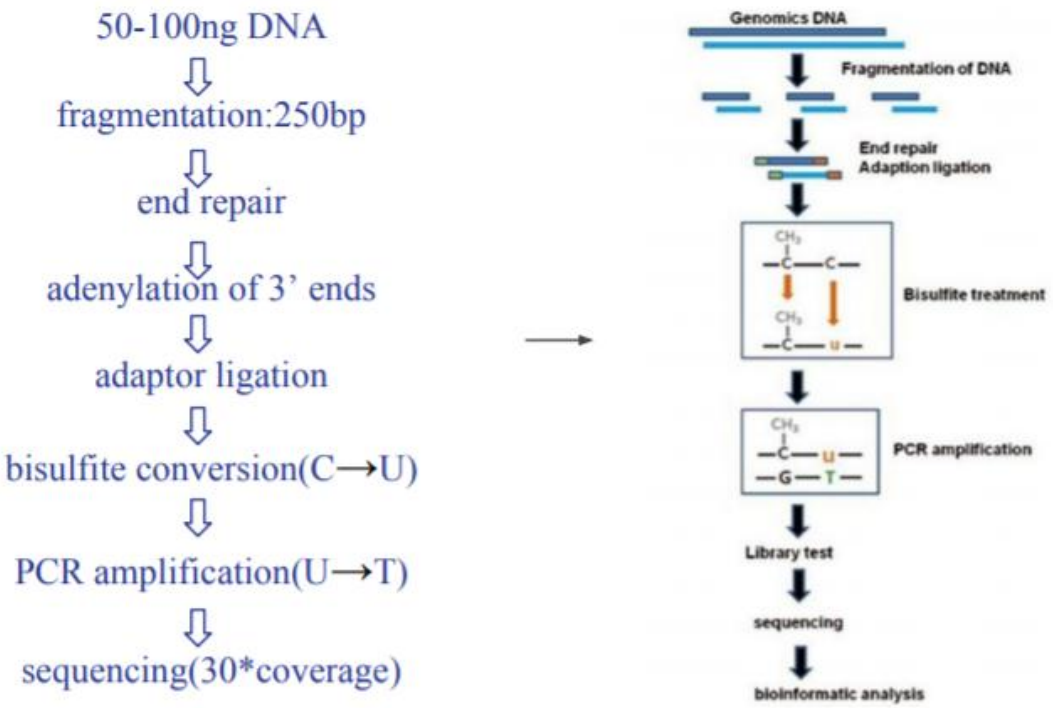


Figure 1.2.2 WGBS steps in bisulfite library preparation

In addition, tabix is one of the tool that will be used to retrieve genomic data. Tabix indexed the bgzip files in tab separated format for example GFF, BED, SAM and VCF. Tabix allows fast data retrieving by query it with the format of "chr1:begin position-end position". Moreover, tabix is a powerful tool that retrieve data from a compressed genomic file. VCF is the genomic file format that will be using for the input as Figure 1.5.3. It contains 8 fix columns that included the information for the chromosome at certain position. It is similar to the input file that is going to be used.

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	NA00001	NA00002	NA00003
20	14370	rs6054257	G	A	29	PASS	NS=3;DP=14;AF=0.5;DB;H2	GT:GQ:DP:HQ	0|0:48:1:51,51	1|0:48:8:51,51	1/1:43:5:.
20	17330	.	T	A	3	q10	NS=3;DP=11;AF=0.017	GT:GQ:DP:HQ	0|0:49:3:58,50	0|1:3:5:65,3	0/0:41:3
20	1110696	rs6040355	A	G,T	67	PASS	NS=2;DP=10;AF=0.333,0.667;AA=T;DB	GT:GQ:DP:HQ	1|2:21:6:23,27	2|1:2:0:18,2	2/2:35:4
20	1230237	.	T	.	47	PASS	NS=3;DP=13;AA=T	GT:GQ:DP:HQ	0|0:54:7:56,60	0|0:48:4:51,51	0/0:61:2
20	1234567	microsat1	GTC	G,GTCT	50	PASS	NS=3;DP=9;AA=G	GT:GQ:DP	0/1:35:4	0/2:17:2	1/1:40:3

Figure 1.2.3 VCF file format

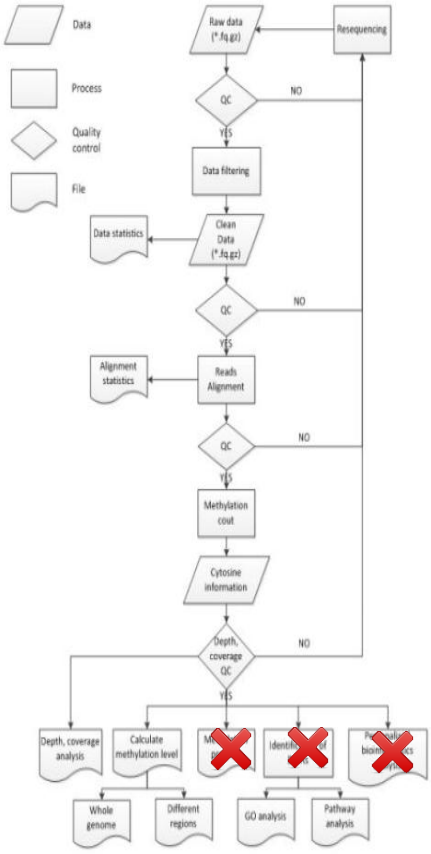
SECTION 1.3-OBJECTIVE

The main objective of this project is to develop a tool that helps researchers to visualize the result of methylation studies. The tool developed aims to help researchers and users alike to analyze their research results and generate figures with publishable quality. Most of the DNA methylation analysis and results are expected to be visualized by using the tool.

Moreover, the sub-objective of the project is to develop an interactive visualization tool that accept a large data genomic input from user. The genomic input that normally needed to visualize the data may be larger than 10GB. Thus, it is important that it can process a large genomic data.

Besides, the visualization tool is expected to display the analysis result of DNA methylation based on user input. The input will be analysed and the quality of the input will be shown. The visualization tool is expected to have an interaction with the user. Extra information of the figures need to be delivered in an interesting method. The project will be focused on develop an interactive DNA methylation profile that allowed user to view the details by focusing and dragging the diagram.

CHAPTER 1: INTRODUCTION



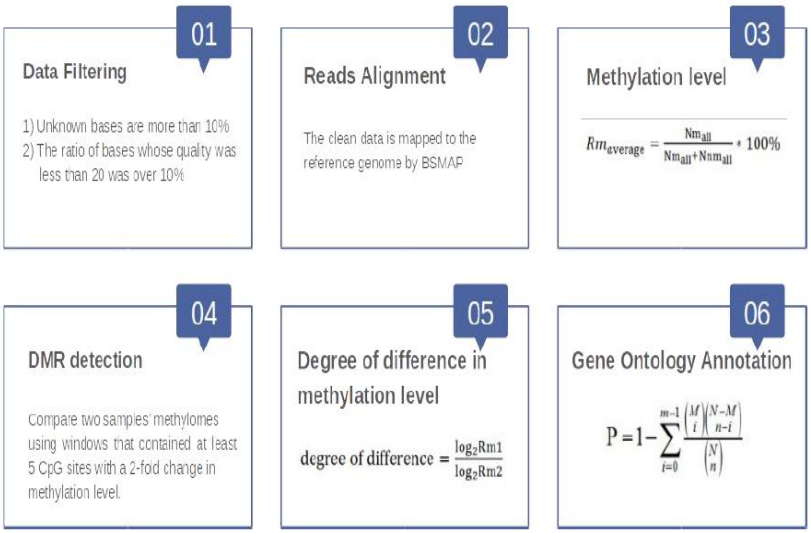


Figure 1.3.1 The cross section area will not be covered in the project. The input need to be processed before upload. The green square highlighted parts that are not going to be focused in the project. The yellow circle highlighted the parts that are going to be visualized in the project.

In this project, bisulphite sequencing mapping is not covered. The input provided should be in BSMAP table format. Raw sequencing input need to be processed through BSMAP before this. The project will only focus on visualization for DNA methylation. Building of the platform will not be covered in this project.

SECTION 1.4- PROPOSED APPROACH AND ACHIVEMENT

Researchers need a visualization tool that can completely visualize the analysis results of DNA methylation. The proposed interactive visualization tool will solve the researcher's problem. An interactive online tool that visualize the methylation studies will be developed at the end of this project. The tool aims to help users control the quality of the input, display the depth of methylated Cytosine and show overview of DNA methylation.

The project intends to handle large genomic data, analyse and visualize the analysis result. Table 1.4.1 shows ratio of cytosine covered at 2x. Distribution of the coverage depth of cytosine shows the overall effective cytosine in the sample. It determines the quality of input. Figure 1.4.1 shows the distribution of the coverage depth of cytosine. Blue line in the graph represent the frequency of the cytosine at the particular effective cytosine count. For example, the total effective cytosine in the table is 10 but the effective cytosine with count 1 is 2. The graph will show that frequency at count=1 is 0.2. The green line represent the accumulative percentage of the effective cytosine count. Figure 1.4.2 shows distribution of methylation level in mC, mCHH, mCHG. At methylation ratio equals to 0.25, fraction of total mC will be 0.4 if the table has four 0.25 methylation ratio and six 1.00 methylation ratio for mCG.

Table 1.4.1 Percentage of cytosine covered by at least 2 read in the content

Sample	Covered cytosines(%)			
	C	CG	CHG	CHH
P-ZWf	90.47	91.63	92.43	89.64
F1-ZWf	89.32	90.68	91.50	88.38
P-ZWm	90.15	91.30	92.14	89.32
F1-ZWm	89.51	90.64	91.57	88.65
ZZm	90.23	91.41	92.26	89.37

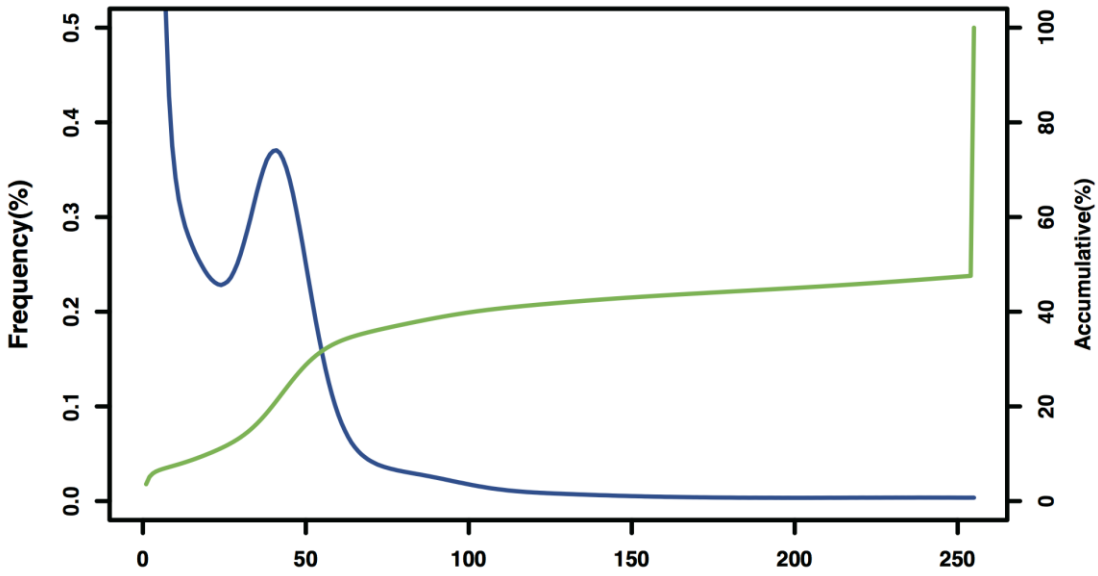


Figure 1.4.1 Distribution of the coverage depth of cytosine

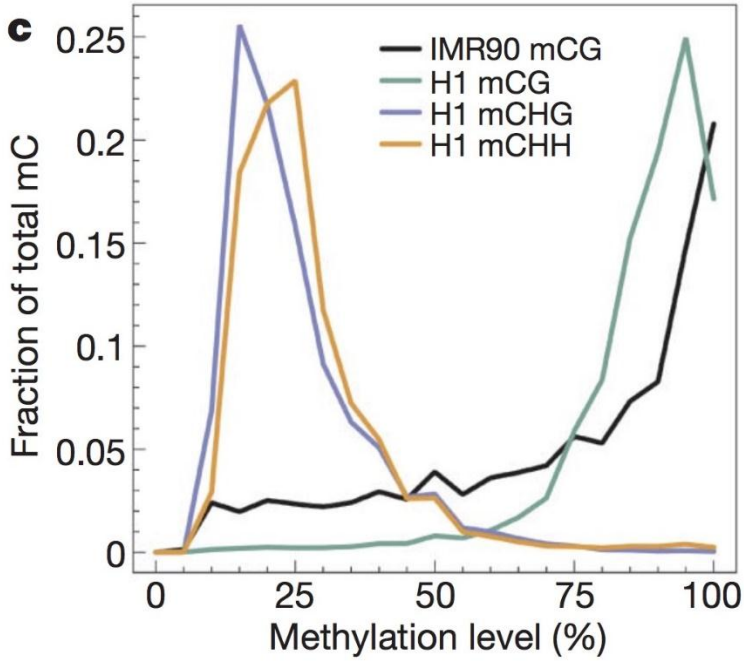


Figure 1.4.2 Distribution of methylation level in mC, mCHH, mCHG (Lister, et.al., 2009)

Moreover, the visualization tool developed will provide analysis results for researcher. There are many analysis results that need to be included in the overview of the DNA methylation. Pie chart and bar chart are drawn to visualize the percentage of methylation level in a sample. It helps researcher to understand which part of the

sample is going wrong by looking at the hypomethylated or hypermethylated part of the sample.

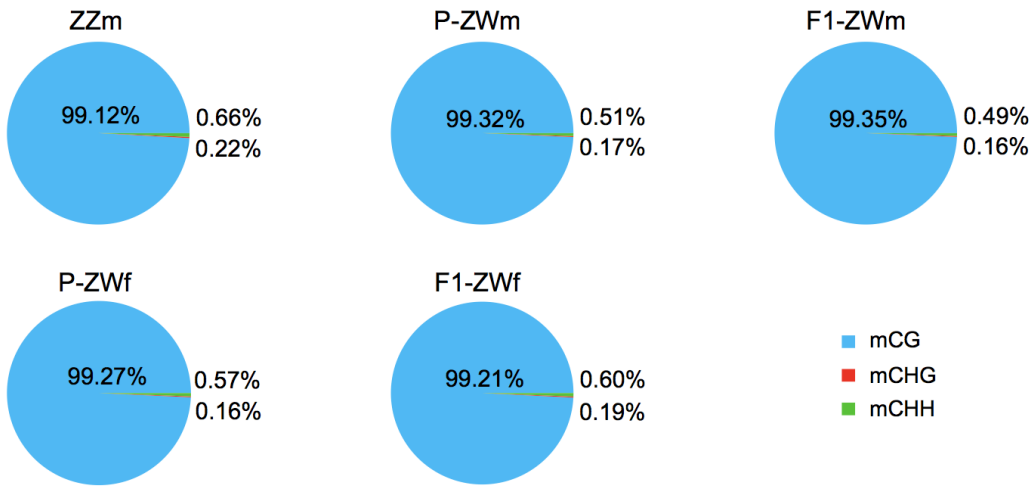


Figure 1.4.3 Pie chart that visualized percentage of CG,CHG and CHH

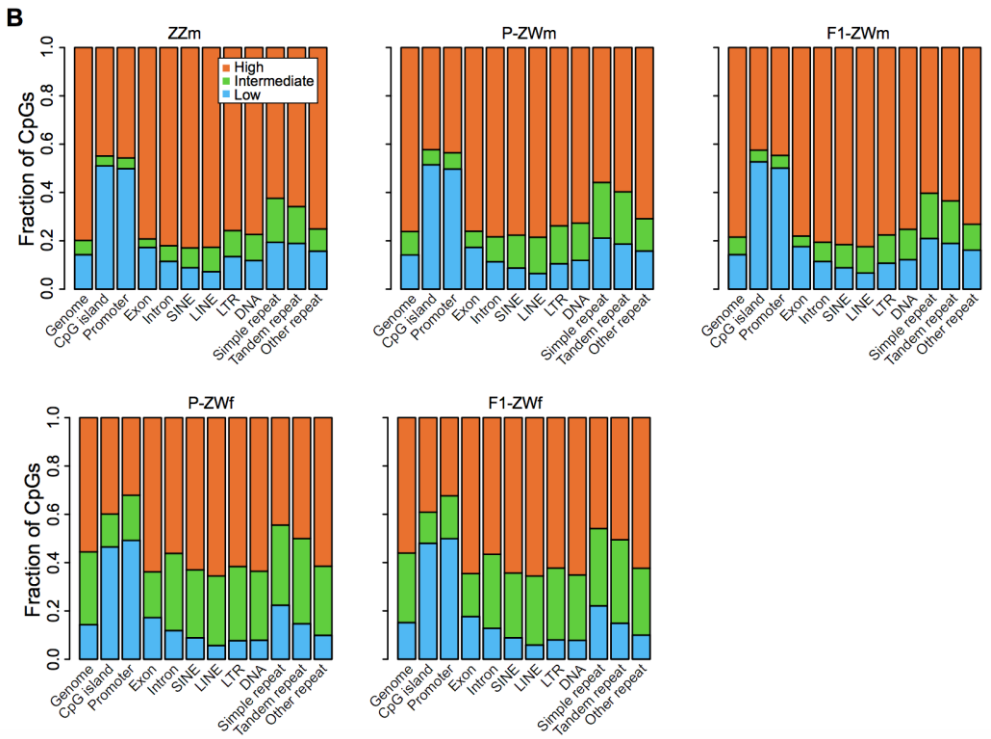


Figure 1.4.4 Bar chart displayed fraction of CpG in which is in low, intermediate and high methylation ratio from different regions.

Figure 1.4.7 shows clustering and PCA analysis of CGs across samples. PCA analysis and heatmap for the sample will be visualized in the project. Heatmap is frequently used in the visualization of gene expression.

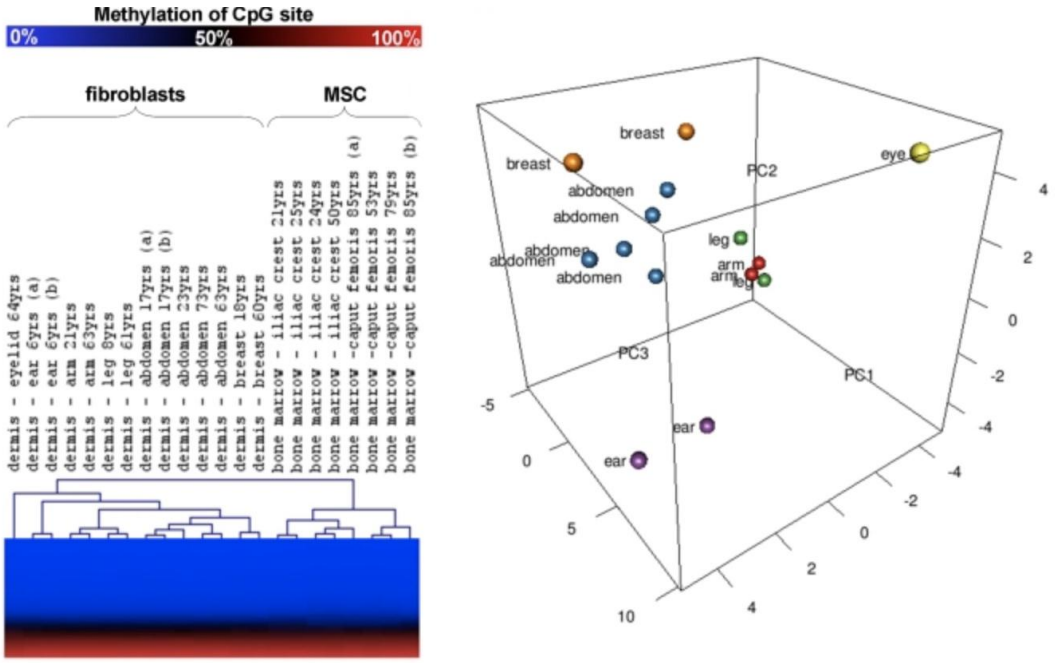


Figure 1.4.5 Clustering and PCA analysis of CGs across samples.

An interactive online tool that helps in methylation studies is being developed as a product of the project. Most of the graphs contain at least one interaction such as “hover” to display detailed information, or “drag” in order to have a clearer view.

SECTION 1.5- IMPACT, SIGNIFICANCE AND CONTRIBUTION

The contribution of the project is a tool that is used to visualize the analysis results of DNA methylation. The tool is developed to display complete DNA methylation analysis results. Projects concerning human DNA are becoming more and more popular. This brings huge impact in the increase of individual genomes. Visualization becomes a big problem for researchers. They need to waste their times in analysis and visualization of their results. The tool is important to researchers to do their analysis and get an interactive visualization of their research results.

The tools allows researchers to get better visualization of the overview of DNA methylation. They also inform the researchers of their data’s quality so they can justify the accuracy of the analysis result. The visualization tool aims to provide better and more interesting interfaces for DNA methylation profiles.

The tools also provide a mean for the researchers who are currently doing research on DNA methylation to generate figures with publishable quality. Researchers can use the tool to display complete DNA methylation results and identify the role or function of a sequence that exists in the functional databases or published biology databases. Researchers may need to identify the function of a sequence that is highly methylated. By using the visualization tool, they can save their time to find out the role and function of the sequence. Researchers will use the tool to retrieve analysis results easily.

The interactive tool developed will be included as one of the visualization modules in the website http://www.dovirus.com/. The tables and visualizations of data will be part of the analysis that can be used directly in publication purpose.

SECTION 1.6- REPORT ORGANIZATION

The report will be organized as stated below. The report consists of 5 chapters, namely introduction, literature review, system design, result achieved, analysis of graph generated, and conclusion.

Chapter 2 will discuss the proposed solution to envision analysis for methylation studies from researchers and developer. Chapter 3 will discuss design specification of the project. Chapter 4 will discuss the result of the project. All development of the project from input to visualization will be included in chapter 4. Conclusion will be the last part of the report. The report details will be summarized in conclusion.

CHAPTER 2:

LITERATURE REVIEW

SECTION 2.1- EXISTING SOLUTIONS OVERVIEW

DNA methylation modifies the action of DNA segment without moving the sequences. DNA methylation is important as it highlights key biomarkers to identify some of the chronic diseases. A lot of research is done on DNA methylation and it shows that some of the biological process like aging and gene silencing have resulted in gene mutation and finally causing cancer. These results and recoveries need to be visualized by using a good visualization and analysing tool. A complete genomic methylation result and analysis should include methylation level of chromosome, CpG sites in the differentially region (DMR), comparison between different chromosome, methylation profiles and some others methylation related results and analysis.

Some visualization tools are developed to perform analysis for DNA methylation research and display the results and figures. QUMA is a quantification tool for DNA methylation analysis. It speeds up the study of bisulfite sequencing data and displays the result. It also allows the researcher who isn't familiar with the analysis of bisulfite sequencing to perform the analysis by using QUMA (Kumaki, et al., 2008, p.W171). The next visualization tool is MethylViewer. It is developed from CpGviewer and is used for MAP-IT (MAP individual templates) and MAP (methyltransferase accessibility protocol) foot printing tasks to produce more complete statistics with an interactive map displaying methylated sites and others (Carr, et al., 2011, p.e5). Methylation plotter is a dynamic visualization web tool of DNA methylation that accepts up to 100 CpG samples as input and produce graphic representation of the results (Mallona, et. al, 2014, p.11). Methylome DB browser is a visualization tool that shows DNA methylation profiles (Xin, et. al 2012). It is an interactive browser that allows user to move the gene's position and shows the methylation pattern of the gene. However, it does not support scroll to enlarge in the browser. CpGviewer is a simple visualization tool that automates the procedure of studying and aligning the DNA sequences of duplicated PCR products derived from bisulphite-treated mammalian DNA (Carr, et. al, 2007, p. e79). Despite that,

CpGviewer does not completely display or analyse the methylation analysis results and only show the summary statistics.

Most of the current existing visualization tools do not include complete analysis. Researchers need to spend more time to analyse what functions the sequence represents like in Figure 2.1.1, which shows the methylation patterns in normal cells and cancer cells. Cancer cells in Figure 2.1.1(B) is hypermethylated compared to normal cells. Lack of available tools that analyse the known sequence and link it to the functional databases adds unnecessary trouble to researchers as well as the readers of their research publications. Researcher have to identify and search for the functional databases in order to know the role or function of the sequence. Hence, we will develop a visualization tool for DNA methylation with the function that links the known sequence and functional database or published biology data. The tool will include complete DNA methylation analysis results and some additional features to perform new analysis and display the figures.

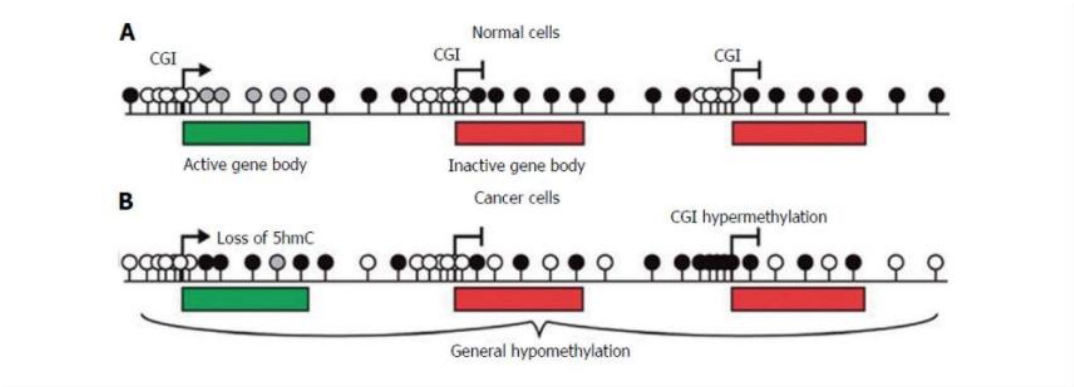


Figure 2.1.1 the methylation patterns of normal and cancer cells. (A) The amount of CpG in mammalian genome is depleted and most of the CpG sites are methylated (black lollipops). CGIs are normally unmethylated (white lollipops). They are rich in CpGs and occur with gene promoter, regardless of gene expression status. The bodies of active genes are enhanced in hydroxymethylated CpGs (grey lollipops). (B) In cancer cells, both DNA methylation and hydroxymethylation are decreases in cancer genomes yet certain CGIs turn out to be abnormally hypermethylated (Sproul et al, 2013).

SECTION 2.2- QUMA

QUMA is developed to visualize the analysis result of methylation research. QUMA is developed to undergo bisulphite sequencing analysis for CpG methylation. QUMA accepts FASTA, GenBank and plain sequence in the target genomic sequence file as input. FASTA represents either nucleotide sequences or peptide sequences in a text-based format ("FASTA", Wikipedia: The Free Encyclopedia). Amino acids in the

sequences is represent using single letter. GenBank is an open sequence database that contains all publicly available DNA sequences and their protein translation.

```
;LCBO - Prolactin precursor - Bovine
; a sample sequence in FASTA format
MDSKGSSQKGSRLLLLLVVSNLLLCQGVVSTPVCPNGPGNCQVSLRDLFDRAVMVSHYIHDLSS
EMFNEFDKRYAQGKGFITMALNSCHTSSLPTPEDKEQAQQTHHEVLMSLILGLLRSWNDPLYHL
VTEVRGMKGAPDAILSRAIEIEEENKRLLEGMEMIFGQVIPGAKETEPYPVWSGLPSLQTKDED
ARYSAFYNLLHCLRRDSSKIDTYLKLLNCRIIYNNNC*

>MCHU - Calmodulin - Human, rabbit, bovine, rat, and chicken
ADQLTEEQIAEFKEAFSLFDKDGDGTITTKELGTVMRSLGQNPTEAELQDMINEVDADGNGTID
FPEFLTMMARKMKDTDSEEEIREAFRVFDKDGNGYISAAELRHVMTNLGEKLTDEEVDEMIREA
DIDGDGQVNYEEFVQMMTAK*

>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV

EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG

LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL

GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX

IENY
```

Figure 2.2.1 FASTA sequence format

Bisulphite alignment, sequence trimming, exclusion of critical sequences and methylation status analysis will be implemented to the input in QUMA. All the data displayed in the web pages can be downloaded in standard file format. QUMA provides almost all of the data processing for analysis of bisulphite sequence. It also provides quality control for the input. QUMA perform analysis and generate result in a very short time. It helps researcher to visualize their research result and perform analysis to get analyzed graphics and statistical results. The figures and tables that generated can be customized. The figure below shows one of the output of the analysis. However, it does not provide detection of DMRs in the tool.

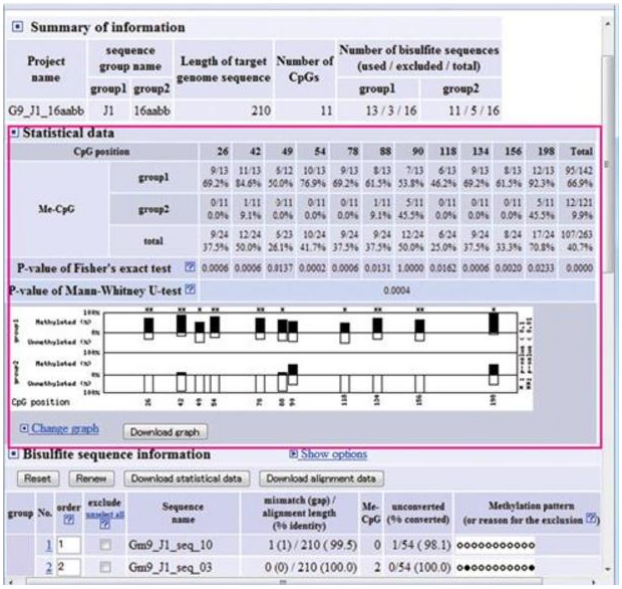


Figure 2.2.2 One of the outputs for QUMA: display the statistical result of the sequences.

SECTION 2.3- MethylViewer

Methylviewer is an advanced CpGviewer that handles MAP (methyltransferase accessibility protocol) and MAP-IT (MAP individual templates) foot printing projects. Methylviewer accepts alignments that are created by itself or imported in FASTA sequence format. It outputs more detailed statistics and interactive maps that show methylation sites and unconverted residues outside methylation sites.

However just like CpGviewer, MethylViewer required user to download before use. The alignment imported can be in FASTA sequence alignment only.

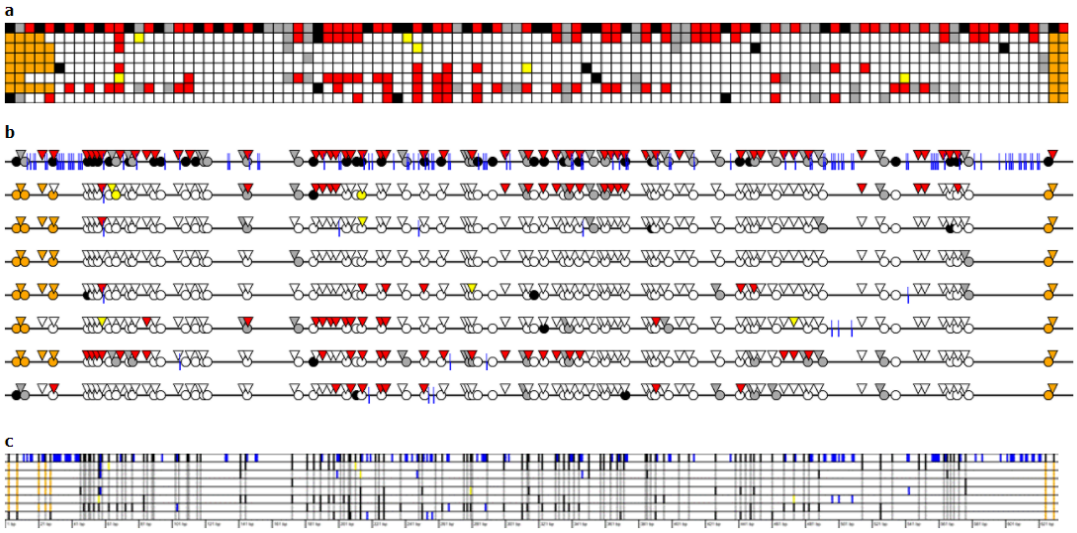


Figure 2.3.1 outputs of MethylViewer. A) The interactive plot. Each square represents a methylation site and its methylation status. B) Scaled "lollipop" image that is used for publication purpose C) dC conversion map show unconverted cytosine residues.

SECTION 2.4- Methylation plotter

Lastly, Methylation plotter is the tool that provides statistical summaries for methylated data. Methylation plotter is developed by shiny, an R framework. It takes a tab-separated file that containing the status of up to 100 CpG in up to 100 different samples in beta values format as input. Outputs of methylation plotter are shown in Figure 2.4.2 and 2.4.3.

The application shows an interactive output that summarizes the status of each CpG site and for every model in "lollipop" or grid styles as results. Different from other existing solution, Methylation plotter perform the subsequent analysis that need to be performed on the beta values that is generated from bisulfite-converted

electropherograms. It provides fast and easy generated custom plot. However, it does not include a complete methylation analysis results.

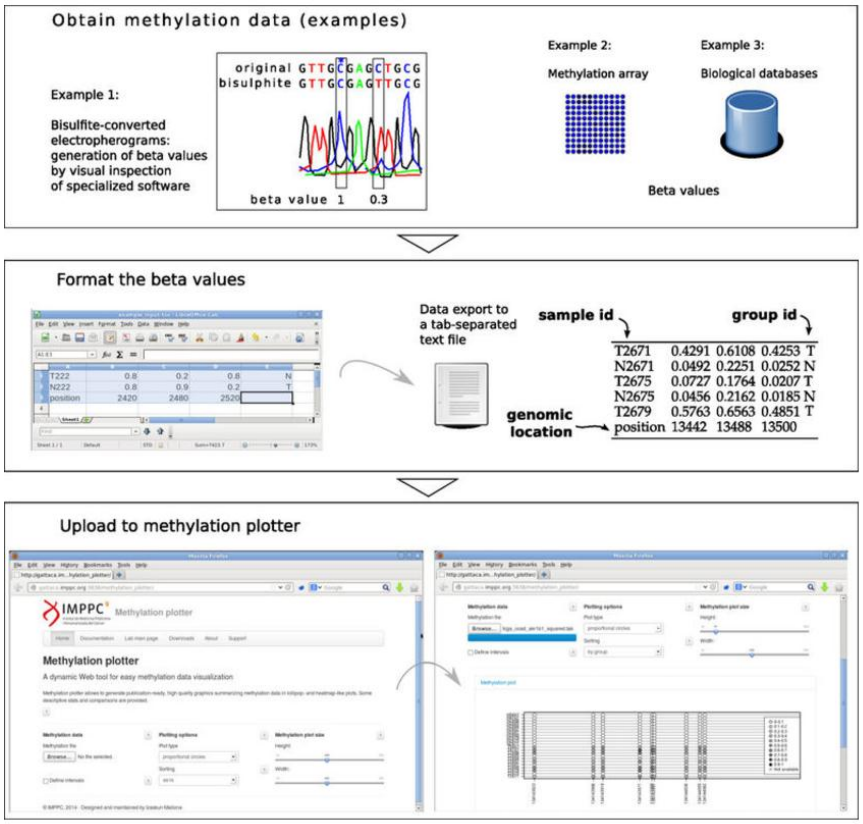


Figure 2.4.1 Data flow of methylation plotter. From the figure above, it shows that the beta values needs to be converted to tab-separate text file before upload to methylation plotter.

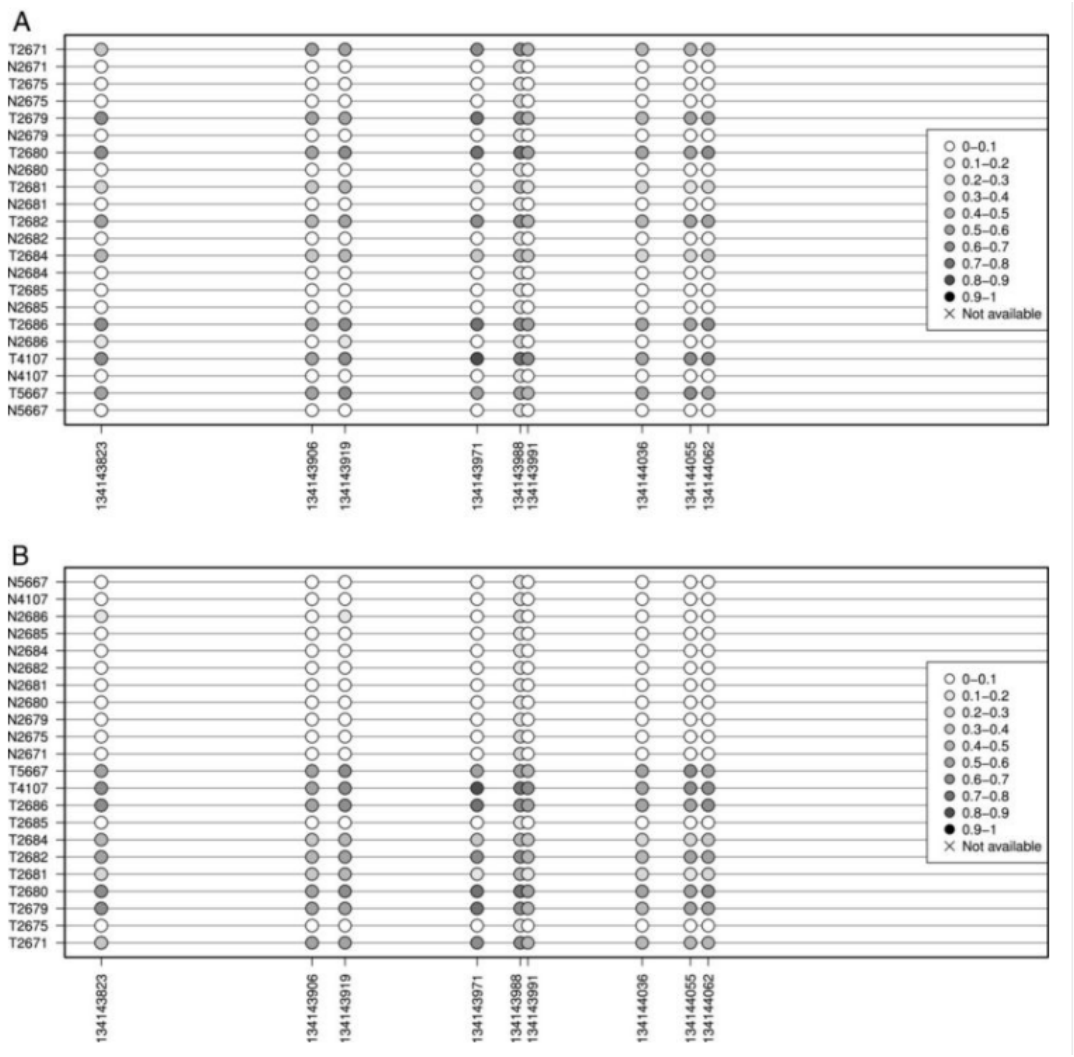


Figure 2.4.2 Output 1 of the Methylation plotter (lollipop look). A) Normal and tumor tissue data are alternated by the input data. B) Data visualization once the samples are explicitly organized based on the tissue type; the pattern of tumor hypermethylation can be spotted easily.

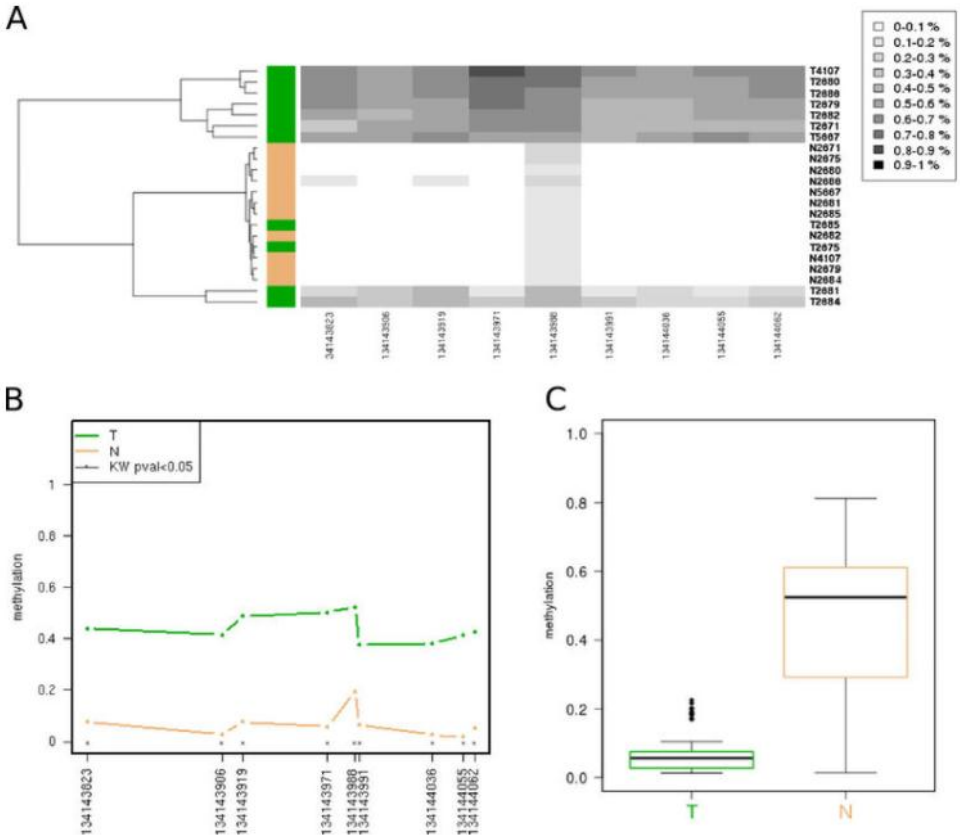


Figure 2.4.3 Output 2 of methylation plotter. A) Unverified hierarchical bundling of the data; sample label colours show the user-provided classification. B) Methylation profiling plot C) boxplots for each set by displaying the methylation data distribution

SECTION 2.5- MethylomeDB

Methylome Database is the database that includes DNA methylation profiles of the brain. It uses UCSC genome browser mirror sites to visualize DNA methylation profiles of the gene. It can be searched by genomic region, gene name and other markers. It is a powerful tool that shows methylation profile by accepting various types of input. However, the methylation profiles of the gene cannot be zoomed in by scrolling. It only displays information when user clicks on it.

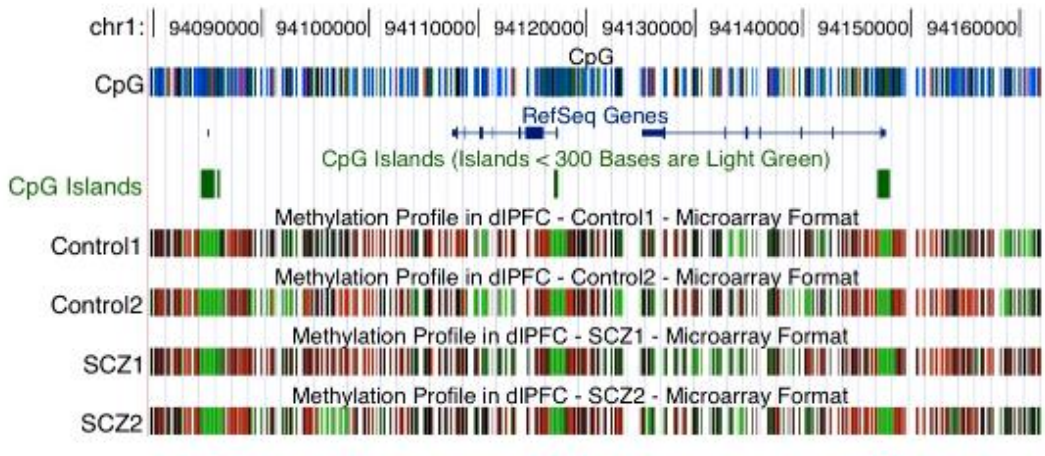


Figure 2.5.1 Search by genomic features in methylomeDB browser.

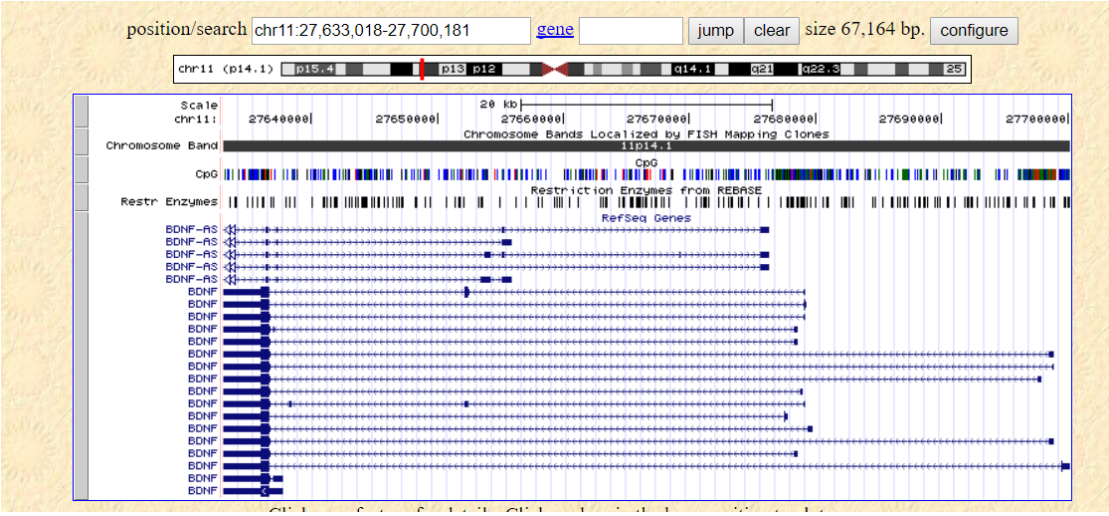


Figure 2.5.2 Methylation profile of gene at specific position.

SECTION 2.6- CpGviewer

Besides, CpGviewer is developed to handle bisulphite sequencing projects. It is used to produce bisulphite-treated templates. CpGviewer accepts plain text sequences or a variety of electropherogram formats as input. CpGviewer aims to identify the methylation status of CpG dinucleotide. The methylation status of CpG dinucleotide is displayed in Figure 2.6.1. The figure is displayed in an interactive view. The detail will be displayed by left click on the square and underlying sequence alignment can be reviewed by right-clicking a square. All the squares in the figures are editable. User can manually edit the methylation status of any of the figure once the programme miscalled a CpG dinucleotide. The output can be saved in text file or image file.

However, CpGviewer does not perform quality check. It only performs sequence alignment and displays in an interactive platform. It also requires user to download the tool to perform visualization of the sequence.

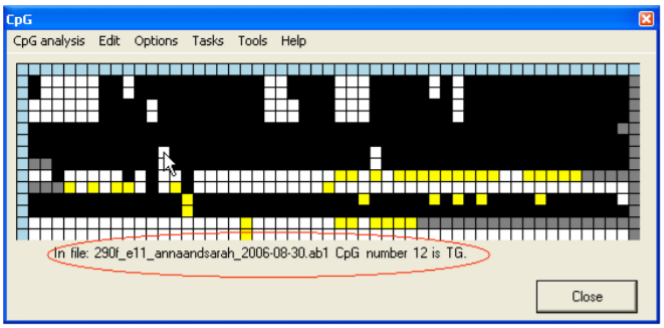


Figure 2.6.1 CpG dinucleotide sequence. The colour in the figure indicated the methylation status. Black colour is methylated, pink and grey are for unknown status and others colours represent unmethylated. The detailed info of nucleotide will be shown by left clicking the square.

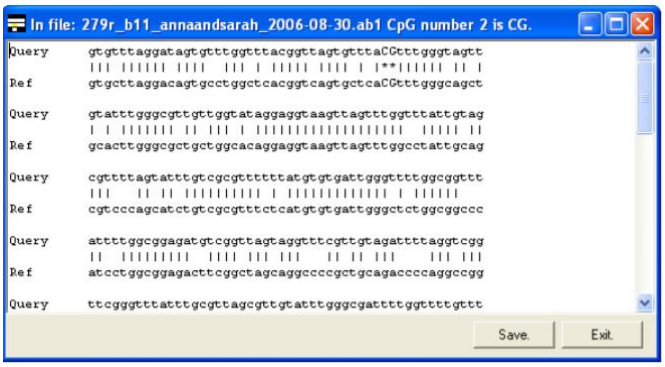


Figure 2.6.2 the underlying sequence that displayed through right clicking the square.

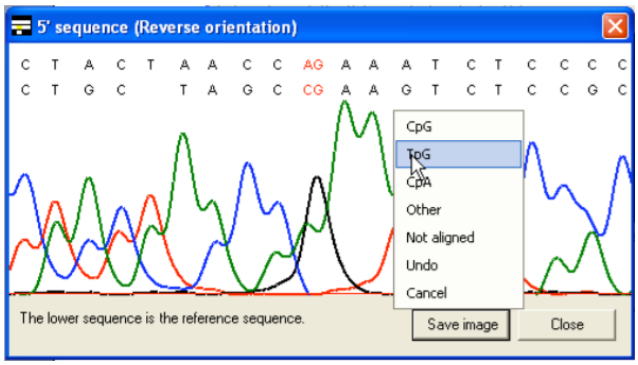


Figure 2.6.3 Sequence alignment of the square In Figure 2.6.2.

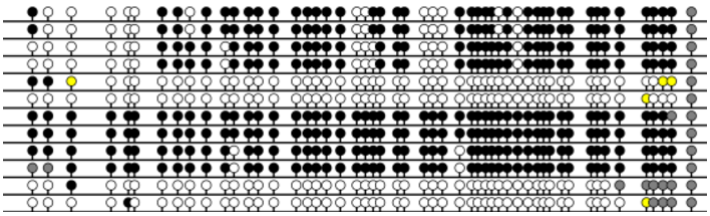


Figure 2.6.4 sequence that show in "lollipop" style which is normally used in publication.

CHAPTER 3:

PROPOSED METHOD/APPROACH

SECTION 3.1- DESIGN SPECIFICATIONS

A tool for visualizing DNA methylation is developed in this project. The visualization of DNA methylation can be divided into 2 parts, respectively quality control and overview of DNA methylation. The user is expected to upload the input for analysis. After the user uploaded the table generated by BSMAP as input, quality control table and graphs will determine the quality of the input. Percentage of methylated-cytosine will be visualized in overview of DNA methylation. Clustering and PCA analysis are performed to classify the sample. Visualization of DNA methylation accepts two type of input, bgzip file which is in VCF format or any gzip file that contains needed information for the graph.

First of all, user is required to upload the results of BSMAP for the samples. There will be many chromosomes in one sample file. BSMAP is a software that perform effective bisulfite sequencing reads mapping in DNA methylation study. Output of BSMAP includes the ratio of effective methylated cytosine, ratio of methylation in the Cytosine, context and some useful information that is related to the sample. Figure 3.1.1 shows standard output tables of BSMAP. Besides, if the methylation result of the sample in not in VCF format and cannot perform tabix indexing, user can upload a gzip file for a sample. The gzip file should contain chromosome name, position, methylation ratio and effective cytosine count so that the analysis can be visualized with valid data.

The tool will analyse the quality of the input at the beginning of the analysis. Poor data will result in showing inaccurate analysis. There are some repair procedures on the sample in the process of WGBS. Thus, eff_CT_count from Figure 3.1.1 shows the accurate number of effective cytosine on the real sample (without any repair).

chr	pos	strand	context	ratio	eff_CT_count	C_count	CT_count	rev_G_count	rev_GA_count	CI_lower	CI_upper
1	90	+	CHH	0.000	1	0	1	0	0	0.000	0.793
1	94	+	CHH	0.000	1	0	1	0	0	0.000	0.793
1	103	+	CHG	1.000	1	1	1	0	0	0.207	1.000
1	113	+	CHH	0.000	1	0	1	0	0	0.000	0.793
1	114	+	CHH	0.000	1	0	1	0	0	0.000	0.793
1	119	+	CHH	0.000	1	0	1	0	0	0.000	0.793
1	125	+	CHH	0.000	1	0	1	0	0	0.000	0.793
1	132	+	CHG	1.000	1	1	1	0	0	0.207	1.000
1	133	+	CG	1.000	1	1	1	0	0	0.207	1.000
1	145	+	CHH	0.000	1	0	1	0	0	0.000	0.793
1	146	+	CHH	0.000	1	0	1	0	0	0.000	0.793
1	149	+	CG	1.000	1	1	1	0	0	0.207	1.000
1	168	+	CG	1.000	1	1	1	0	0	0.207	1.000
1	172	+	CHH	0.000	1	0	1	0	0	0.000	0.793
1	177	+	CHH	0.000	1	0	1	0	0	0.000	0.793
1	184	+	CG	0.000	1	0	1	0	0	0.000	0.793
1	187	+	CHH	0.000	1	0	1	0	0	0.000	0.793

Figure 3.1.1 Input table by BSMAP (1 of the chromosome in the sample)

For the first part of the project, quality control, there will be 3 tables (or graphs) in for user to interact with. The first table shows the percentage of covered effective cytosine in the sample. The number of covered effective cytosine that is larger than 2, 5 and 10 is divided with total covered effective cytosine to generate the table. Poor sample will result in low percentage of covered cytosine. The tool enables user to select different number of reads for covered cytosine at 2x, 5x, and 10x for quality control table. 1x will show 100% for each sample so it will not be one of the selection. Table 3.1.1 shows the table that will be illustrate for quality control. However, there are no visualization API in d3.js for tables. Thus, TypeScript and HTML is used to draw the table.

Figure 3.1.2 displays the frequency and accumulative of effective cytosine in the sample. The effective cytosine count at each point will be counted and divided by the total number of sequence to get the frequency. Accumulative in Figure 3.1.2 add up the frequency at each point.

Figure 3.1.3 demonstrates distribution of the methylation level in mC, mCHH, mCHG. The methylation level corresponding to ratio from the input. The count of each ratio is sum up and divide by the total count of mCG, mCHG and mCHH. If there is five out of ten CH have 20% methylated ratio, fraction of total mC will equal to 0.5. Each sample will display their own distribution graph. Therefore, the number of graph depends on the number of sample. Methylation level in Figure 3.1.3 is the effective methylation

ratio from the input. User can control the reads accepted for the graph. The reads range from 1 to 10.

Sample	Covered cytosines(%)			
	C	CG	CHG	CHH
P-ZWf	90.47	91.63	92.43	89.64
F1-ZWf	89.32	90.68	91.50	88.38
P-ZWm	90.15	91.30	92.14	89.32
F1-ZWm	89.51	90.64	91.57	88.65
ZZm	90.23	91.41	92.26	89.37

Table 3.1.1 Percentage of covered cytosine (2x read).

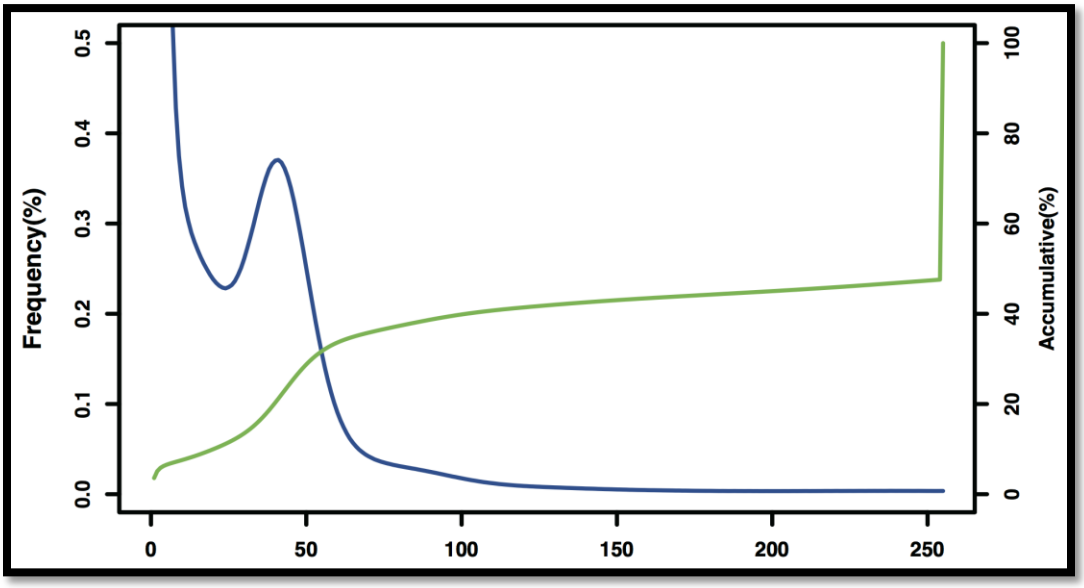


Figure 3.1.2 Distribution of the coverage depth of cytosines. Blue line represent frequency and green line represent accumulative. X-axis indicates the effective count of the sample.

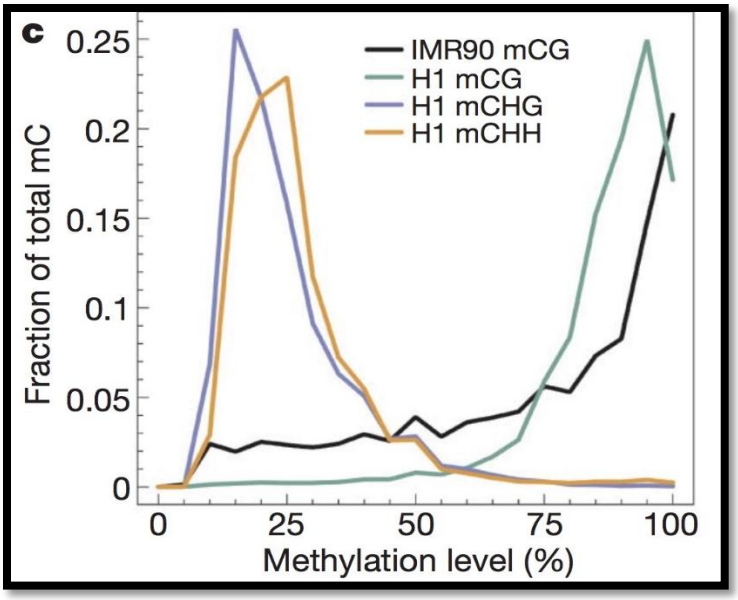


Figure 3.1.3 Distribution of the methylation level in mC, mCHH, mCHG. The y-axis shows the fraction of all methylated-cytosines in each methylation level/ratio in x-axis (Lister et. al, 2009).

Overview of DNA methylation will be visualized in part 2 of the project. The overview helps to determine the contribution of DNA methylation to variability of cell and phenotypes. Overview of DNA methylation consists of 4 graphs.

Figure 3.1.6 indicates the percentage of methylated cytosine in each sample. Number of cytosine shows the methylated cytosine in the sample. Total number of methylated cytosine in each content type is divided by total effective cytosine to get the percentage of methylated cytosine. Figure 3.1.7 shows fraction of CpG in low (<0.25), medium (> 0.25 and <0.75), and high (> 0.75) methylation levels in various genomic elements. The position of the genomic element will be provided and methylation level of each genomic element will be grouped and classified into low, intermediate and high level by comparing methylation ratio of the sample.

Figure 3.1.8 shows PCA analysis and heatmap of CpG sites for the samples. Heatmap is a very frequently used matrix in visualization of gene expression. Heatmap that shows the methylation ratio of each position is calculated. Due to large amount of data from many samples, the methylation ratio is divided by 10k to ensure the heatmap can be visualized smoothly. Euclidean distance is used to calculate the distance between each sample. The distance between samples form a distance matrix by getting minimum among the matrix. Dendrogram between samples is visualized by using distance matrix.

PCA analysis is a technique that used to analyze and simplify the data into principle component. The PCA analysis can be performed on whole genome, CpG Island, promoter and other genomic region. Methylation pattern is identified and clustered using PCA analysis. The color of the sphere inside the cube is colored based on the group of sample. From that, user can identified the characteristics of methylation pattern throughout figure 3.1.6.

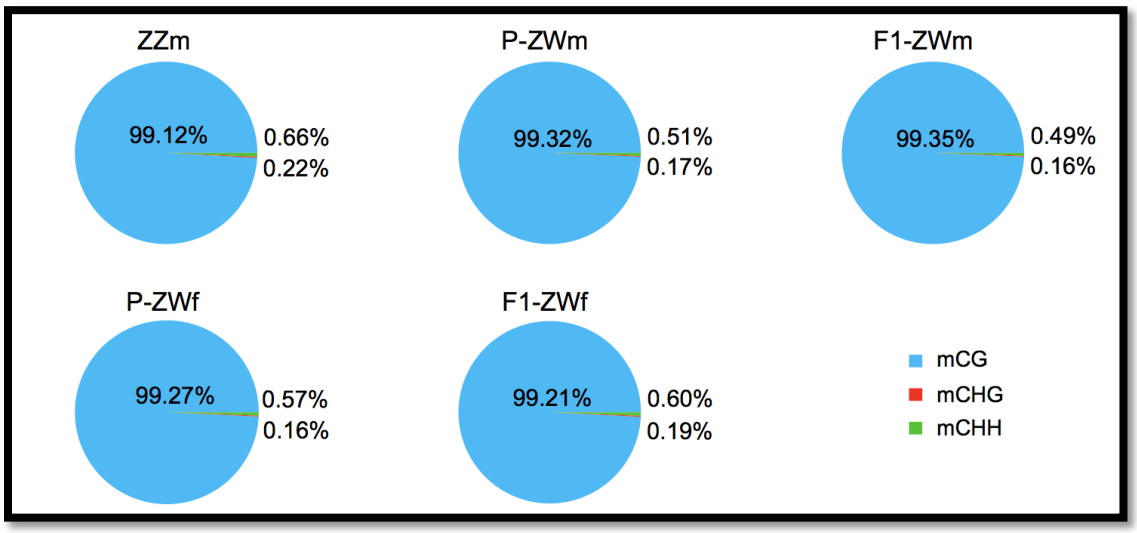


Figure 3.1.4 Percentage of methylated cytosines including mCG, mCHG and mCHH.

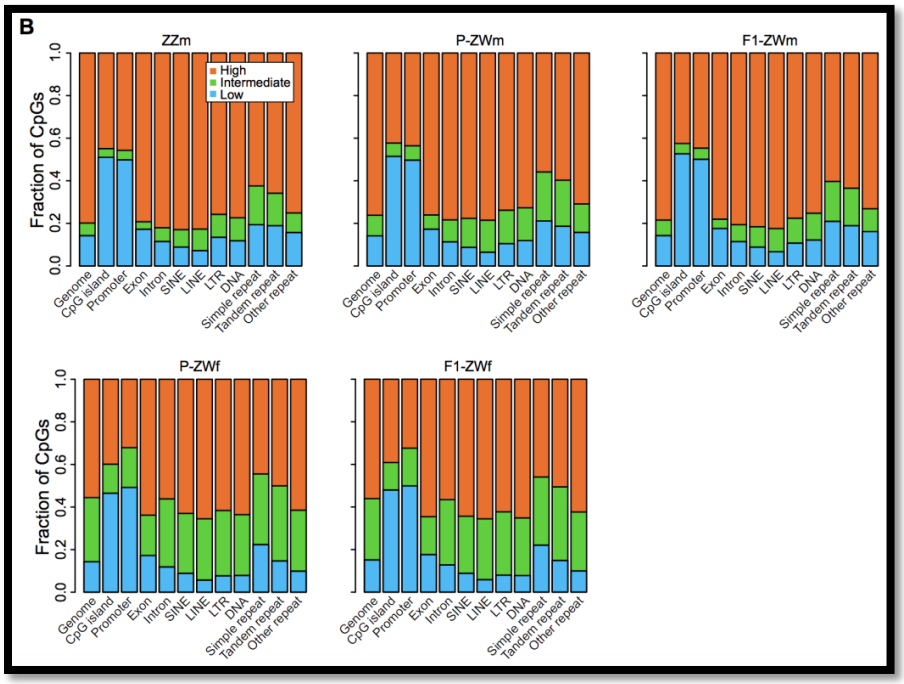


Figure 3.1.5 Element target. Fraction of CpG in low (<0.25), intermediate (> 0.25 and <0.75), and high (>

0.75) methylation levels (Shao et. al, 2014)

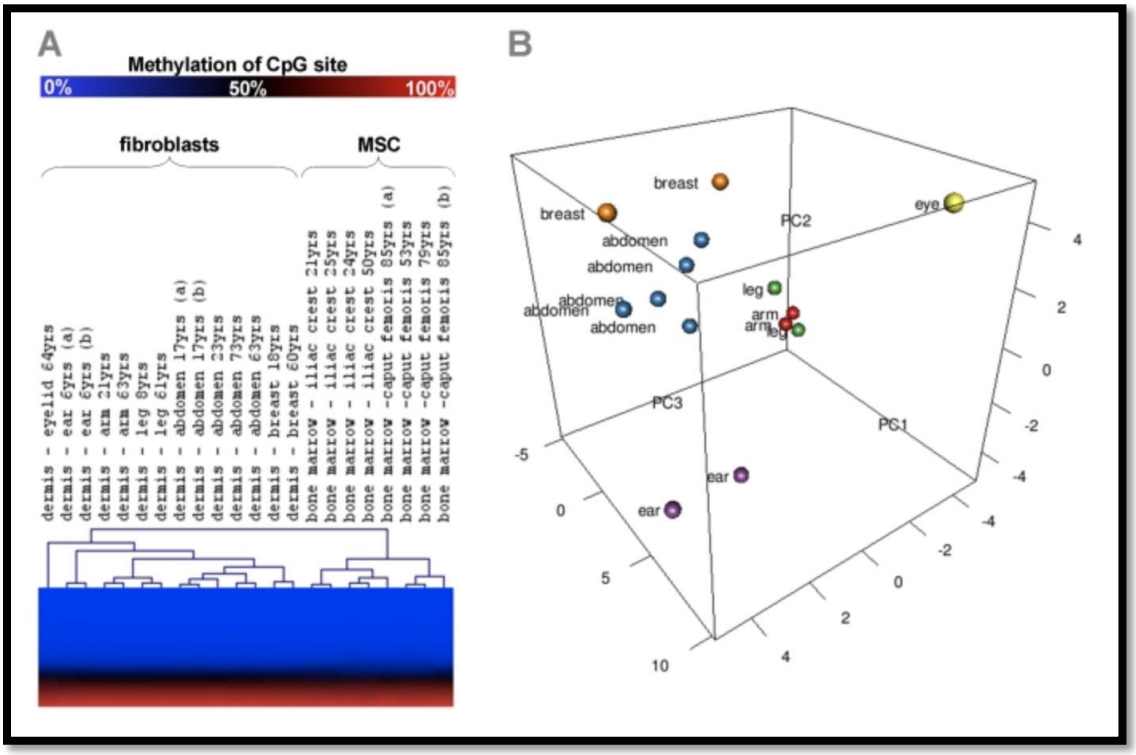


Figure 3.1.6 Clustering and PCA analysis of methylation of CpG sites across samples (figure is just for illustration purpose)

The system that is going to be used for the project is run on Ubuntu; python2.7 is used to run the server. The software used in the project are git, tabix, Python2.7/pip, django, Postgres SQL, node.js and npm. Git is used to clone and upload the project from the platform developer. The server runs on Python2.7. Django is the web framework used in the project. The web framework used in the project is developed using django and node.js. Postgres SQL is an open source database used in the project. Visual Studio Code is used to edit and view the code of the project. Tabix is used in fast retrieval of large data genomic data files.

The languages used in the project are TypeScript, SASS, d3.js, three.js and SVG. SVG is an XML-based markup language that is used to define vector based graphics in XML format. SVG allows every component to support interactivity and animation. Drawing area is the part that will display the major graph. The graph will be displayed by using SVG in order to let user able to interact with every elements in the figure.

D3.js will be used to handle the interaction between user and the figure. D3.js is a JavaScript library that helps to visualize the figure. It helps to provide dynamic and visualization data figure. D3.js helps us to handle SVG efficiently. It also provides some elements just like html that is always used in visualization.

HTML and CSS are the indispensable elements in a web page development. HTML is a typical markup language that is used to develop web pages while CSS is used to present the data and attributes in HTML according to different interpretation method or styles. SASS act as the extension of CSS makes CSS to become more powerful by having more attributes and elements. Python is used in data processing and some statistical analysis throughout the module. Python is powerful in handling huge amount of data.

Typescript will be one of the main language used in the project. Typescript is based on ES6 that provides all the features in JavaScript. Typescript handles complicated data structure easily.

Functional testing and interface testing will be performed to ensure the visualization of DNA methylation is performing well. Interface testing is used to ensure the flow of the modules go smooth while functional testing test the function of every module. Functional testing include database testing and flow testing will be used to ensure the graphs displayed well without error. Performance of the visualization will be tested through different size of file. The visualization planned to work well with large data file.

CHAPTER 3: PROPOSED METHOD/APPROACH

SECTION 3.2-SYSTEM DESIGN / OVERVIEW

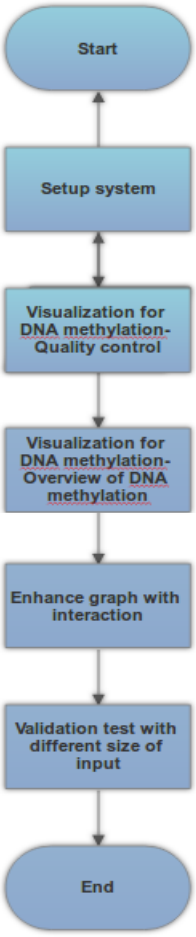


Figure 3.2.1 Workflow of the project

SECTION 3.2.1-SYSTEM SETUP

Figure 3.2.1 shows the work flow of the project. The system is setup before visualization start. First of all, Ubuntu is installed in the laptop. Git is downloaded on Ubuntu using the following command:

```
$ sudo apt-get update
$ sudo apt-get install git
```

Tabix is installed by the following command. Tabix is used to retrieve the genomic file in BED, GFF, VCF or SAM format. Segment, starting position and ending position are the standard parameters in tabix indexed file.

```
$ sudo apt-get update
$ sudo apt-get install tabix
```

Python 2.7 and pip is installed by using the following command. The command below install dependencies for Python2.7.

```
$sudo apt-get install build-essential checkinstall
$sudo apt-get install libreadline-gplv2-dev libncursesw5-dev libssl-dev
libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev
```

The following command helps to download Python2.7.13:

```
$version=2.7.13
$cd ~/Downloads/
$wget https://www.python.org/ftp/python/$version/Python-$version.tgz
```

Then, extract the downloaded file and change to the directory where the file is located

```
$tar -xvf Python-$version.tgz
$cd Python-$version
```

Finally install Python 2.7 using the following command. ./configure checks whether the application is ready to install and shows the errors if building of application failed. Checkinstall command keeps track of all files installed by make install. It also simplify the process for package removal or distribution.

```
$./configure
$make
$sudo checkinstall
```

After Python2.7 is installed, pip is installed. The command used to install pip is listed as below. First command install Easy Install for Python packages. Then pip is installed and followed by virtualenv.

```
$ sudo apt-get install python-pip python-dev build-essential
$ sudo pip install --upgrade pip
$ sudo pip install --upgrade virtualenv
```

Postgres SQL is installed to manage the database. Command used to install Postgres package is recorded as below. Postgresql -contrib package add more utilities and function to Postgres SQL

```
$sudo apt-get update
$sudo apt-get install postgresql postgresql-contrib
```

Lastly, node.js and npm are downloaded. The latest version of node.js is needed in the project. Curl is a tool that helps in retrieving files to and from a server through ftp, http, https and other supported protocol. 3rd command and 4th command used to install required PPA for latest Node.js on Ubuntu. 5th command installed node.js and other dependencies on Ubuntu. Last two commands help us to check on the version of node.js and npm to ensure latest node.js and npm are installed.

```
$sudo apt-get update
$sudo apt-get install curl
$sudo apt-get install python-software-properties
$curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
$sudo apt-get install nodejs
$node -v
$npm -v
```

Once everything is installed, the project is cloned by using the following command.

```
$git clone git@git.lhc.moe:dovirus
```

After finished cloning the project, go to dovirus directory to install packages.

```
$cd dovirus
$pip install -r requirements.txt
$npm install
```

The project is initialized after installation of the packages.

```
$echo -e "BVD3_ENABLE_SAMPLES = False\n
BVD3_INDEX_PACKAGE = 'dovirus' "  > bvd3/settings_bvd3.py
```

The database is setup by using the following command. 1st command is used to create a user named virus while 2nd command is used to create a database. -O in 2nd command represent owner. 3rd command switch to server using postgres account. Last command changed the password of user named virus to ‘virus_test’.

```
$sudo -u postgres createuser virus --createdb
$sudo -u postgres createdb -O virus virus_dev
$sudo -u postgres psql
#ALTER USER virus WITH PASSWORD ‘virus_test’;
```

The following command helps to migrate existing py file to the packages and create a super user for the project.

```
$python manage.py migrate
$python manage.py createsuperuser
```

Lastly, run the server and the platform is successfully setup. The platform can be accessed through http://localhost:8000/admin/ . A new project is created to generate graph in the platform as shown in Figure 3.2.2.

```
$python manage.py runserver
$npm run watch
```

SECTION 3.2.2-OVERVIEW OF PLATFORM

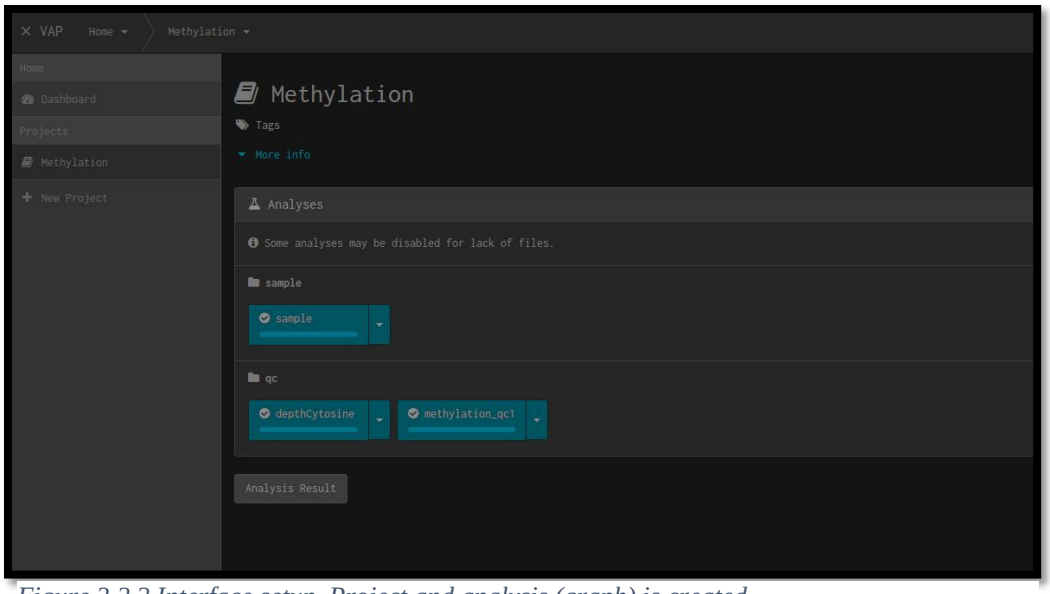


Figure 3.2.2 Interface setup. Project and analysis (graph) is created.

There are a lot of directory in the repository in Figure 3.2.3. Bvd3 stores the setting needed for the platform. Db is the directory created when the project is created. File uploaded will be stored inside db under a file key. node_modules is the modules that is created after node.js installed. Virus directory is the directory that the visualization module files will be stored. Requirements.txt states the requirements of the software like Django that are going to be installed properly after the project clone. Manage.py checked the installation of Django on the system. If Django is successfully installed, setting_shared.py in bvd3 will be executed. All the database, password, application information and some other related information are listed in bvd3/setting_shared.py.

Reader directory stores python code for another file reader mode in which users need to upload whole dataset and the data is processed and exposed to a JSON API. Page directory includes the files that will be used for visualization such as drawing a table. Templates directory includes many djhtml template files used in the website. However, the visualization that is going to be developed in the project will not be focused on the interface of the webpage.

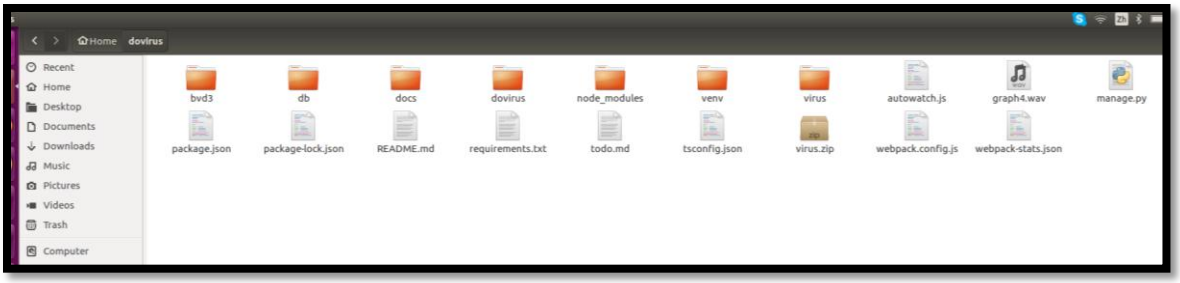


Figure 3.2.3 File structure of dovirus repository.

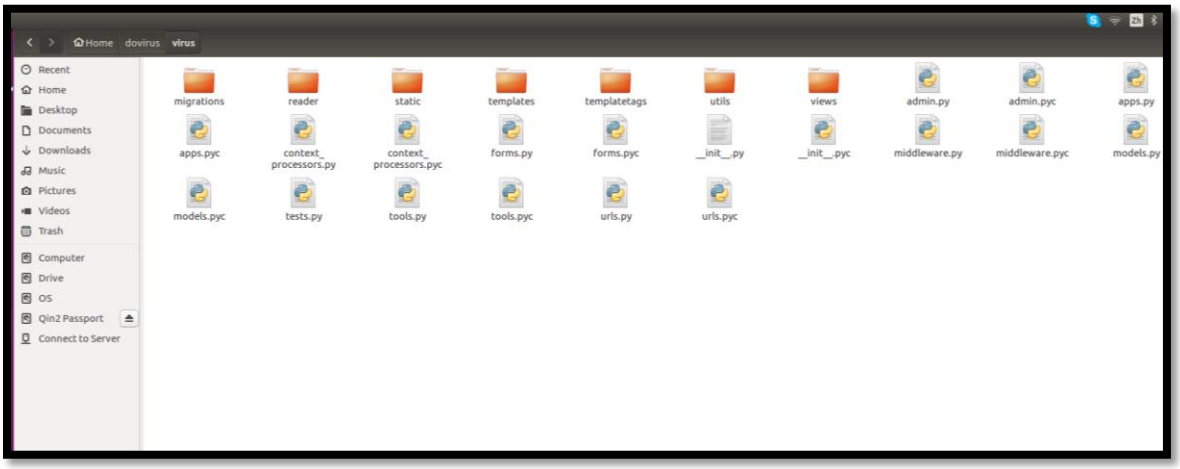


Figure 3.2.4 File structure of virus file. Virus file is the main directory that will often be used throughout the project.

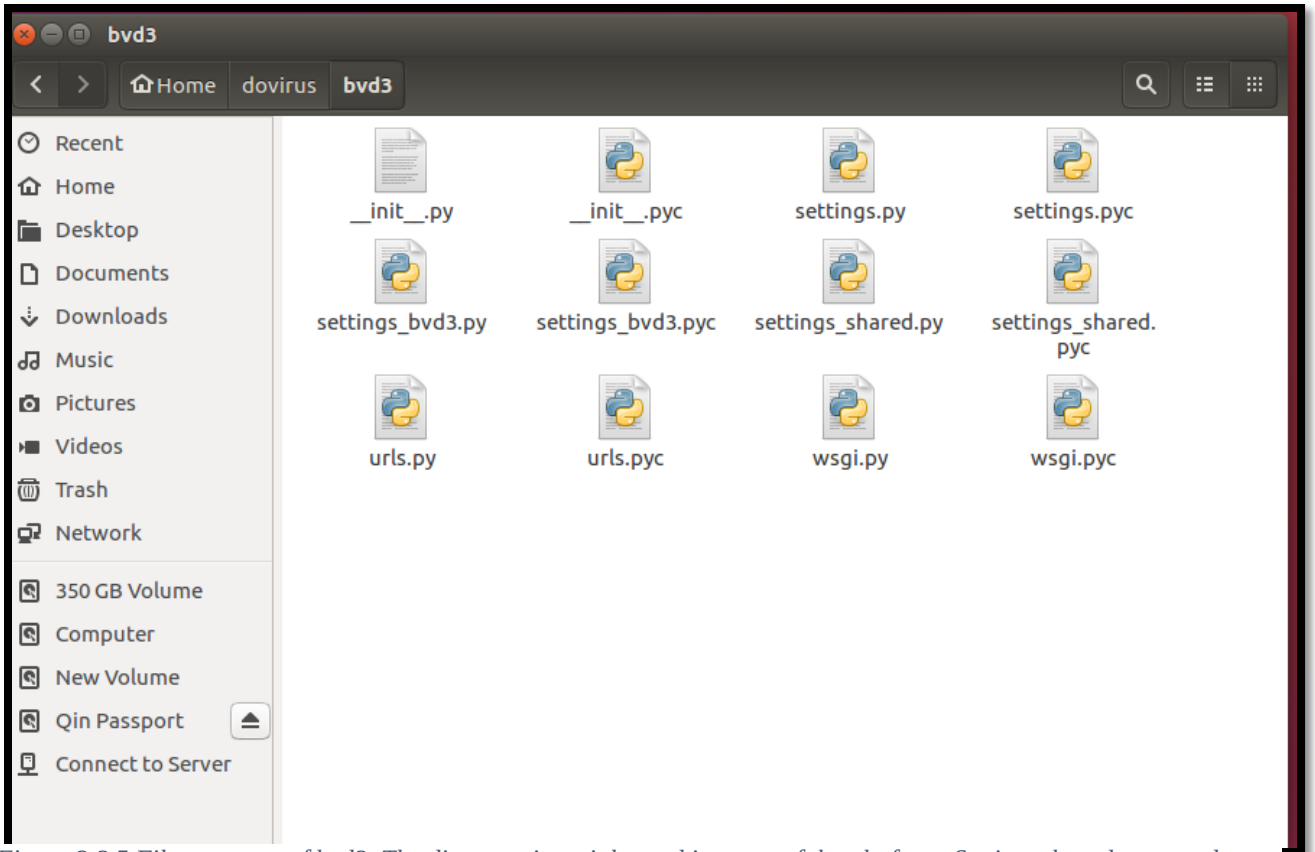


Figure 3.2.5 File structure of bvd3. The directory is mainly used in setup of the platform. Setting_shared.py stored details about the database and password of the platform.

Static directory stores app directory that will be used in visualization, bvd3 directory will be used in component defination, css directory stores sass file that responsible for some css design in login page and other related page. App directory stores the analysis files that are used for visualization of DNA methylation. The analysis module file consists of several files that are used to visualize the graph. Controller.ts manages editor setup of the graph while visualization.ts manages the visualization part and file reading part.

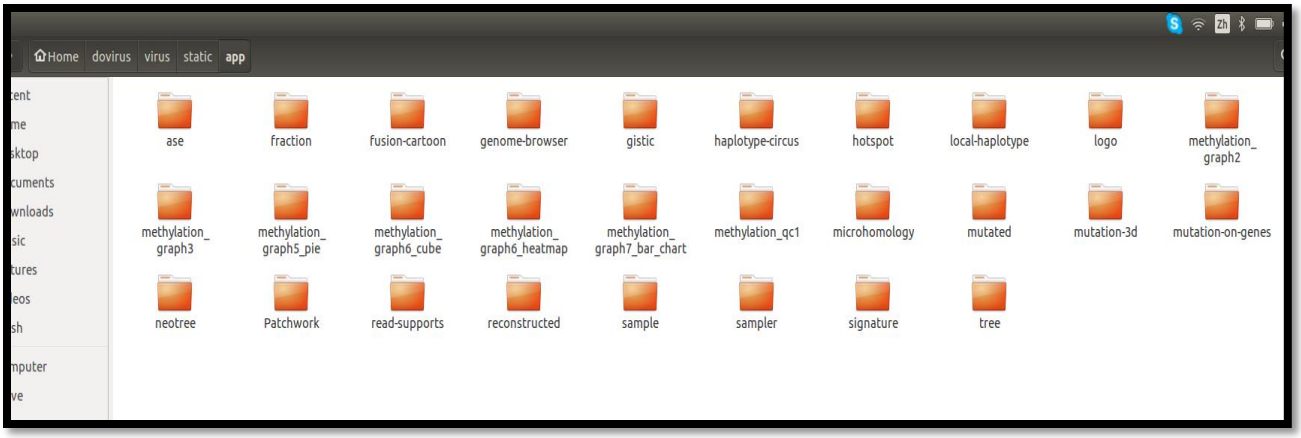


Figure 3.2.6 File structure of static. Static file is the file that stored most of the coding that used in the platform. Graphs, front end coding of the website and some elements structure file is stored in static directory.

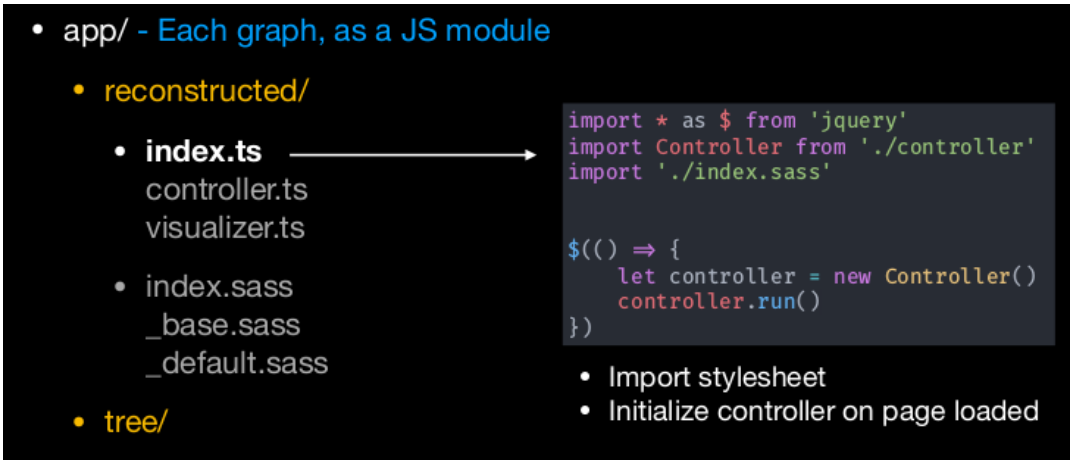


Figure 3.2.7 File structure of each graph. Reconstructed (graph) is illustrated in the figure.

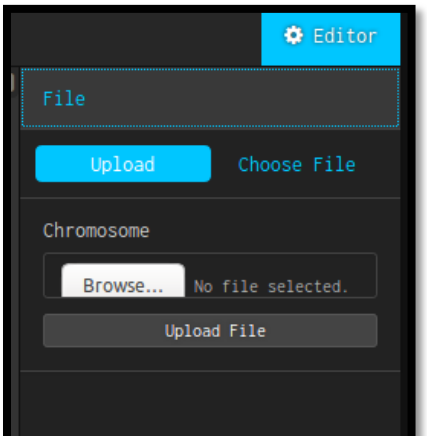


Figure 3.2.8 Editor that retrieved input from user to perform analysis. Upload file is one of the action.

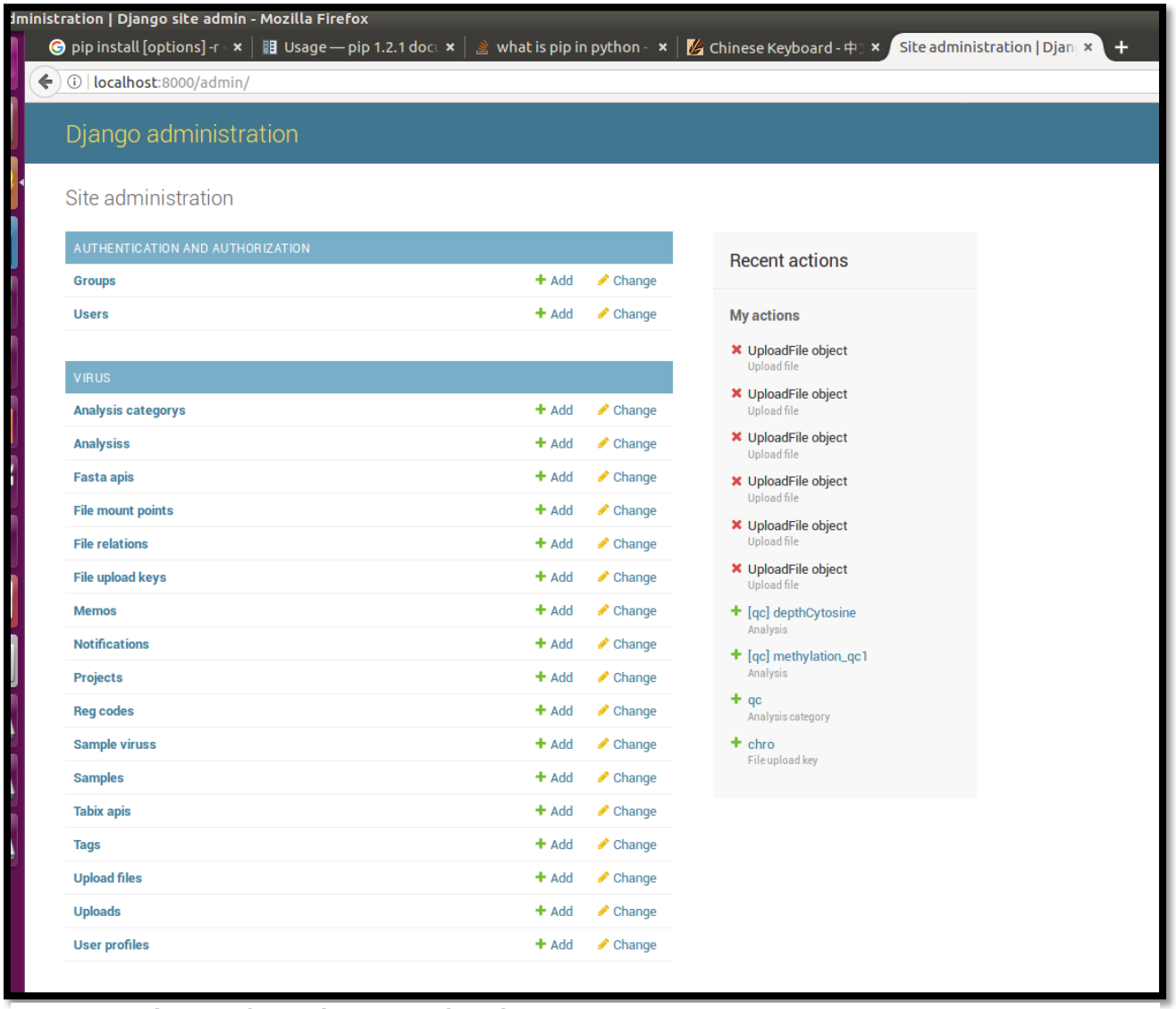


Figure 3.2.9 admin site that used to manage the website.

Js module needs to be added in analysis at localhost:8000/admin in order to visualize the graph. Let's take sample graph as example. The details of the graph that is going to be visualized will be added like Figure 3.2.12. One JS file represent one graph. File keys represent the files that need to be uploaded to the key. In the sample graph, user needs to upload two files. The files' path will be saved in the uploaded file as Figure 3.2.14.

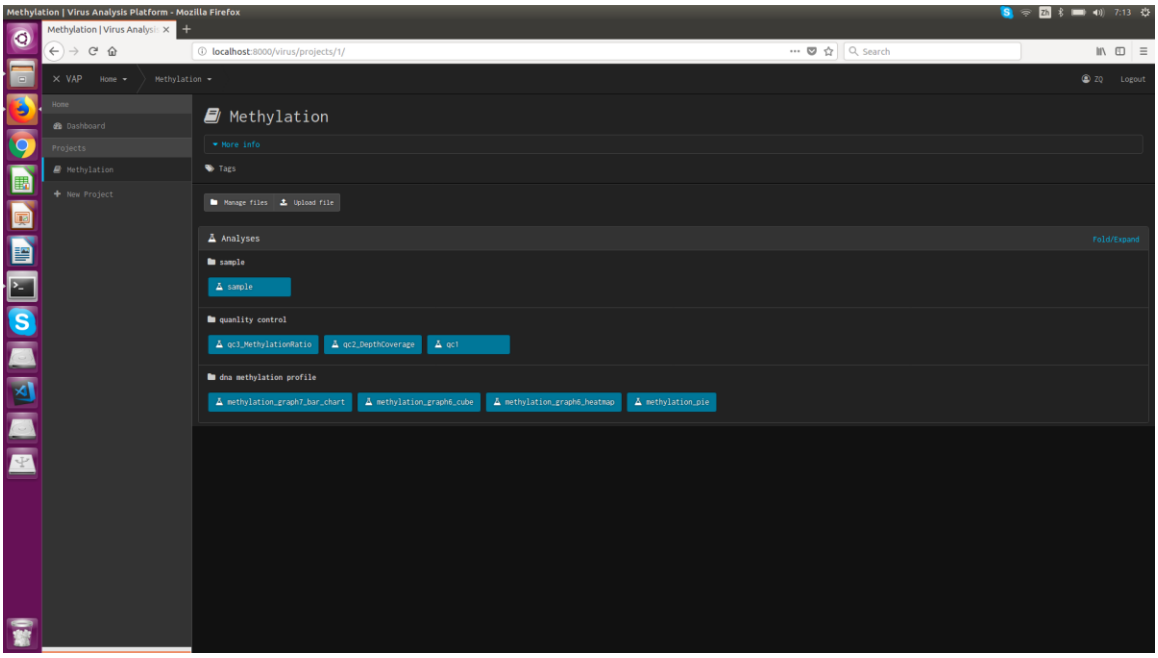


Figure 3.2.10 After selected on analysis in admin site, the analysis (graph) of the project will be displayed according to category.

Figure 3.2.11 User can add analysis (graph) to the project.

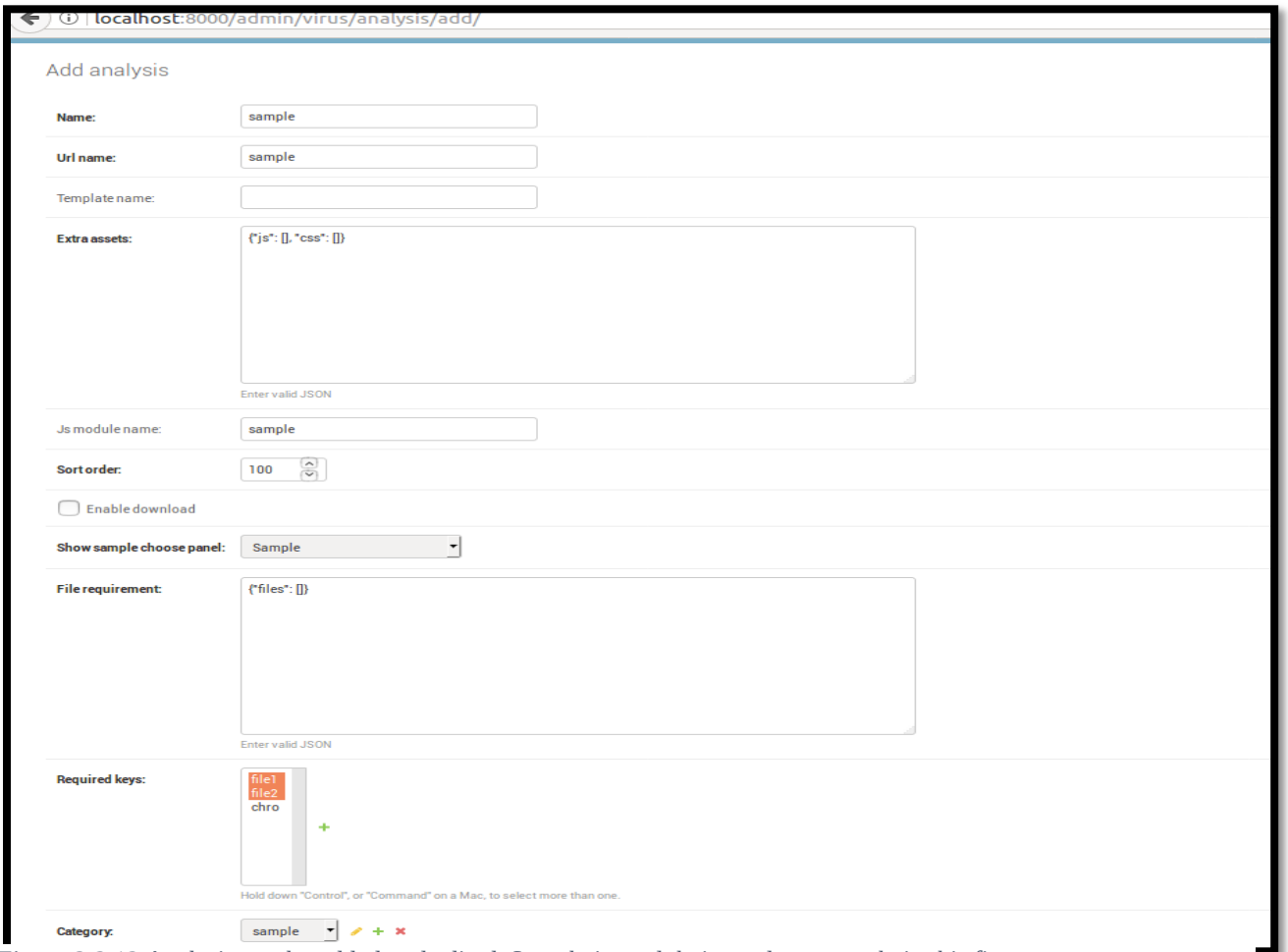


Figure 3.2.12 Analysis can be added and edited. Sample js module is used as example in this figure.

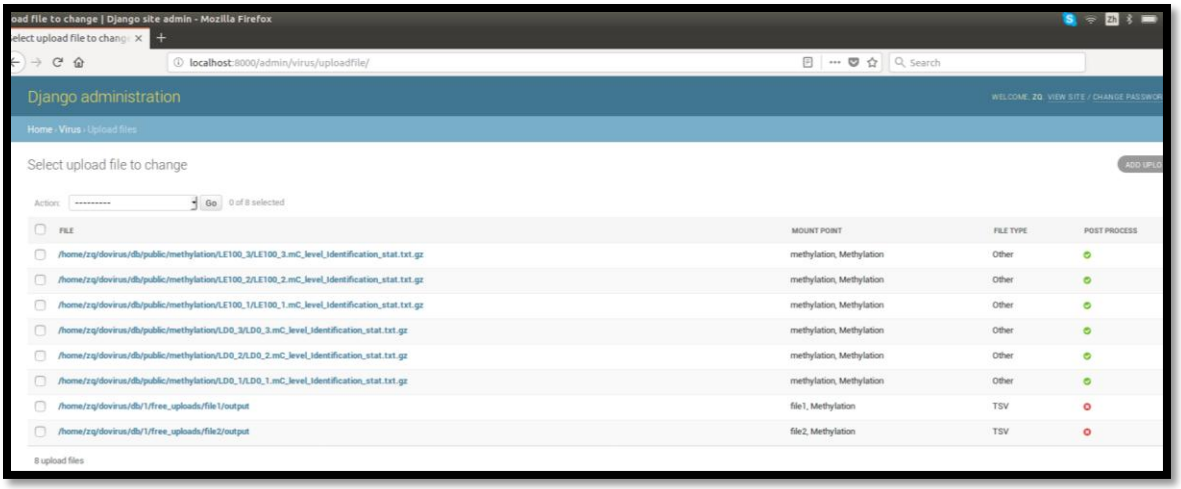


Figure 3.2.13 File uploaded to the database. File directory is stored under their file key. In this example, output is stored under file key- file1 in Methylation project. The file will be read in TSV mode.

Figure 3.2.15 shows the structure of the coding. Data will be read and processed in run().Variables in apiMap represent the files that are going to be uploaded to the file key. The visualization coding can be run and debugged using any browser like Chrome or Firefox. The error messages in console helps to find error.

```
alizer.ts .../methylation_graph2 ●    TS visualizer.ts .../sample ●

    public size: Size
    public option: Option
    public data: Data
    public shared: ASESharedObj

    public controller: Controller

    public svg: D3HTMLSelection

    static apiRoot: string = "/virus/api"
    static dataTypes = ["data1", "data2"]
    static apiMap = {
        data1: {file_key: "file1"},
        data2: {file_key: "file2"}
    }

    static defaultOptions: Option = {
        debugMode: false,
        svgWidth: 1200
    }

    constructor() {
        super()
    }

    public init(op?: Option) {
        super.initVisualizer(this, op)
        this.size = new Size(this.option.svgWidth)
    }

    public run(): void {
        this.loadText("data1", (d) => {
            this.data.data1 = "1"
        })
        this.loadText("data2", (d) => {
            this.data.data2 = "1"
        })
    }

    public redraw(layers: string[] | string) {
        this.isRedraw = true
        this.size = new Size(this.option.svgWidth)
        super.redraw(layers)
    }

    protected afterDataLoaded() {
        this.processData()
    }

    protected afterDrawing() {

    }
```

Figure 3.2.14 Structure of program code. Graph is visualized in a. File is loaded and data is processed in b.

CHAPTER 4:

RESULT DELIVERED

SECTION 4.1- HANDLE DATA

Tabix is a genomic tool that indexes bgzip file and retrieves data from the file. It is the solution found to handle large genomic data files uploaded by user. Although there is a default tabix api built in the platform, it is handled by admin only and does not support upload function for tabix api. Thus, a modification on the upload file function is needed.

In the project, UploadFile model is modified to process large amount of data. File reading function that provided by d3 is not enough to read large amount of data. Browser has a small processing power and memory. Thus, in order to handle the data, UploadFile function is enhanced. The UploadFile model is changed from storing file, file type and mount point to storing file, file type, mount point, post-processed indicator, non-tabix indicator, file attribute pattern, chromosome list , starting and ending list of chromosome, sample upload file for tabix (sample name), sample upload file for gzip file (sample name).

There are 4 necessary information included in a sample file: position, effective cytosine count, methylation ratio and context type. These 4 attributes will be named as “pos”, “methyC”, “ratio” and “contexttype”. The attributes in all methylation input files should follow these naming. All the tabix files with same sample name will be grouped under same sample. Multiple tabix input file is accepted for one sample while gzip file can only upload one file for each sample. UploadFile model accepts 3 different upload file types: a normal text file- CSV, TSV or other file types with no post-processing steps.

A tabix indexed file with chromosome name, position, methylation ratio, context type, effective cytosine count can be uploaded. This type of file will perform some post-processing steps such as generating tabix indexed file and generating data that will be used for methylation analysis. The tabix file should be uploaded in bgzip format and can be indexed by tabix by using VCF type. To upload a tabix file, user needs to upload the file and provide the following information:

a) Mount point: methylation- Methylation should be chosen for all samples that are required to generate graph. Only the sample in this file will generate the analysis.
b) Post process: The checkbox needs to be checked and indicates whether the file needs to be processed.
c) Sample upload file: Select a Tabix api model that stored the related information like file name, column type and column name. New Tabix api model can be added for a new sample. Files for a sample should have same sample upload file.
d) Chromosome list: One file can contain one or many chromosomes. If there is only one chromosome, the input for chromosome list will be chr1. If there are many chromosomes, the input for chromosome list will be chr2,chr3. The list of chromosome should be separated by comma and no space in between.
e) Starting list: For the file that only contains one chromosome, only one starting index is needed. For instance, 10000. For the file that consists of many chromosomes, starting index for each chromosome needs to be listed according to their position in chromosome list respectively. For example, 10000,20000. List of starting index for chromosome needs to be separated by comma and no space is allowed.
f) Ending list: For the file that only has one chromosome, only one ending index of the chromosome is needed. For instance, 94100000. For the file that consists of many chromosomes, ending index for each chromosome needs to be mentioned according to their position in chromosome list respectively. For example, 12000000,20025000. List of ending index for chromosome need to be separated by comma and no space is allowed.

Besides, a gzip file with information needed can be uploaded and similar process applied in tabix file will be performed in the gzip file. The file should be zipped using gzip. To upload a gzip file, user needs to upload a gzip file and complete related information:

a) Mount point: methylation- Methylation should be chosen for all samples that are required to generate graph. Only the sample in this file will generate the analysis.
b) Sample upload zip: Name of the sample and should not contain “/”.

c) File attribute pattern: Attributes inside the sample. For unused attributes, use _ to skip the attributes. For example, _,pos,methyC,ratio,contexttype. List of attributes for chromosome needs to be separated by comma and no space is allowed.

```
class UploadFile(models.Model):
    TYPES = (('CSV', "CSV"), ('TSV', "TSV"), ('OTH', "Other"))

    file = models.FileField(upload_to=file_upload_path)
    mount_point = models.ForeignKey(FileMountPoint, related_name='files')
    updated_time = models.DateTimeField(auto_now=True)


    file_type = models.CharField(max_length=3, choices=TYPES, default='OTH')

    sample_upload_tabix = models.ForeignKey(TabixAPI, related_name='%(class)s_url_name',blank=True, null=True,help_text="Enter "
                                            "sample name for the sample that will be used in the analysis.")
    sample_upload_zip = models.CharField(max_length=256,blank=True,help_text="Enter "
                                         "sample name for the sample that will be used in the analysis.")
    chromosome_list=models.CharField(max_length=256,blank=True,
                                     help_text="Enter a list of string (chr1,chr2,chr3) separated by comma, representing the chromosome "
                                               "contained in the file.")
    starting_list=models.CharField(max_length=256,blank=True,
                                   help_text="Enter a list of string (0,1,2) separated by comma, respect to the chromosome list. ")
    ending_list=models.CharField(max_length=256,blank=True,
                                 help_text="Enter a list of string (1000,2000,3000) separated by comma, respect to the chromosome list. "
                                           "and starting_list.")
    file_attribute_pattern=models.CharField(max_length=1024,blank=True,
                                            help_text="Enter a list of string separated by comma; a single underline (_) "
                                                      "stands for ignoring this column.")
    post_process=models.BooleanField(default=False)
    non_tabix=models.BooleanField(default=False)
```

Figure 4.1.1 Model for UploadFile.

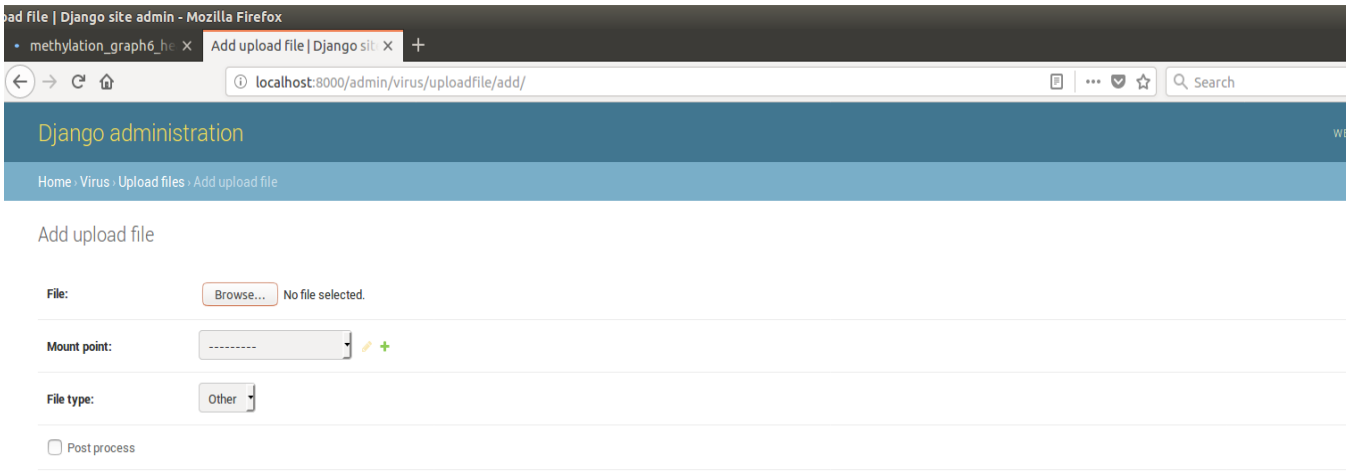


Figure 4.1.2 Normal file upload

CHAPTER 4: RESULT DELIVERED

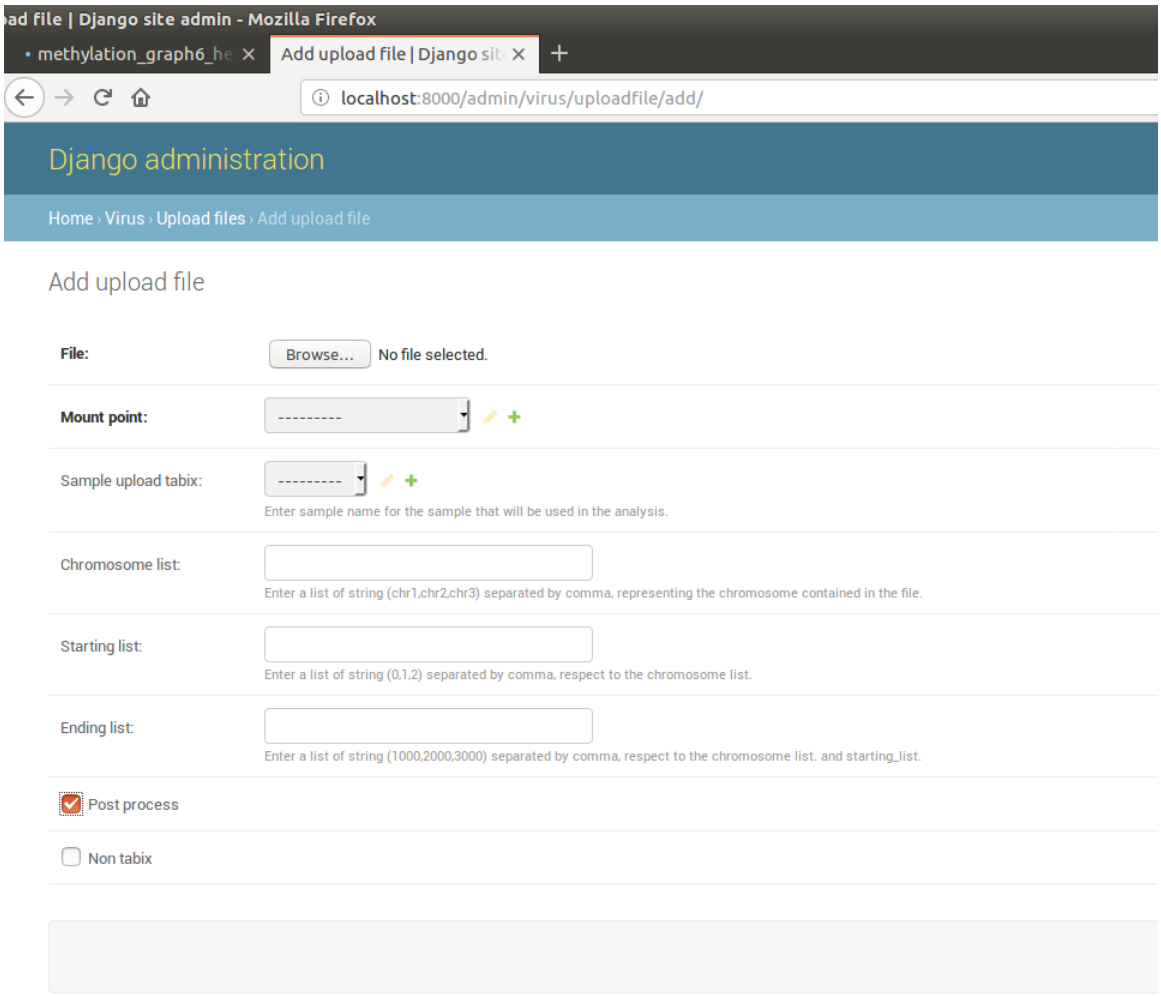


Figure 4.1.3 Tabix file upload interface

CHAPTER 4: RESULT DELIVERED

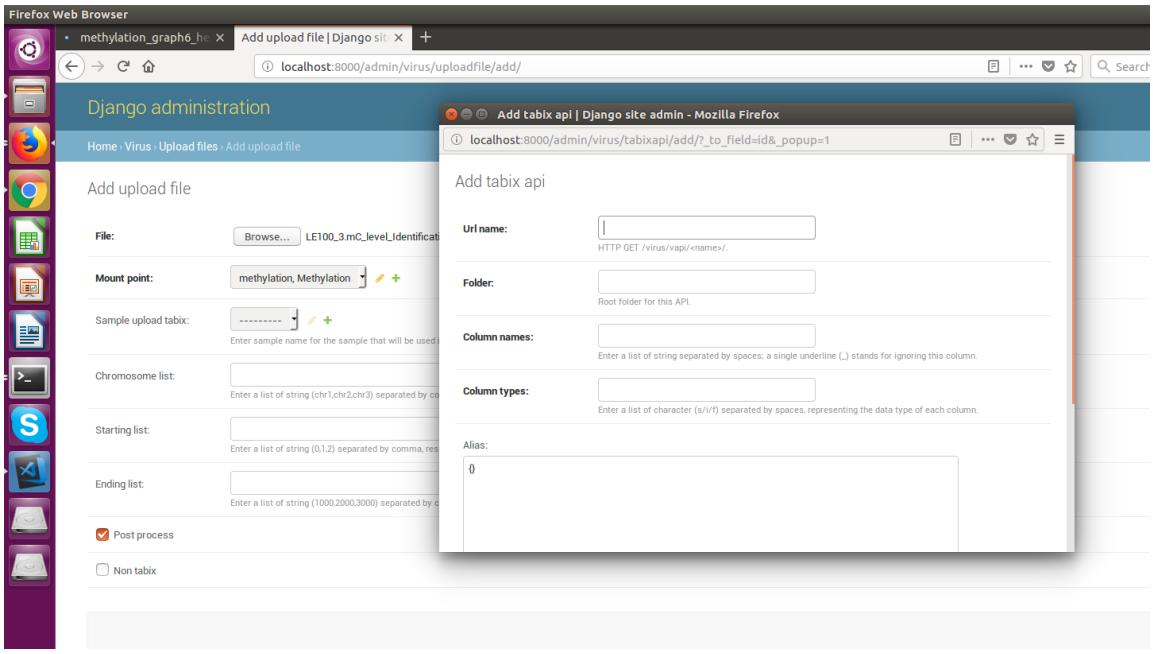


Figure 4.1.4 Add new tabix api for new sample.

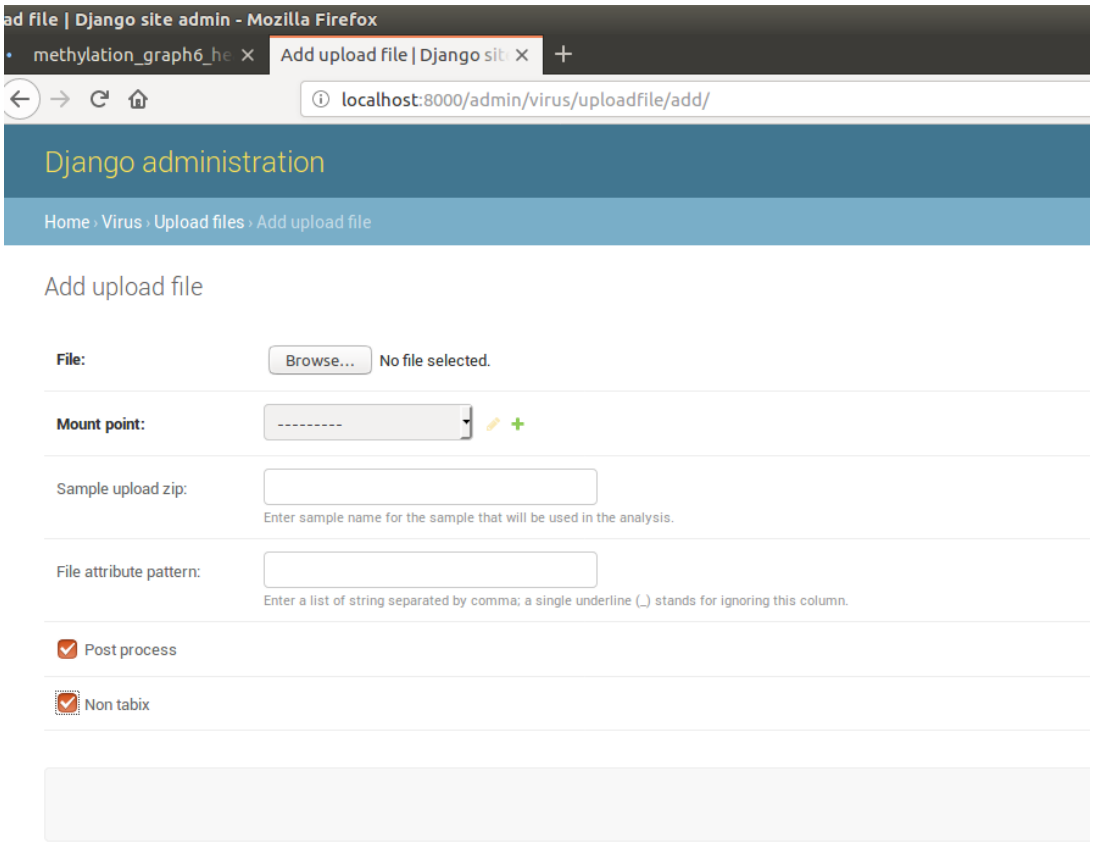


Figure 4.1.5 GZIP file upload interface

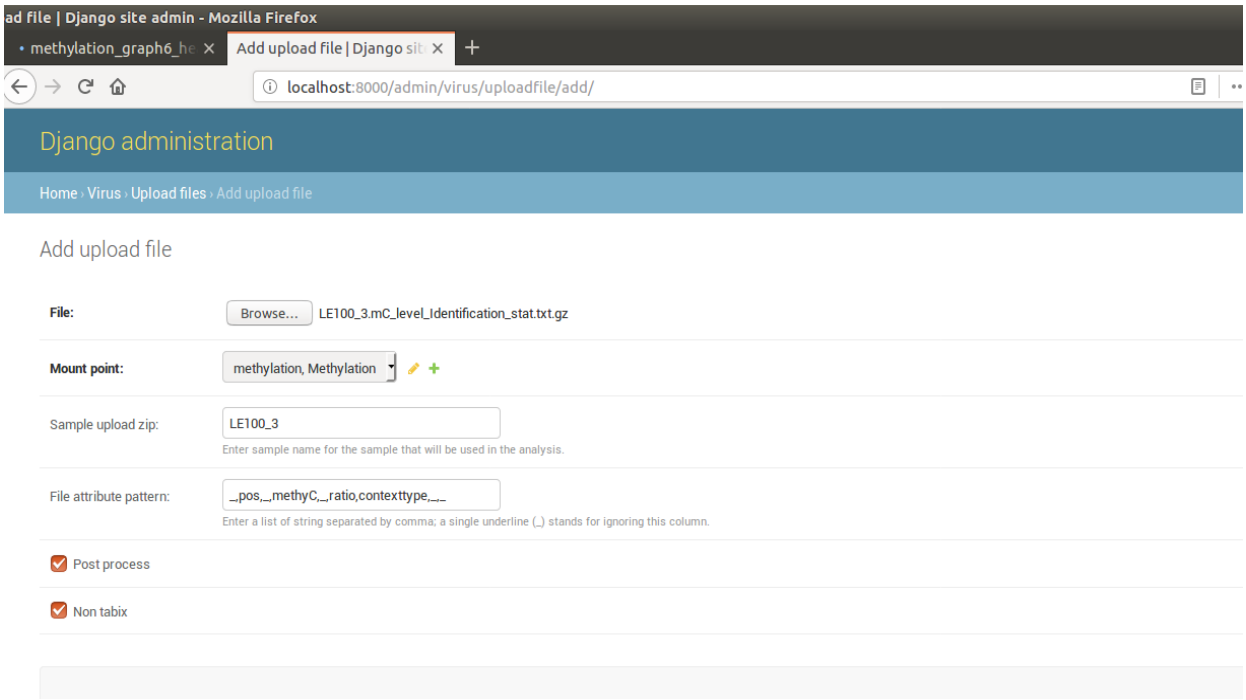


Figure 4.1.6 Example of GZIP file fill in context

Tabix_index helps to create a tabix indexed file. P.wait() is called to wait tabix indexed file created before running another function. After creating tabix indexed file, process_tabix_api is used to process data and generate file. If uploaded file is in gzip format, another process will be run to generate data.

```
@staticmethod
def tabix_index(mount_point,sample_file,filename,
    preset="vcf"):
    """Call tabix to create an index for a bgzip-compressed file."""
    p=Popen(['tabix', '-p', preset, os.path.join(settings.BASE_DIR, 'db','public',
                    mount_point,sample_file,
                    filename)])


    #This makes the wait possible
    p_status = p.wait()
```

Figure 4.1.7 Subprocess that generated tabix indexed file

```
def save(self, *args, **kwargs):
    #add error handling !!!!!!!!!WAITING
    print(time.strftime('%l:%M%p %Z on %b %d, %Y'))
    super(UploadFile, self).save(*args, **kwargs)  # Call the "real" save() method.
    if self.post_process == True and self.non_tabix==False:
        self.file.storage.generate_filename(self.file.name)
        self.tabix_index(self.mount_point.fkey.name, self.sample_upload_tabix.url_name,self.file.name)
        self.process_tabix_api(self)
    elif self.post_process == True and self.non_tabix==True:
        self.file.storage.generate_filename(self.file.name)
        processed_data=self.process_zip_api(self)
        self.bindtofile(self,processed_data,1)
```

Figure 4.1.8 The file will be saved and processing of file start

All the processed files will be stored in db/public/methylation/{sample name}. The data for a sample will be processed in `def process_zip_api (self)` and `def process_tabix_api (self)`. The details of data processing will be explained in the respective section for the graph. Figure 4.1.9 (a) shows that program is reading the position that will be considered in graph 7- stacked bar chart according to region. All file names of position range for respective element will be stored in element.list. These region ranges will be provided by system to process and generate related data for stack bar chart. `Stack_bar_data` is initialized as shown in Figure 4.1.9 (a). `Io.BufferedReader` is used when reading gzip file content as shown in Figure 4.1.9 (b). It will be much faster compared to using `gzip` only. With only `gzip`, it will be two times slower than `Io.BufferedReader`.

```
@staticmethod
def process_zip_api(self):
    result = []
    count={}
    total={}
    row_return={}
    list_cytosine=['C','CG','CHG','CHH']
    list_cytosine_3=['CG','CHG','CHH']#use for qc3
    depth_cov_by_methylC={}
    ratio_frequency=[]
    context_percentage={}
    total_ratio=0

    read_list=[1,2,5,10,100]
    start=time.time()
    range_list={}
    element_file_name={}
    stack_bar_data={}
    #process_file_get_position for bar chart
    element_sum_file_name=os.path.join(settings.BASE_DIR, 'db','public',
                                       "methylation","region",
                                       "element.list")
    with open(element_sum_file_name, "r") as elements_file:
        for element_ind in elements_file:
            element=element_ind.split('\n')[0].split('\t')
            if not (element[0]=="#Element"):
                element_file_name[element[0]]=element[1]
        print element_file_name


    for element_ind in element_file_name:
            element_small_file_name=os.path.join(settings.BASE_DIR, 'db','public',
                                                 "methylation","region",
                                                 element_file_name[element_ind])
            print "reading file"+element_small_file_name
            with  open(element_small_file_name, "r") as element_line:
                    range_list[element_ind]=[]
                    for line in element_line:
                        temp=line.split('\n')[0].split('\t')
                        range_list[element_ind].append((temp[1].strip().split(' ')[0],temp[2].strip().split(' ')[0]))
            sorted(range_list[element_ind])
            stack_bar_data[element_ind]={}
            stack_bar_data[element_ind].setdefault("0.25-0.75",0)
            stack_bar_data[element_ind].setdefault("0.25",0)
            stack_bar_data[element_ind].setdefault("0.75",0)
            print element_ind
            element_line.close()
```

(a)

```
for ratio_fre_Cnt in range(0,10):
    ratio_frequency.append({})

for index in list_cytosine:
    for read_ind in read_list:
        count[index+str(read_ind)]=0
        total[index+str(read_ind)]=0
    for ratio_fre_Cnt in range(0,10):
        ratio_frequency[ratio_fre_Cnt][index]={}
    context_percentage[index]=0
context_percentage["total"]=0


row_return['read_num']=read_list
row_return['seg_name']=''
```

(b)

```
with gzip.open(self.file.name,'r') as gz_read:
    col_list=self.file_attribute_pattern.split(",")
    print "read-in buffer"
    with io.BufferedReader(gz_read) as f:
        print "for-loop start"
        for line in f:
            entry=line.split("\n")[0]

            col_count = -1
            row={}
            row_split=entry.split("\t")
            if col_count < 0:
                col_count = len(row_split)
            for i in xrange(col_count):
                col = row_split[i]
                col_name = col_list[i]
                if col_name == '_': continue
                row[col_name] = col
            fre=int(row["methyC"])
            #qc3
            for index in list_cytosine_3:#exclude C #maybe need to directly calculate total
                if (row["contexttype"])==index:
                    if row["ratio"]!="NA":
                        total_ratio+=1
                        for ratio_fre_Cnt in range(0,10):
                            if ratio_frequency[ratio_fre_Cnt][index].get(row["ratio"])==None:

                                if fre>=ratio_fre_Cnt:
                                    ratio_frequency[ratio_fre_Cnt][index][row["ratio"]]=1
                            else:
                                if fre>=ratio_fre_Cnt:
                                    ratio_frequency[ratio_fre_Cnt][index][row["ratio"]]+=1
                    #dmp3
                    context_percentage[index]+=1
                    context_percentage["total"]+=1
                    break


            #qc1
            for read_ind in read_list:
                for index in list_cytosine:#exclude C
                    read_num=read_ind
                    if (row["contexttype"])==index:
                        total[index+str(read_num)]+=fre
                        if (fre>=read_num):
                            count[index+str(read_num)]+=fre
                        break


                if (row["contexttype"].find('C')!=-1):
                    total['C'+str(read_num)]+=fre
                    if(fre>=read_num):
                        count['C'+str(read_num)]+=fre


            #qc2
            if depth_cov_by_methylC.get(fre)==None:
                depth_cov_by_methylC[fre]=1
            else:
                depth_cov_by_methylC[fre]+=1
```

```
        for index in list_cytosine:#qc1
            for x in read_list:
                row_return['count'+index+str(x)]=count[index+str(x)]
                row_return['total'+index+str(x)]=total[index+str(x)]
        row_return["depth_cov_by_methylC"]=depth_cov_by_methylC#qc2
        row_return["ratio_frequency"]=ratio_frequency#qc3
        row_return["ratio_frequency_total"]=total_ratio#qc3
        row_return["context_percentage"]=context_percentage#dmp3
        row_return["stack_bar_data"]=stack_bar_data#stack_bar7

        result.append(row_return)
    end=time.time()
    end-start
    return result
```

Figure 4.1.9 Post processing of file

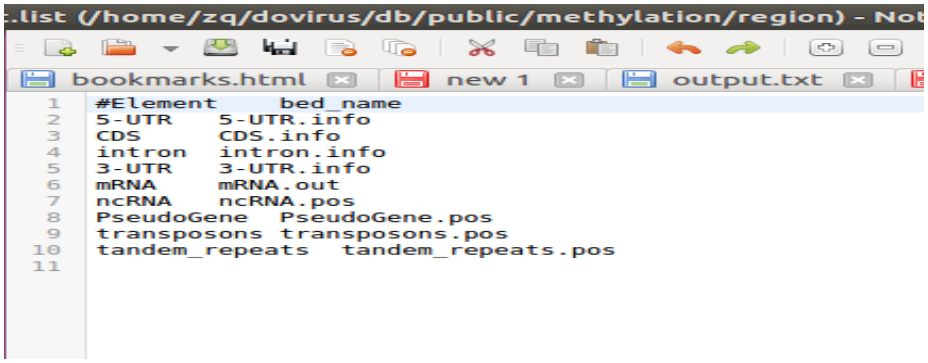


Figure 4.1.10 Element.list

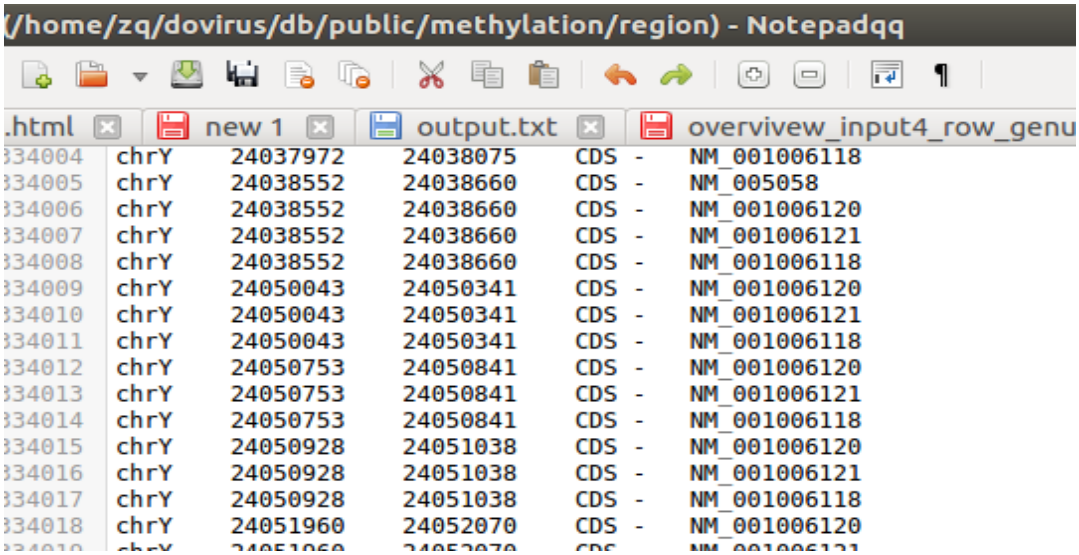


Figure 4.1.11 CDS.info

Data files generated in the processing step of data will be deleted when all related sample files are deleted. The data files will be deleted only when all the tabix files in a sample or gzip file are deleted. When tabix file is deleted, the context in tabix api is checked. If it is empty, the tabix model will be deleted.

```
def Tabixfile_cleanup(sender, instance, *args, **kwargs):

    if instance.post_process==True and instance.non_tabix==False:
        try:
            Tabix_obj=TabixAPI.objects.get(url_name=instance.sample_upload_tabix.url_name)
        except TabixAPI.DoesNotExist:
            return
        if len(Tabix_obj.alias)==0:
            Tabix_obj.delete()


    for field_name, _ in instance.__dict__.iteritems():
        field = getattr(instance, field_name)

        if issubclass(field.__class__, FieldFile) and field.name:
            if instance.post_process==True:
                #remove tabix file from os
                if os.path.exists(field.name+".tbi"):
                    os.remove(field.name+".tbi")
            field.delete(save=False)


post_delete.connect(Tabixfile_cleanup, sender=UploadFile)
```

Figure 4.1.12 Tabix model cleanup

SECTION 4.2 - PERCENTAGE OF CYTOSINE COVERED BY AT LEAST 2 READ IN (TABLE)

Table cannot be drawn using d3.js. Thus, TypeScript and DataTable are used to generate table that indicates the quality of data. The percentages of cytosine covered for each sample are read and used in table. The table is categorized into 2 parts which are data processing and data loading.

Data processing of the table will be started when user uploads the sample files. Effective cytosine count for C, CG, CHG and CHH will be summed up and divided by total number of effective count. Percentages of cytosine covered for C, CG, CHG and CHH are stored in the directory that contains file for sample. Effective cytosine count for 2 reads, 5 reads, 10 reads and 100 reads is calculated and saved as depth_cov{reads}.csv. If the effective count for 2 reads is generated, it will be stored into depth_cov2.csv. List_file[ind_list_file] in Figure 4.2.2 (a) represents the number of reads. Data is appended to file that stores data for Table 1.

```
#qc1
for read_ind in read_list:
    for index in list_cytosine:#exclude C
        read_num=read_ind
        if (row["contexttype"])==index:
            total[index+str(read_num)]+=fre
            if (fre>=read_num):
                count[index+str(read_num)]+=fre
            break


    if (row["contexttype"].find('C')!=-1):
        total['C'+str(read_num)]+=fre
        if(fre>=read_num):
            count['C'+str(read_num)]+=fre
```

Figure 4.2.1 Data processing for Table 1

```
TS visualizer.ts .../methylation_graph7_bar_chart   TS visualizer.ts .../methylation_graph5_pie   qc_methylation.djhtml
        file.closed

    #qc1
    for ind_list_file in range(0,len(list_file)):

        completeName = os.path.join(settings.BASE_DIR, 'db','public',
                        self.mount_point.fkey.name,file_name,
                        'depth_cov'+str(list_file[ind_list_file])+'.csv')

        with open(completeName,'ab+') as file:
            for ind_processed_data in processed_data:
                for name in ind_processed_data:
                    name=str(name)
                    all=string.maketrans('','')
                    nodigs=all.translate(all, string.digits)
                    int_name=name.translate(all, nodigs)
                    try:
                        int(list_file[ind_list_file])
                        compare_val=int(list_file[ind_list_file])
                        int_name=int(int_name)
                    except ValueError:
                        compare_val=list_file[ind_list_file]
                        int_name=int_name

                    if int_name==compare_val:
                        file.write(str(cnt_alias))
                        file.write('\t')
                        file.write(ind_processed_data['seg_name'])
                        file.write('\t')
                        file.write(name)
                        file.write('\t')
                        file.write(str(ind_processed_data[name]))
                        file.write('\n')

        file.closed
```

Figure 4.2.2 Save data processed

The data will be loaded into table through api. Web address used in data retrieving for first table is api/process/methylation?option=1&read=2.Parameter (read) in address will be changed according to user selection. Methylation in the address shows that the program will find the analysis data from the file named methylation. Figure 4.2.3(a) shows Q() helps to find the folder that contains 'methylation' in their folder names. Tabix Api objects that match qset query will be stored in `api` and `api` will be used to retrieve file name of the sample. Count and total of the coverage cytosine is

retrieved from depth_cov{reads}.csv for tabix samples and gzip samples. The percentages of coverage cytosine are calculated and returned. Figure 4.2.3 shows the results returned from data loading. Figure 4.2.3 (a) shows that the data is retrieved from option 1 with number of reads larger than 2. Percentage of coverage cytosine for each sample will be returned as request.

```
418
419  @login_required
420  def methylation_tabix_custom(request,url_name):
421
422      #change style----direct search from upload file-----get all processed file.
423      count=0
424      data_name=['C','CG','CHH','CHG']
425      data_sample=[]
426      flag_tabix=False
427      flag_upload=False
428
429      option = __safe_get(request, 'option')
430      read_num = __safe_get(request, 'read')
431
432
433      url_list=[]
434      your_search_query =url_name
435
436      qset = Q()
437
438      try:
439          for term in your_search_query.split('/'):
440              qset |= Q(folder__contains=term)
441              api = TabixAPI.objects.filter(qset)
442          for api_item in api:
443
444              if option=="1":
445                  url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
446                                  api_item.folder,
447                                  'depth_cov'+str(read_num)+'.csv'))
448              elif option=="2":
449                  url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
```

(a)

Figure 4.2.3 Reads and filter TABIX or UploadFile object based on qset.

```
zer.ts .../methylation_graph6_heatmap ●   api.py ●   manage.py   TS visualizer.ts .../methylation_graph3   TS visualizer.ts .../methylation_graph6_cube   TS visualize

if flag_tabix==True or flag_upload==True:
    for url_item in url_list:
        data_passback={}
        data_temp={}
        file = open(url_item, "r")
        count+=1
        if option=="1":
            for name in file:
                nameArr=name.split('\n')[0].split('\t')
                if data_temp.get(nameArr[2])==None:
                    data_temp[nameArr[2]]=nameArr[3]
                else:
                    data_temp[nameArr[2]]+=nameArr[3]

            for ind_data_name in data_name:
                    string_name1="count"+str(ind_data_name)+str(read_num)
                    string_name2="total"+str(ind_data_name)+str(read_num)
                    data_passback[ind_data_name]=round(int(data_temp[string_name1])*1.0/int(data_temp[string_name2])*100,2)
```

Figure 4.2.4 Data loading – reads data from file.

CHAPTER 4: RESULT DELIVERED

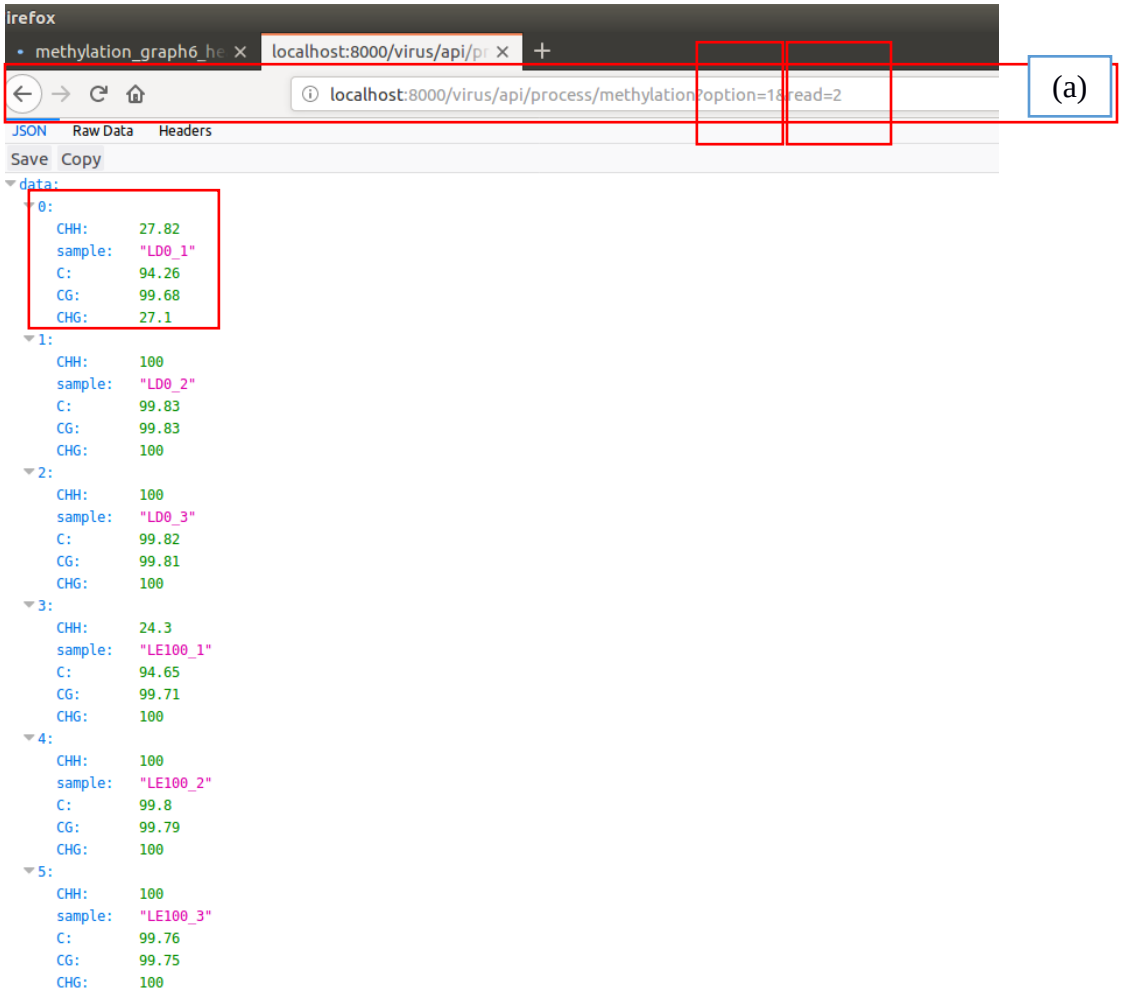


Figure 4.2.5 Data returned from process/methylation?option=1&read=2

Html template for the table is amended to remove memo and the reserved area for SVG drawing as Figure 4.2.6. Template for this table is named as qc_methylation.djhtml and stored under templates directory. Radio button for selection and table are drawn using html. File- data_tables_methyl.ts appends related rows into the table through ajax and DataTable. Number of reads will be retrieved as input to display related percentages of covered cytosine file for each sample. The table can be exported into excel or csv.

CHAPTER 4: RESULT DELIVERED

```
ualizer.ts .../methylation_graph3    TS visualizer.ts .../methylation_graph6_cube    TS visualizer.ts .../methylation_graph7_bar_chart    TS visualizer.ts .../methylation_graph5_
{% extends "virus/base.djhtml" %}

{% load static %}
{% load virus_extra %}

{% block head %}

{% endblock %}
{% block title %}Quality control{% endblock %}
{% block main %}


    <h1 >Depth Overview</h1>



            <div id="radio_button_read_num" >
                <ul>
                <input type="radio" checked="checked" class="read_num_radio" name="read_num" value=2>  2x 
                <input type="radio" class="read_num_radio" name="read_num"  value=5>  5x 
                <input type="radio" class="read_num_radio" name="read_num" value=10>  10x 
                <input type="radio" class="read_num_radio" name="read_num" value=100>  100x 
                </ul>
            </div>


    <div class="bvd3-table-container">

        <table id="qc-table" class="table table-hover" cellspacing="0" width="100%">
        <thead>
            <tr>
                <th>Sample</th>
                <th>C</th>
                <th>CG</th>
                <th>CHG</th>
                <th>CHH</th>
            </tr>
        </thead>
    </table>
```

Figure 4.2.6 Djhtml template for table 1

CHAPTER 4: RESULT DELIVERED

```
ualizer.ts .../methylation_graph6_cube    TS visualizer.ts .../methylation_graph7_bar_chart    TS visualizer.ts ..

$('input[name="read_num"]').on('change', function() {

    $("#qc-table").DataTable().ajax.reload();
 });

$("#qc-table").DataTable({
    dom: dtDomOption,
    scrollX: true,
    lengthMenu: lengthMenu,
    pageLength: 25,

    ajax: {
        type:"GET",
        url: "/virus/api/process/methylation?option=1",
        data: function ( d ) {
            d["read"]=$('input[name="read_num"]:checked').val()
        },

        dataSrc: "data",

        },
    columns:[
        { data: "sample", title: "Sample" },
        { data: "C", title: "C" },
        { data: "CG", title: "CG" },
        { data: "CHG", title: "CHG" },
        { data: "CHH", title: "CHH" }

    ],
    columnDefs: [ {
        "searchable": false,
        "orderable": false,
        "targets": 0
    } ],

    buttons: buttonsOption

});
```

Figure 4.2.7 Ajax loads data into Table 1

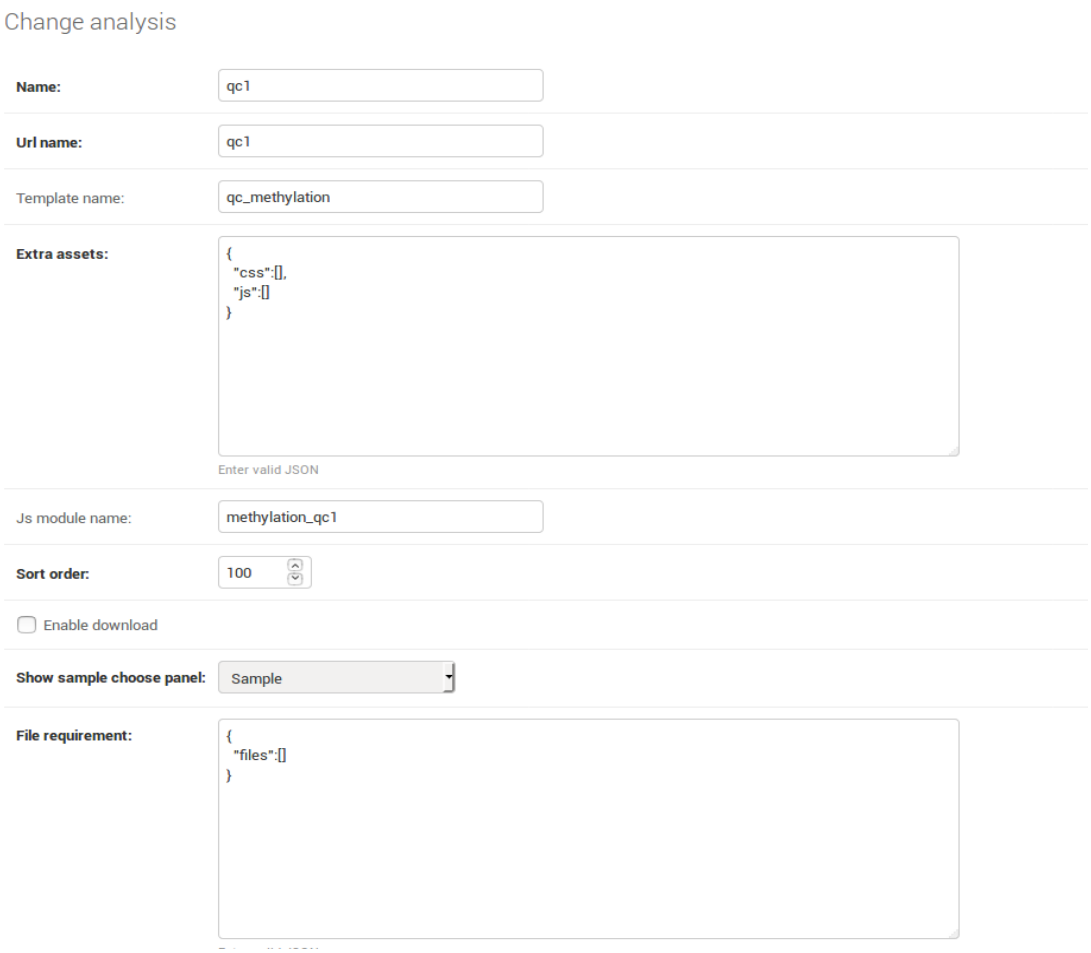


Figure 4.2.8 Setup of percentage of coverage cytosine's table

From Figure 4.2.9, user can find that CHH of LD0_1 and LE100_1 are abnormal. The percentage of coverage cytosine in CHH is lower than others. Thus, user can check the sample uploaded to identify the problem. The table helps user to control quality of the subsequent graph. 10x reads and 100x reads will return unsatisfactory result in the following samples due to the samples only have few rows in effective cytosine more than 10 or 100.

CHAPTER 4: RESULT DELIVERED

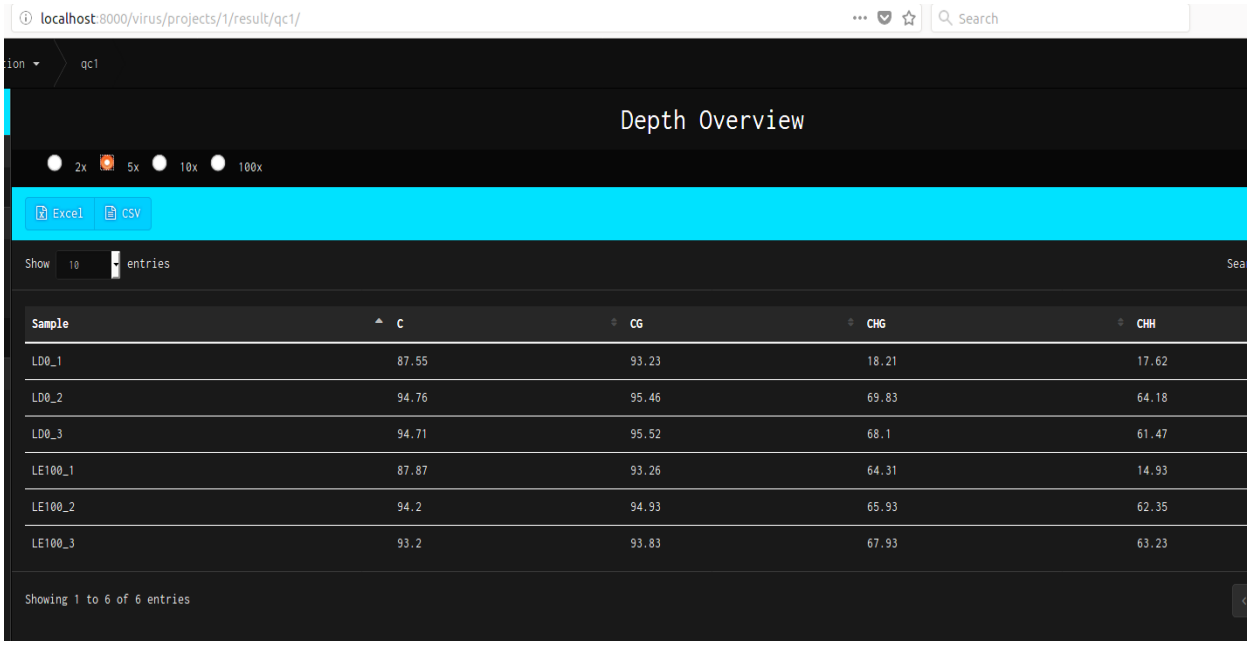


Figure 4.2.9 Output of percentage of coverage cytosine (5x reads).

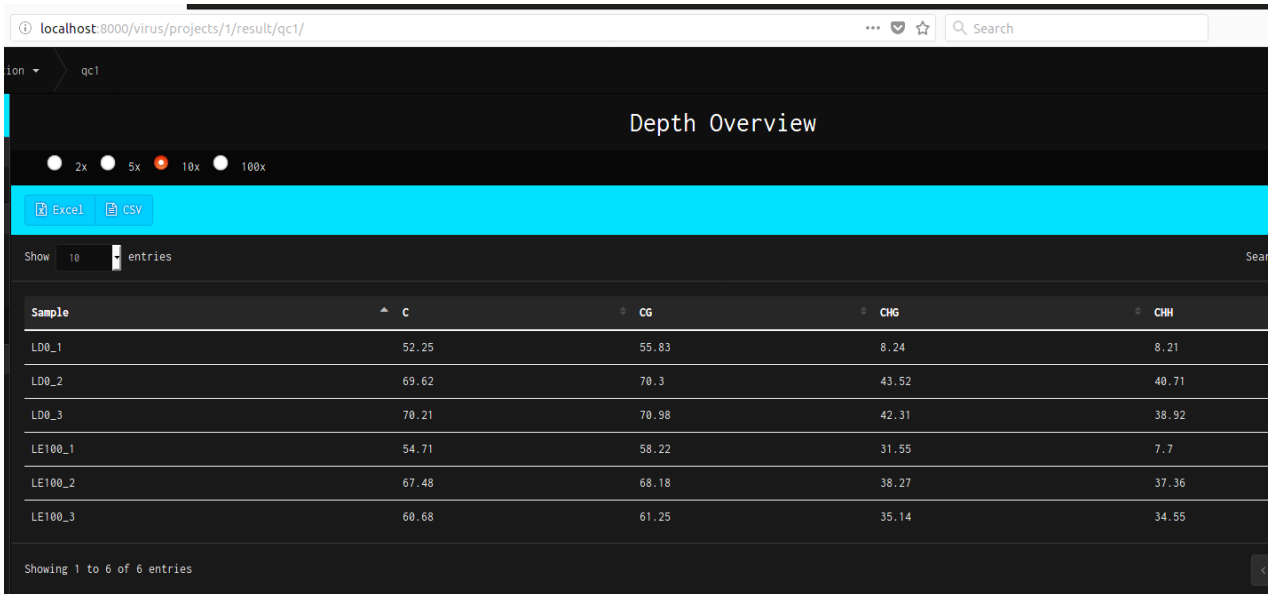


Figure 4.2.10 Output of percentage of coverage cytosine (10x reads).

SECTION 4.3- DISTRIBUTION OF THE COVERAGE DEPTH OF CYTOSINES (LINE GRAPH)

The graph that visualizes distribution of the coverage depth is drawn using TypeScript and d3 version 4. Figure 4.3.1 shows the distribution of the coverage depth of cytosine in a sample. Yellow line represents frequency and orange line represents accumulative. Methyl_C from the input is used to generate the graph. The graph is

separated into three parts which are data loading, data processing and visualization of the graph.

Data processing processes the data in mCytosine. Depth of the coverage cytosine is calculated by dividing total coverage cytosine of each read with total coverage cytosine. Count of the coverage cytosine for each read is stored into depth_coverageQc2.csv and depth of coverage cytosine will be calculated only when the data is loaded into visualization part.

```
#qc2
if depth_cov_by_methylC.get(fre)==None:
    depth_cov_by_methylC[fre]=1
else:
    depth_cov_by_methylC[fre]+=1
```

Figure 4.3.1 Data processing

```
#qc2                                    settings: settings
completeName = os.path.join(settings.BASE_DIR, 'db','public',
                            self.mount_point.fkey.name,file_name,
                            'depth_coverageQc2.csv')
with open(completeName,'ab+') as file:
    for ind_processed_data in processed_data:
        freData=ind_processed_data["depth_cov_by_methylC"]
        for ind_freData in freData:
            file.write(str(cnt_alias))
            file.write('\t')
            file.write(ind_processed_data['seg_name'])
            file.write('\t')
            file.write(str(ind_freData))
            file.write('\t')
            file.write(str(freData[ind_freData]))
            file.write('\n')
file.closed
```

Figure 4.3.2 Save data into file

Data loading loads data from a specific web address for graph. The graph retrieves data from api/process/methylation?option=2&read=0. Number of read is not important in the graph. It will not be processed in the data loading function. Thus, 0 is simply assigned to parameter (read). Code in Figure 4.3.3 will be run as request. For distribution of coverage depth of cytosine that visualized using line graph, depth_coverageQc2.csv for each sample will be read. The frequency is sorted by coverage reads as shown in Figure 4.3.4 (a). Item.getter(0) gets number of reads of the samples and sorts it. Figure 4.3.5 and Figure 4.3.6 show the result returned from the request in raw data form and JSON form.

manage.py | TS visualizer.ts .../methylation_graph3 | TS visualizer.ts .../methylation_graph6_cube | TS visualizer.ts .../methylation_graph7_bar_chart

```
                    url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                    api_item.folder,
                                    'methylation_percentage.csv'))
            flag_tabix=True
    except TabixAPI.DoesNotExist:
        flag_tabix=False

    try:

        api = UploadFile.objects.filter(non_tabix=True)
        for api_item in api:

            if option=="1":
                url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                    api_item.mount_point.fkey.name,api_item.sample_upload_zip,
                                    'depth_cov'+str(read_num)+'.csv'))
            elif option=="2":
                url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                    api_item.mount_point.fkey.name,api_item.sample_upload_zip,
                                    'depth_coverageQc2.csv'))
            elif option=="3":
                url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                    api_item.mount_point.fkey.name,api_item.sample_upload_zip,
                                    'depth_coverageQc3_'+str(read_num)+'.csv'))
            elif option=="4":
                url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                    api_item.mount_point.fkey.name,api_item.sample_upload_zip,
                                    'methylation_percentage.csv'))
            elif option=="7":
                url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                    api_item.mount_point.fkey.name,api_item.sample_upload_zip,
                                    'stack_bar_data.csv'))
            flag_upload=True
    except UploadFile.DoesNotExist:
        flag_upload=False

    if flag_tabix==True or flag_upload==True:
        for url_item in url_list:
            data_passback={}
```

(a)

Figure 4.3.3 Get the file for depth of coverage cytosine

```
manage.py   TS visualizer.ts .../methylation_graph3   TS visualizer.ts .../methylation_graph6_cube   TS visualizer.ts .../methylation_graph7_bar_chart   TS visualizer.ts

    data_temp={}
    file = open(url_item, "r")
    count+=1
    if option=="1":
        for name in file:
            nameArr=name.split('\n')[0].split('\t')
            if data_temp.get(nameArr[2])==None:
                data_temp[nameArr[2]]=nameArr[3]
            else:
                data_temp[nameArr[2]]+=nameArr[3]

        for ind_data_name in data_name:
                string_name1="count"+str(ind_data_name)+str(read_num)
                string_name2="total"+str(ind_data_name)+str(read_num)
                data_passback[ind_data_name]=round(int(data_temp[string_name1])*1.0/int(data_temp[string_name2])*100,2)

    elif option=="2":
        for name in file:
            nameArr=name.split('\n')[0].split('\t')
            if data_temp.get(nameArr[2])==None:
                data_temp[int(nameArr[2])]=int(nameArr[3])
        data_passback = dict(sorted(data_temp.items(), key=operator.itemgetter(0)))

    elif option=="3":
        data_name=['CG','CHH','CHG']
        for ind_data_name in data_name:
            data_temp[ind_data_name]={}
        for name in file:
            nameArr=name.split('\n')[0].split('\t')
            if nameArr[3]!="NA":
                if data_temp[nameArr[2]].get(nameArr[3])==None:
                    data_temp[nameArr[2]][float(nameArr[3])]=int(nameArr[4])
                else:
                    data_temp[nameArr[2]][float(nameArr[3])]=int(nameArr[4])
            elif nameArr[1]=="total":
                data_temp["total"]=int(nameArr[4])

        #sort fail?
        #for ind_data_name in data_name:

OUTPUT   DEBUG CONSOLE   TERMINAL                                                    1: bash
```

(a)

Figure 4.3.4 Read rows from file and sort it

CHAPTER 4: RESULT DELIVERED

Figure 4.3.5 Result returned from methylation?option=2&read=2-RawData

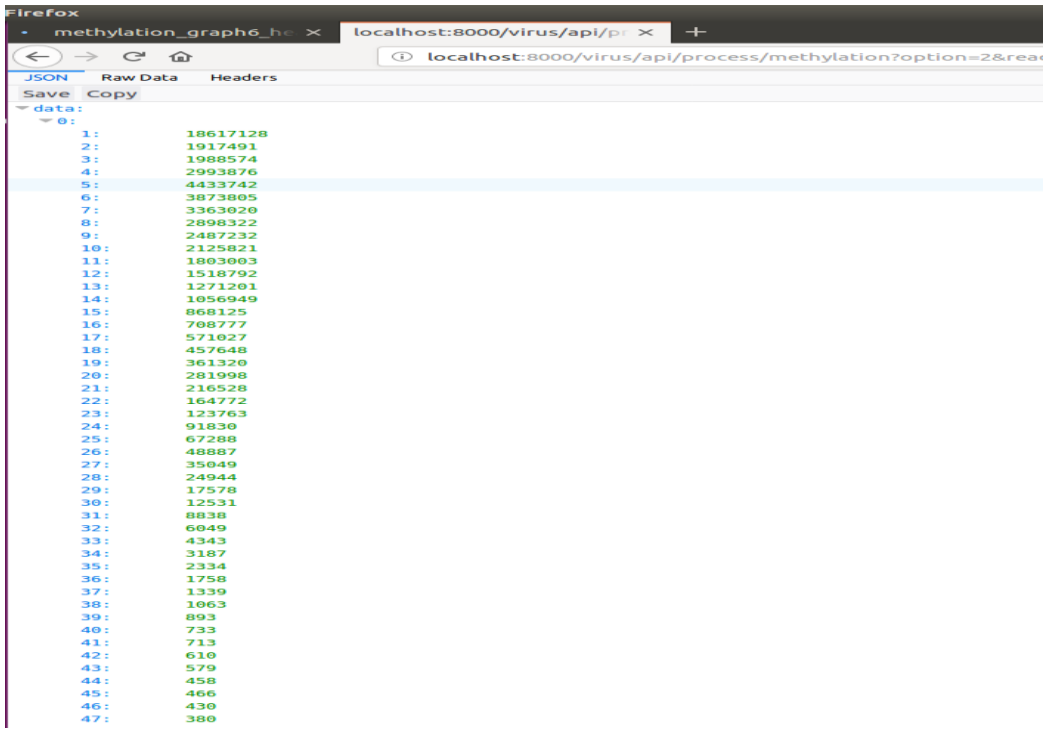


Figure 4.3.6 Result returned from methylation?option=2&read=2-JSON form

Data is processed after retrieving from the api. Frequency of coverage cytosine is calculated by dividing the count of each read with the total count in all effective cytosine

read in Figure 4.3.7. X-domain is reduced to cut off value that is too small in the graph. The x-domain cut off condition is any point that has more than 2 continuous round off value of zero. If the continuous count is smaller than 2, then counter that counts for zero will be reset. A frequency list without two continuous zero is retrieved. For instance, {0:1000,1:500,2:300,3:0,4:0.000001,5:0.000002} will be reduced into {0:1000,1:500,2:300}. Figure 4.3.9 shows how the x and y position of the graph being calculated. One row will display 3 distribution graphs. A margin will be added to x and y position of the graphs to avoid the graphs sticking together.

```
ethylation_graph7_bar_chart   TS visualizer.ts .../methylation_graph5_pie   qc_methylation.djhtml   TS controller.ts   models.py   # dmp7style.css

421
422         this.loadData("mCytosine", (d) => {
423
424
425             data_file = d
426
427
428             //Array
429             for (let j of data_file["data"]) {
430
431
432                 this.data.total = 0
433                 var sample_name
434                 this.data.yFreDataLocal = []
435                 this.data.yAccDataLocal = []
436
437                 sample_name = j["sample"]
438
439                 for (var test in j) {
440                     if (test != "sample" && test != "0") {
441                         this.data.yFreDataLocal[test] = parseInt(j[test])
442                         if (parseInt(j[test]) > 0) {
443                             this.data.total += parseInt(j[test])
444                         }
445                     }
446                 }
447                 //fix the position of 0 to avoid 0 too high
448                 for (var cnt = 0; cnt < this.data.yFreDataLocal.length; cnt++) {
449                     if (this.data.yFreDataLocal[cnt] > 0) {
450                         this.data.yFreDataLocal[cnt] /= this.data.total
451
452                     }
453                     else
454                         this.data.yFreDataLocal[cnt] = 0
455                 }
456
```

Figure 4.3.7 Data processing before drawing of graph- Calculate frequency and fix starting point

```
456
457             var zerocnt = 0
458             for (var cnt = 0; cnt < this.data.yFreDataLocal.length; cnt++) {
459                 console.log(Math.round(this.data.yFreDataLocal[cnt]*1000000)/1000000);
460                 if ((Math.round(this.data.yFreDataLocal[cnt]*1000000)/1000000) == 0) {
461                     zerocnt++
462
463                 }
464                 else {
465
466                     if (zerocnt < 2) {
467                         zerocnt = 0
468                     }
469                     else {
470                         this.data.yFreDataLocal = this.data.yFreDataLocal.slice(0, cnt-2)
471                         break
472                     }
473                 }
474             }
475
476             var val = 0, cnt = 0
477             for (var outcnt = 0; outcnt < this.data.yFreDataLocal.length; outcnt++) {
478
479
480                 val += this.data.yFreDataLocal[cnt++] * 100
481
482                 this.data.yAccDataLocal[outcnt] = val
```

Figure 4.3.8 Data processing before drawing of graph-Reduce domain of x

```
hylation_graph7_bar_chart   TS visualizer.ts .../methylation_graph5_pie   qc_methylation.djhtml   TS controller.ts   models.py   # dmp7style.css   TS visualizer.ts .../meth
477             for (var outcnt = 0; outcnt < this.data.yFreDataLocal.length; outcnt++) {
478
479
480                 val += this.data.yFreDataLocal[cnt++] * 100
481
482                 this.data.yAccDataLocal[outcnt] = val
483             }
484
485             this.data.DepGraph[this.data.count] = <any>{}
486             //need to perform deep copy----DONE
487             var ycount = Math.round((this.data.count - 1) / 3)
488             var xcount = this.data.count % 3
489             this.data.DepGraph[this.data.count] = (
490                 {
491                     //X,Y position incorrect ----DONE
492                     xGraph: (this.data.graphwidth + 80) * xcount, yGraph: (this.data.graphheight + 80) * ycount,
493                     yFreData: this.data.yFreDataLocal,
494                     yAccData: this.data.yAccDataLocal,
495                     sampleName: sample_name
496
497                 })
498             this.data.DepGraph[this.data.count].yFreData = this.data.yFreDataLocal.map(function (d1) { return d1 })
499             this.data.DepGraph[this.data.count].yAccData = this.data.yAccDataLocal.map(function (d1) { return d1 })
500
501             this.data.count++
502         }
503
504
505
506     })
```

Figure 4.3.9 Store the processed data into DepGraph for graph visualization

Each sample will display their own distribution graphs. For example, if there are 3 sample files uploaded, there will be 3 graphs being added to the layer. Thus, the x-position, y-position and data for each graph will be calculated and stored for visualization process. A for loop is used to append all the graphs for samples into a svg as shown in Figure 4.3.10(a). Domain of y axis is retrieved via d3.extent that returns an array of minimum and maximum values of the data as shown in Figure 4.3.10 (b). Part c from Figure 4.3.10 initializes axis, range and position of x and y. Figure 4.3.11 (a) called axis

that initialized in Figure 4.3.13. Figure 4.3.11 (b) append “g” to svg to group all the item added to depthline together. Figure 4.3.11 (c) and Figure 4.3.12 (a) draw path to the graph. Fill of path in Figure 4.3.12 (a) is set to none to avoid the path drawn has an enclosed area. Figure 4.3.13 changes the height of the svg based on number of rows that is needed to draw all the graphs for samples. Frequency of coverage depth is presented in orange line while accumulative of coverage depth is presented in yellow line as shown in Figure 4.3.14.

```
150     this.addLayer('Dep', ['depth']).renderer(function ($size: Size, $data: Data, $op: Option, vz: Visualizer) {
151
152         for (var graphCnt = 0; graphCnt < $data.count; graphCnt++) {
153             var yFre = $data.DepGraph[graphCnt].yFreData
154             /* this.appendRect("cdep-bg", $data.DepGraph[graphCnt].xGraph, $data.DepGraph[graphCnt].yGraph, width, height)
155                 .attr("fill", '#fff')*/
156             if (($data.DepGraph[graphCnt].yFreData.length) == null)
157                 $data.DepGraph[graphCnt].yFreData.length = 0
158             let scaleDepX = d3.scaleLinear()
159                 .domain([0, $data.DepGraph[graphCnt].yFreData.length])
160                 .range([0, $data.graphwidth]);
161             let axisVdepX = d3.axisBottom(scaleDepX);
162             let vDep_axisX = this.append("g")
163                 .attr("class", "axis axis-x")
164                 .attr("transform", tool.translate($data.DepGraph[graphCnt].xGraph, $data.graphheight + $data.DepGraph[graphCnt].yGraph))
165                 .call(axisVdepX)
166             var max
167             if ($data.DepGraph[graphCnt].yFreData.length > 1)
168                 max = Math.ceil($data.DepGraph[graphCnt].yFreData.reduce(function (a, b) {
169                     return Math.max(a, b);
170                 }) * 10) / 10
171             else
172                 max = 1
173             let scaleDepY = d3.scaleLinear()
174                 .domain(d3.extent($data.DepGraph[graphCnt].yFreData, function (d) {   return d; }))
175                 .range([$data.graphheight, 0])
176
177
178             let axisDepY = d3.axisLeft(scaleDepY).tickValues(scaleDepY.ticks(5))
179
180
181             let dep_axisY = this.append("g")
182                 .attr("class", "axis axis-y")
183                 .attr("transform", tool.translate($data.DepGraph[graphCnt].xGraph, $data.DepGraph[graphCnt].yGraph))
184                 .call(axisDepY);
185
186             let scaleAccDepY = d3.scaleLinear()
187                 .domain([100, 0])
```

(a)

(b)

(c)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash

Figure 4.3.10 Visualization- Define axis

```
let scaleAccDepY = d3.scaleLinear()
    .domain([100, 0])
    .range([0, $data.graphheight])

let axisThirdDepY = d3.axisRight(scaleAccDepY).tickValues(scaleAccDepY.ticks(5))
let dep_thirdAxisY = this.append("g")
    .attr("class", "axis axis-y third")
    .attr("transform", tool.translate($data.graphwidth + $data.DepGraph[graphCnt].xGraph, $data.DepGraph[graphCnt].yGraph))
    .call(axisThirdDepY);

let depthline = this.append("g")
    .attr("class", "line")
    .attr("id", graphCnt)
    .attr("transform", tool.translate($data.DepGraph[graphCnt].xGraph, $data.DepGraph[graphCnt].yGraph))

this.selectAll(".axis")
    .selectAll("text")
    .style("font-size", "14px");
//line for others how to show on correct pane ----DONE
var line = d3.line<number>()
    .x(function (d, i) { return scaleDepX(i); }) // set the x values for the line generator
    .y(function (d) {
        return scaleDepY(d);
    })
    .curve(d3.curveCatmullRom.alpha(0.5));

var line2 = d3.line<number>()
    .x(function (d, i) { return scaleDepX(i); })
    .y(function (d) {

        return scaleAccDepY(d);
    })
    .curve(d3.curveCatmullRom.alpha(0.5));
```

(a)

(b)

(c)

Figure 4.3.11 Visualization- Call axis and draw line

```
var pathFre = depthline.append("path")
    .datum($data.DepGraph[graphCnt].yFreData)
    .attr("class", "line_Fre")
    .attr("stroke", '#fff7bc')
    .style("stroke-width", "2px")
    .style("fill", "none")
    .attr("d", line)

var divFre = d3.select("body").append("div")
    .attr("class", "Fretooltip")
    .style("opacity", 0);

var pathAcc = depthline.append("path")
    .datum($data.DepGraph[graphCnt].yAccData)
    .attr("class", "line_Acc")
    .attr("stroke", '#d95f0e')
    .style("stroke-width", "2px")
    .style("fill", "none")
    .attr("d", line2)

this.append("text")
    .attr("text-anchor", "middle")
    .attr("transform", "translate(" + ($data.DepGraph[graphCnt].xGraph + 20) + "," + ($data.graphheight / 2 + $data.DepGraph[graph
    .text("Frequency(%)");

this.append("text")
    .attr("text-anchor", "middle")
    .attr("transform", "translate(" + ($data.graphwidth + $data.DepGraph[graphCnt].xGraph - 10) + "," + ($data.graphheight / 2 + $
    .text("Accumulative(%)");

this.append("text")
    .attr("text-anchor", "middle")
    .attr("transform", "translate(" + ($data.graphwidth / 2 + $data.DepGraph[graphCnt].xGraph) + "," + ($data.graphheight - 10 + $
    .text("Depth");

this.append("text")
    .attr("text-anchor", "middle")  // this makes it easy to centre the text as the transform is applied to the anchor
```

(a)

Figure 4.3.12 Visualization- Draw line and text

```
ar_chart   TS visualizer.ts .../methylation_graph5_pie   qc_methylation.djhtml   TS controller.ts   models.py   # dmp7style.css   TS visualizer.ts

        }
        vz.svg.attr("height", $data.graphheight * Math.floor(($data.count - 1) / 3) + $data.graphheight+250)

    })
```

Figure 4.3.13 Visualization- resize the svg for line graphs.

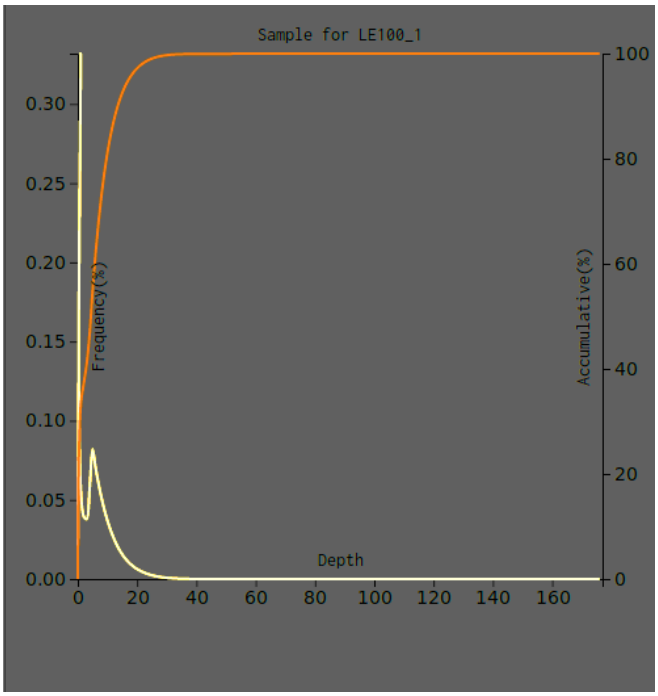


Figure 4.3.14 Distribution of coverage of depth of cytosine for sample LE100_1

Interaction of the graph includes hover and click. When user hovers the line, circles will appear on the frequency and accumulative line (Figure 4.3.15 -a). Detail information like Figure 4.3.17 will be displayed. A non-visible rectangle is appended to each graph to detect the position of the mouse (Figure 4.3.15 -b). It is better than directly applying mouse move on the line. This is because if mouse move is applied on line, circles are hard to appear at two different lines when user hovers. Thus, event on mouse over and on mouse move is applied on the rectangle. A div that contains detail information and circle of the line graph will be shown when the mouse moves into the rectangle. Figure 4.3.15 (b) and Figure 4.3.16 (a) retrieve detail information and show in div (Figure 4.3.17). If one of the line in the line graph is hidden, the detail information of the graph will not be shown in the hover box. Parameters, `clickfreparam` and `clickaccparam` indicate the visibility of the frequency line and accumulative line.

```
thylation_graph7_bar_chart   TS visualizer.ts .../methylation_graph5_pie   qc_methylation.djhtml   TS controller.ts   models.py   # dmp7style.css   TS

268
269         var mousePerLine = mouseG.selectAll('.mouse-per-line')
270             .data($data.DepGraph[graphCnt].yFreData)
271             .enter()
272             .append("g")
273             .attr("class", "mouse-per-line");
274
275         mouseG.append("circle")
276             .attr("r", 7)
277             .attr("class", "circleFre")
278             .style("stroke", "none")
279             .style("fill", "none")
280             .style("stroke-width", "1px")
281             .style("opacity", "0");
282
283         mouseG.append("circle")
284             .attr("r", 7)
285             .attr("class", "circleAcc")
286             .style("stroke", "none")
287             .style("fill", "none")
288             .style("stroke-width", "1px")
289             .style("opacity", "0");
291         mouseG.append('svg:rect') // append a rect to catch mouse movements on canvas
292             .attr("id", "rect")
293             .attr('width', $data.graphwidth)
294             .attr('height', $data.graphheight)
295             .attr('fill', 'none')
296             .attr('pointer-events', 'all')
297             .on('mouseout', function () { // on mouse out hide circles and text
298                 d3.selectAll(".mouse-line")
299                     .style("opacity", "0");
300                 d3.selectAll(".mouse-over-effects circle")
301                     .style("opacity", "0");
302                 divFre.transition()
303                     .duration(500)
304                     .style("opacity", 0);
305             })
```

(a)

(b)

Figure 4.3.15Interaction-draw circle and rectangle to detect movement

CHAPTER 4: RESULT DELIVERED

```
lation_graph7_bar_chart   TS visualizer.ts .../methylation_graph5_pie   qc_methylation.djhtml   TS controller.ts   models.py   # dmp7style.css   TS visualizer.ts .../methylation_graph7

06
07          .on('mousemove', function (this: any) { // mouse moving over canvas
08
09              var mouse = d3.mouse(this);
10              var xmouse = Math.floor(scaleDepX.invert(mouse[0]))
11              //check if the dot in x range for three line//dONE
12              //text display for three line in one dialog box---done
13              //multiple graph hover function
14
15              var text_html = ""
16              var heightcount = 0
17              var d = "M" + mouse[0] + "," + $data.graphheight;
18              d += " " + mouse[0] + "." + 0:
19
20
21
22              var linne_mouse = d3.selectAll(".mouse-line")
23              d3.select(linne_mouse["_groups"][0][depthline.attr("id")])
24                  .attr("d", function () {
25                      return d;
26                  })
27
28                  .style("opacity", "1.0");
29              if ($data.DepGraph[depthline.attr("id")]["yFreData"].hasOwnProperty(xmouse)&&$data.clickfreparam==false) {
30
31                  var circle_hover = d3.selectAll(".mouse-over-effects .circleFre")
32                  d3.select(circle_hover["_groups"][0][depthline.attr("id")])
33                      .attr("transform", tool.translate(mouse[0],
34                          scaleDepY($data.DepGraph[depthline.attr("id")]["yFreData"][xmouse]) - 5)).style("opacity", "0.9")
35                      .transition()
36                      .duration(200)
37                      .style("fill", "none")
38                      .style("stroke", "#fff7bc")
39                      .style("stroke-width", 1.5)
40                      .attr("r", 4)
41
42                  text_html = "Depth: " + xmouse + "<br/>Frequency: " + $data.DepGraph[depthline.attr("id")]["yFreData"][xmouse].toFixed
43
```

(a)

(b)

Figure 4.3.16 Retrieve information for frequency line

```
345
346                 }
347             if ($data.DepGraph[depthline.attr("id")]["yAccData"].hasOwnProperty(xmouse)&&$data.clickaccparam==false) {
348
349                     var circle_hover = d3.selectAll(".mouse-over-effects .circleAcc")
350                     d3.select(circle_hover["_groups"][0][depthline.attr("id")])
351                         .attr("transform", tool.translate(mouse[0],
352                             scaleAccDepY($data.DepGraph[depthline.attr("id")]["yAccData"][xmouse]) - 5)).style("opacity", "0.9")
353                         .transition()
354                         .duration(200)
355                         .style("fill", "none")
356                         .style("stroke", "#d95f0e")
357                         .style("stroke-width", 1.5)
358                         .attr("r", 4)
359                     if(text_html!="")
360                         text_html += "<br/>Accumulative: " + $data.DepGraph[depthline.attr("id")]["yAccData"][xmouse].toFixed(8)
361                     else
362                         text_html = "Depth: " + xmouse + "<br/>Accumulative: " + $data.DepGraph[depthline.attr("id")]["yAccData"][xmouse
363
364
365                 }
366                 if (text_html != "") {
367                     divFre.transition()
368                         .duration(200)
369                         .style("opacity", .9);
370                     divFre.html(text_html)
371                         .style("text-align", "left")
372                         .style("left", (d3.event.pageX + 10) + "px")
373                         .style("top", (d3.event.pageY - 28) + "px");
374                 }
375                 //check for 2 data
376             });
377
378
```

(a)

(b)

Figure 4.3.17 Retrieve information for accumulative line

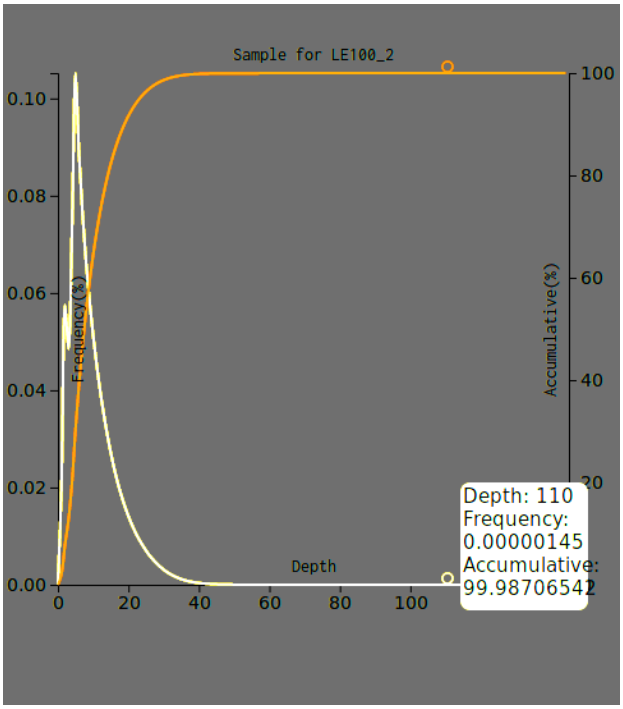


Figure 4.3.18 Distribution graph with hover box

When the graph cannot be viewed due to two graphs stack together, user can click on the legend to hide another graph and double click to show the graph. To show only frequency line graph, click on the yellow legend to hide the orange accumulative line and double click to show it as in Figure 4.3.20. If only orange accumulative line graph needs to be shown on the graph, orange legend needs to be clicked to hide yellow frequency line as in Figure 4.3.21. Complete distribution of coverage cytosine that visualizes 6 uploaded samples is shown in Figure 4.3.22.

CHAPTER 4: RESULT DELIVERED

```
thylation_graph7_bar_chart   TS visualizer.ts .../methylation_graph5_pie   qc_methylation.djhtml   TS controller.ts   models.py   # dmp7style.css   TS vi
103         this.addLayer("legend", ["legend"]).renderer(function ($size: Size, $data: Data, $op: Option, vz: Visualizer) {
104             //make color flexible---WAITING
105             let types = [["accumulative", "Accumulative", "#d95f0e"], ["frequency", "Frequency", "#fff7bc"]]
106
107             let legend = this.selectAll("g")
108                 .data(types)
109                 .enter().append("g")
110                 .attr("transform", (d, i) => tool.translate(0, i * 24 + 10));
111
112             legend.append("rect")
113                 .attr("x", -18)
114                 .attr("width", 18)
115                 .attr("height", 18)
116                 .attr("fill", d => d[2])
117                 .attr("class", d => d[0])
118                 .on("click", function (d, i) {
119                     if (d[2] === "#fff7bc") {
120                         d3.selectAll(".line_Acc").style("visibility", "hidden");
121                         d3.selectAll(".dotAcc").style("visibility", "hidden");
122                         $data.clickaccparam = true
123                     }
124                     else if (d[2] === "#d95f0e") {
125                         d3.selectAll(".line_Fre").style("visibility", "hidden");
126                         d3.selectAll(".dotFre").style("visibility", "hidden");
127                         $data.clickfreparam = true
128                     }
129                 })
130                 .on("dblclick", function (d) {
131                     if (d[2] === "#fff7bc") {
132                         d3.selectAll(".line_Acc").style("visibility", "visible");
133                         d3.selectAll(".dotAcc").style("visibility", "visible");
134                         $data.clickaccparam = false
135                     }
136                     else if (d[2] === "#d95f0e") {
137                         d3.selectAll(".line_Fre").style("visibility", "visible");
138                         d3.selectAll(".dotFre").style("visibility", "visible");
139                         $data.clickfreparam = false
140                     }
141                 });
```

Figure 4.3.19 Interaction-click

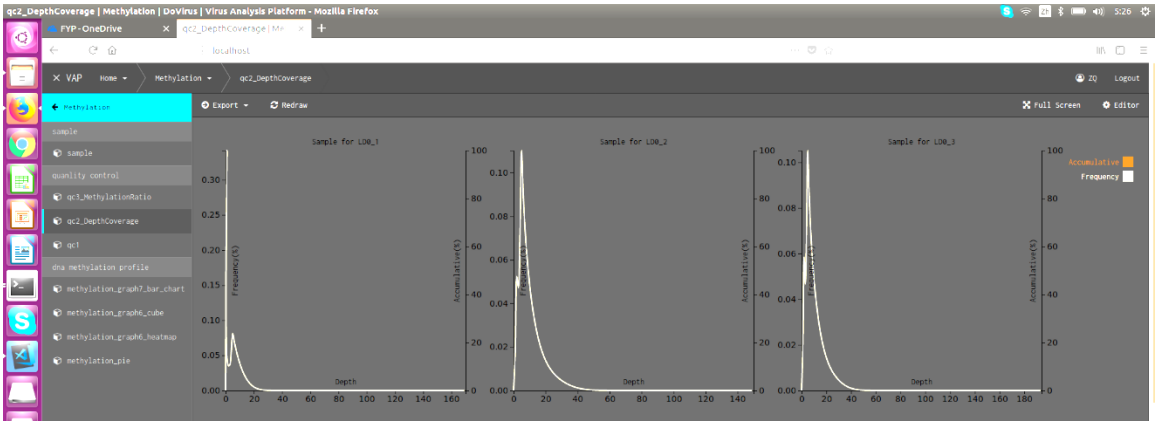


Figure 4.3.20 Frequency line graph

CHAPTER 4: RESULT DELIVERED

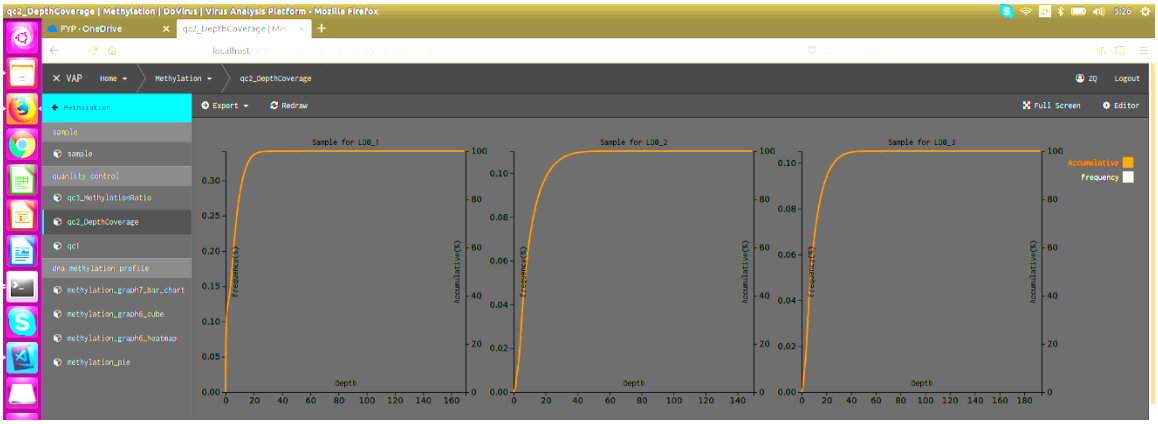


Figure 4.3.21 Accumulative line graph

CHAPTER 4: RESULT DELIVERED

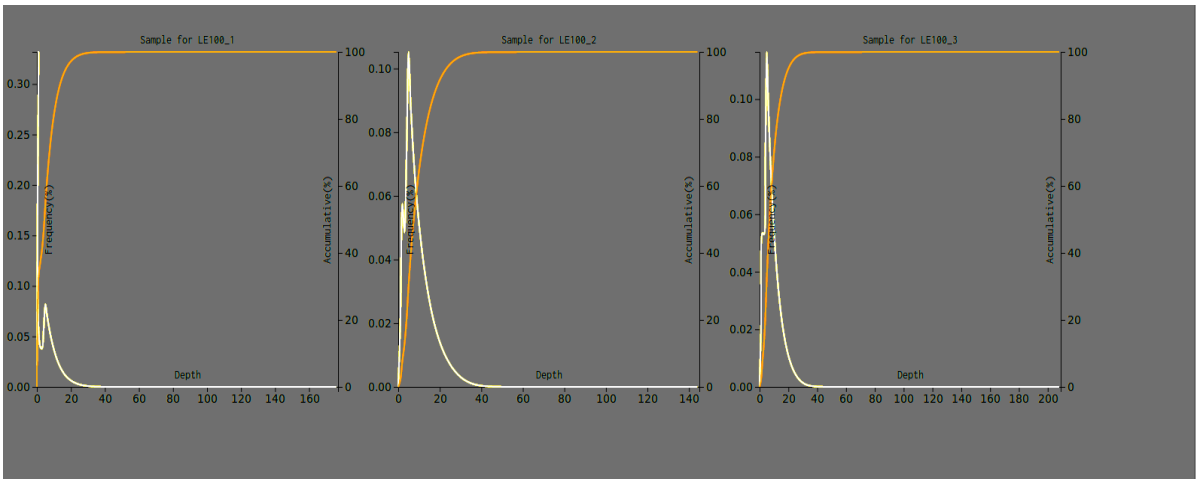
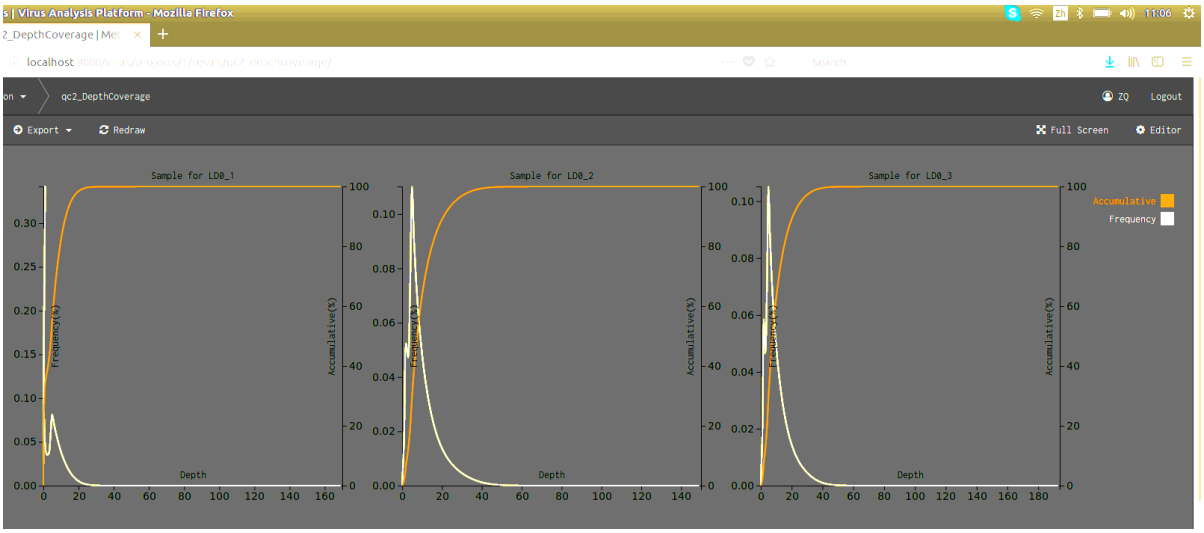


Figure 4.3.22 Example of complete visualization for distribution of depth coverage in cytosine



SECTION 4.4- DISTRIBUTION OF THE METHYLATION LEVEL IN MC, MCHH, MCHG (LINE GRAPH)

The graph shows the distribution of the methylation level in different contexts. Implementation of the graph is separated into three parts which are data processing, data loading and visualization of the graph as the previous graph.

Data processing processes the data in mCytosine. There are 3 sets of data for y-axis: fraction of total methylated-cytosine for methylated-C, methylated-CHG and methylated-CHH. The methylation ratio of each context is counted and divided by total methylated cytosine in the chromosome. The data will be processed and saved in depth_coverageQc3_{reads}.csv. The count for each methylation ratio and total count of methylation ratio will be retrieved.

```
#qc3
for index in list_cytosine_3:#exclude C #maybe need to directly calculate total
    if (row["contexttype"])==index:
        if row["ratio"]!="NA":
            total_ratio+=1
            for ratio_fre_Cnt in range(0,10):
                if ratio_frequency[ratio_fre_Cnt][index].get(row["ratio"])==None:

                    if fre>=ratio_fre_Cnt:
                        ratio_frequency[ratio_fre_Cnt][index][row["ratio"]]=1
                else:
                    if fre>=ratio_fre_Cnt:
                        ratio_frequency[ratio_fre_Cnt][index][row["ratio"]]+=1
```

Figure 4.4.1 Data processing- get counts for methylation ratio of context

```
#qc3
for int_name in range(0,10):
    completeName = os.path.join(settings.BASE_DIR, 'db','public',
                    self.mount_point.fkey.name,file_name,
                    'depth_coverageQc3_'+str(int_name)+'.csv')
    with open(completeName,'ab+') as file:

            file.write(str(cnt_alias))
            file.write('\t')
            file.write("total")
            file.write('\t')
            file.write("NA")
            file.write('\t')
            file.write("NA")
            file.write('\t')
            file.write(str(ind_processed_data["ratio_frequency_total"]))
            file.write('\n')

            freData=ind_processed_data["ratio_frequency"]
            for ind_freData in freData[int_name]:

                for dataind_freData in freData[int_name][ind_freData]:
                    file.write(str(cnt_alias))
                    file.write('\t')
                    file.write(ind_processed_data['seg_name'])
                    file.write('\t')
                    file.write(str(ind_freData))
                    file.write('\t')
                    file.write(str(dataind_freData))
                    file.write('\t')
                    file.write(str(freData[int_name][ind_freData][str(dataind_freData)]))
                    file.write('\n')
    file.closed
```

Figure 4.4.2 Save data processed into specific file.

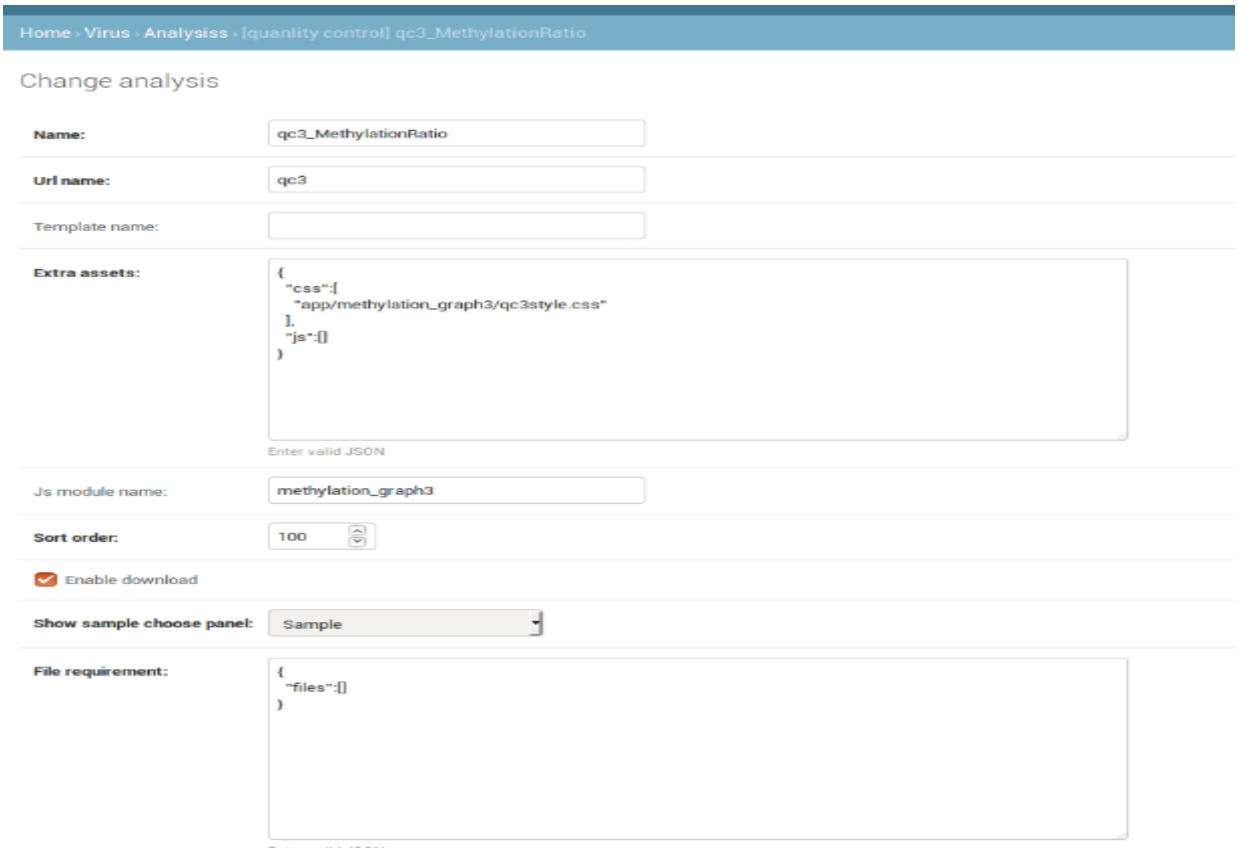


Figure 4.4.3 Setup of distribution of methylation ratio

Data loading loads data from api/process/methylation?option=3&read={reads} for graph. User can choose number of reads range from 0 to 9 from the editor side bar. Default reads for the graph is 0. If number of reads is 0, it means all counts in methylation ratio are included. If number of reads is 1, it means only methylation ratio that consists of effective cytosine coverage larger than 1 will be included. Thus, the graph will be drawn based on the data retrieved from api/process/methylation?option=3&read={reads} at the beginning. The count that received from the address will be divided by total number of count to form fraction of total methylated cytosine at specific level of methylation ratio.

CHAPTER 4: RESULT DELIVERED

```
api.py ●   manage.py   TS visualizer.ts .../methylation_graph3   TS visualizer.ts .../methylation_graph6_cube   TS visualizer.ts .../
                                                    api_item.folder,
                                                    'methylation_percentage.csv'))
                elif option=="7":
                    url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                                    api_item.folder,
                                                    'stack_bar_data.csv'))
                flag_tabix=True
        except TabixAPI.DoesNotExist:
            flag_tabix=False

        try:

            api = UploadFile.objects.filter(non_tabix=True)
            for api_item in api:

                if option=="1":
                    url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                                    api_item.mount_point.fkey.name,api_item.sample_upload_zip,
                                                    'depth_cov'+str(read_num)+'.csv'))
                elif option=="2":
                    url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                                    api_item.mount_point.fkey.name,api_item.sample_upload_zip,
                                                    'depth_coverageQc2.csv'))
                elif option=="3":
                    url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                                    api_item.mount_point.fkey.name,api_item.sample_upload_zip,
                                                    'depth_coverageQc3 '+str(read_num)+'.csv'))
                elif option=="4":
                    url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                                    api_item.mount_point.fkey.name,api_item.sample_upload_zip,
                                                    'methylation_percentage.csv'))
                elif option=="7":
                    url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                                    api_item.mount_point.fkey.name,api_item.sample_upload_zip,
                                                    'stack_bar_data.csv'))
                flag_upload=True
        except UploadFile.DoesNotExist:
            flag_upload=False
```

Figure 4.4.4 Data loading-retrieve data as request

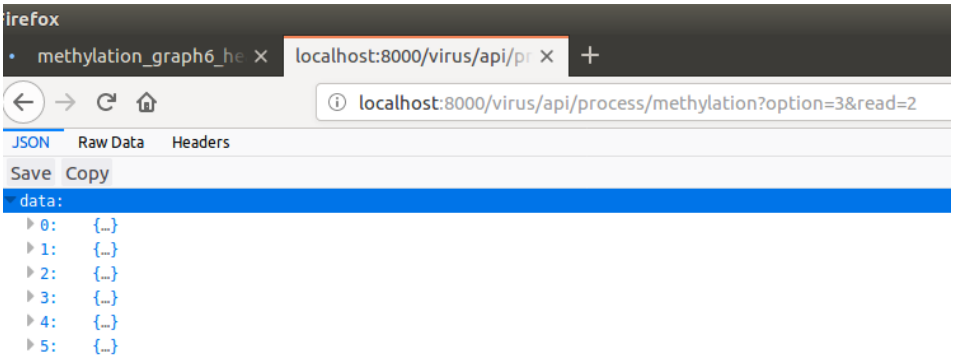


Figure 4.4.5 Data loading-data returned for each sample

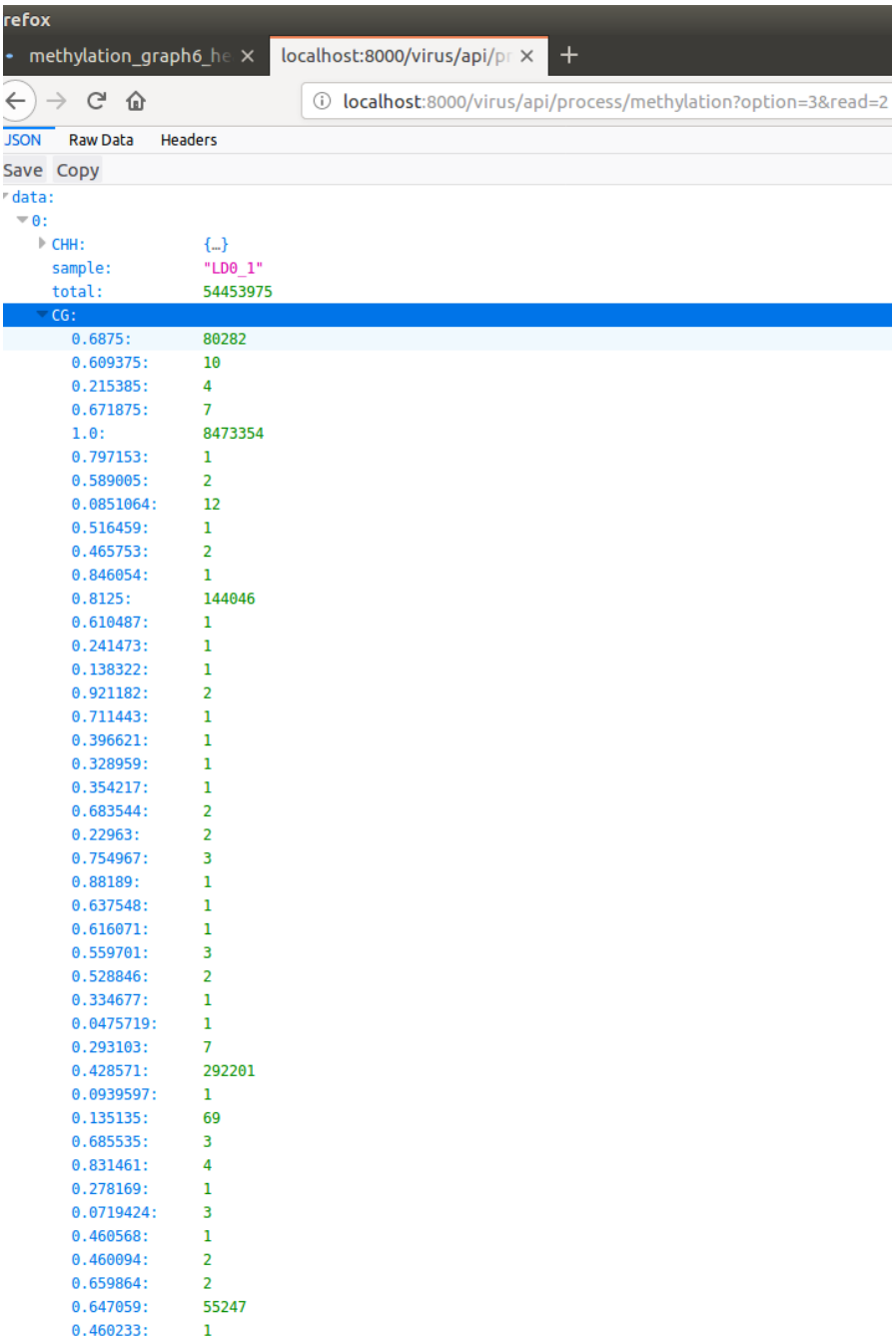


Figure 4.4.6 Data retrieved for first sample.

The visualization and interaction of the graph are similar to the previous line graph. Lines for CG, CHG and CHH are added to the graph with same method as previous line graph. Interaction added for this graph are hover and click. User can hover on the graph to view the detail information of the point. A transparent rectangle is placed in each graph to detect the movement of mouse. When the mouse moves within the

rectangle, the hover box will display related information. If only two lines have data on the point, then information for the point for those two lines will be displayed. For instance, if C has no data on x=10 while A and B have a value of 10 and 2 at x=10, only information for A and B will be displayed. When the graph cannot be viewed due to three graphs stack together, user can click on the legend to hide another 2 graphs and click on reset legend to show the graph.

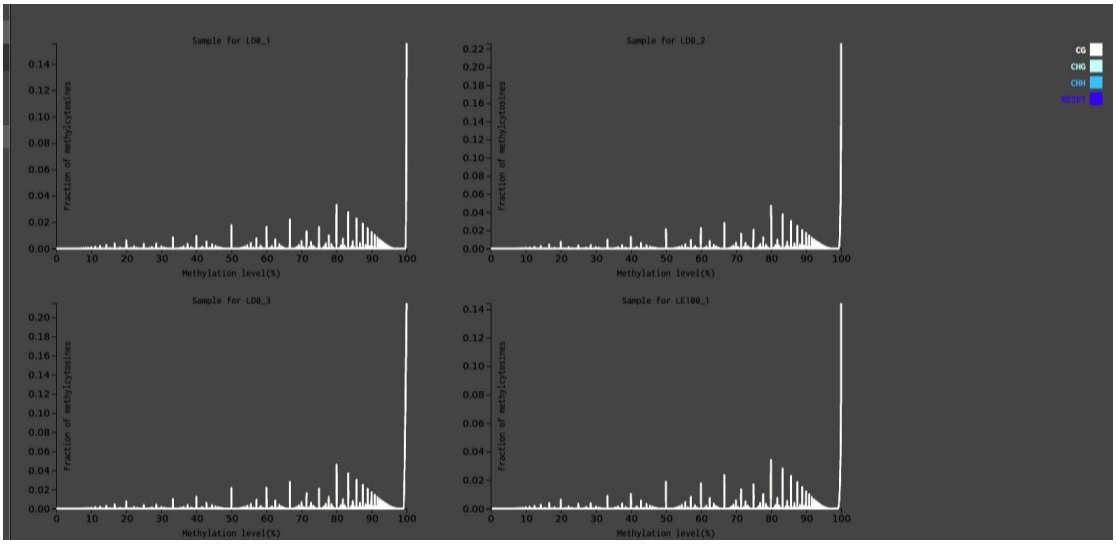


Figure 4.4.7 Distribution of CG after clicked on CG legend

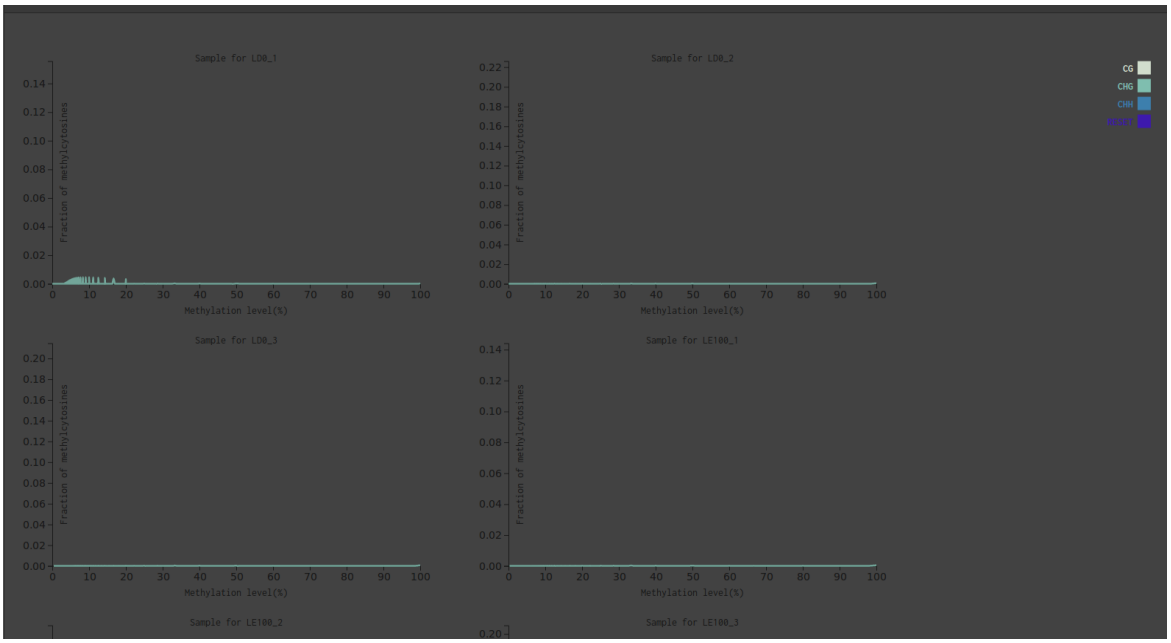


Figure 4.4.8Distribution of CHG after clicked on CHG legend

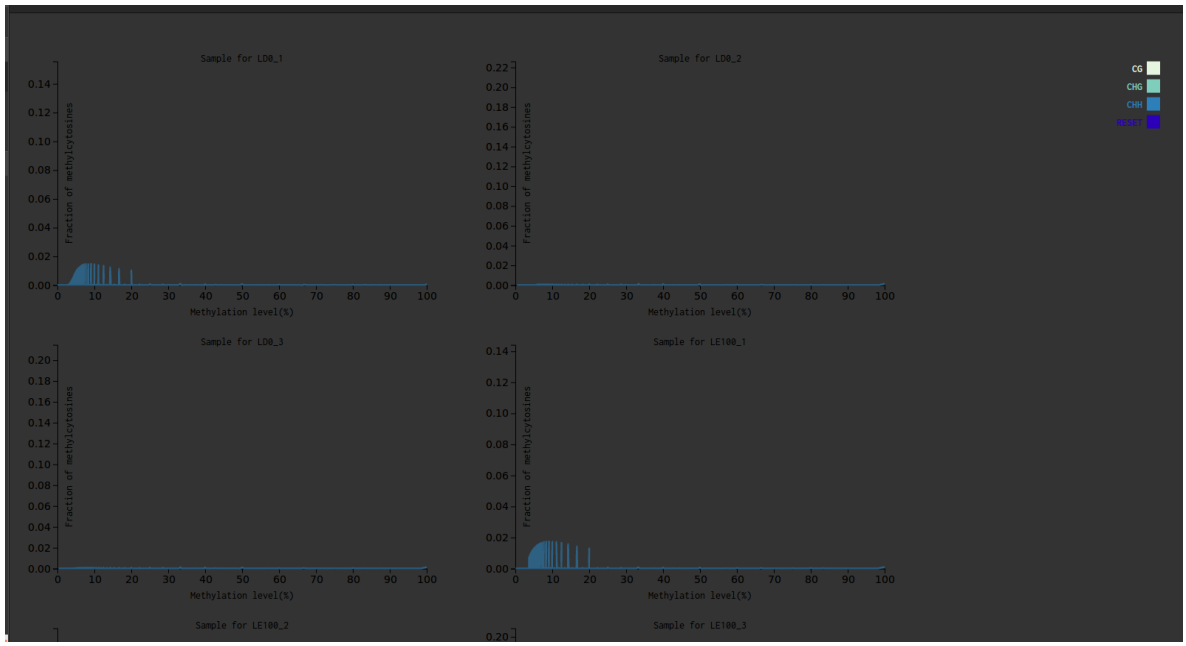


Figure 4.4.9 Distribution of CHH after clicked on CHH legend

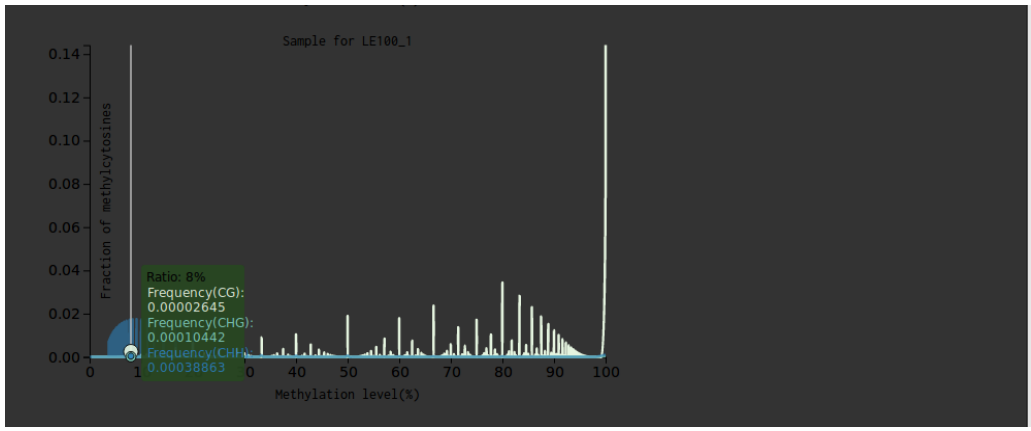


Figure 4.4.10 Hover box that show extra details based on CG, CHG and CHH. The additional detail displayed based on the color of context.

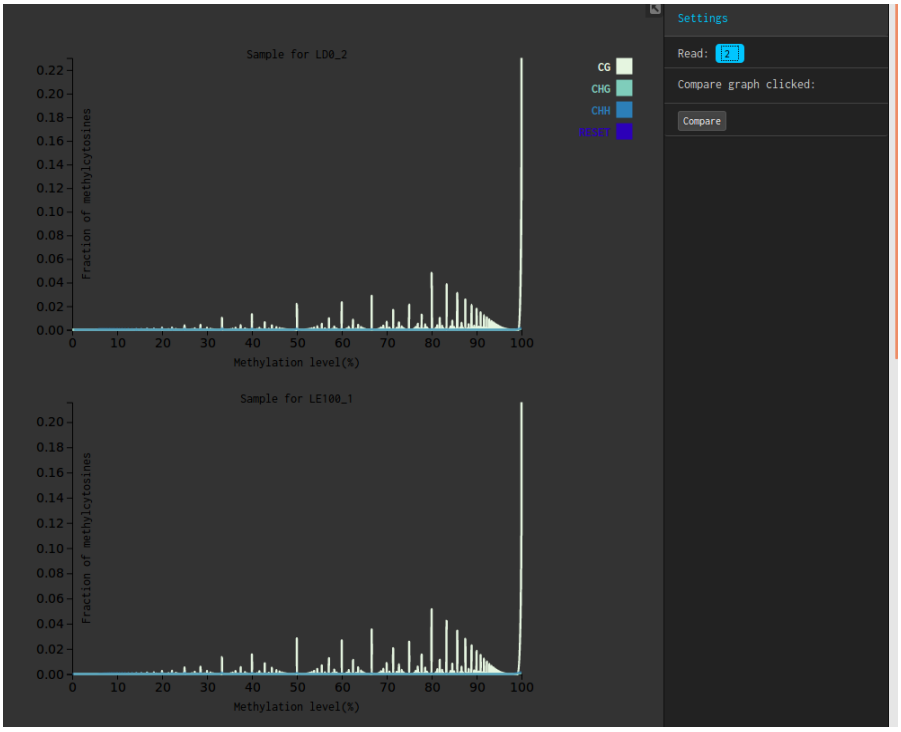


Figure 4.4.11 User can select the distribution of methylation that is larger than n reads. N is in range of 1 to 10.

SECTION 4.5- PERCENTAGE OF METHYLATED CYTOSINES INCLUDING MCG, MCHG AND MCHH (PIE CHART)

Percentage of methylated cytosine is visualized using pie chart. Number of pie charts depends on the number of samples. Data processing, data loading and visualization of pie chart are included in the graph.

Data is being processed when sample file is uploaded. Similar to line graph from 4.4, the count for each methylated context (CG, CHG, CHH) is counted and saved in methylation_percentage.csv. Data is retrieved from api/process/methylation?option=4&read=0 (Figure 4.5.3). Count for methylated context for each sample will be returned as shown in Figure 4.5.4. Count for each methylated context is retrieved and fed into pie chart generator in d3.

CHAPTER 4: RESULT DELIVERED

```
#dmp3
context_percentage[index]+=1
context_percentage["total"]+=1
break
```

Figure 4.5.1 Data preprocessing – get the number of count of each context

```
#dmp3
completeName = os.path.join(settings.BASE_DIR, 'db','public',
                    self.mount_point.fkey.name,file_name,
                    'methylation_percentage.csv')
with open(completeName,'ab+') as file:
    for ind_processed_data in processed_data:
        freData=ind_processed_data["context_percentage"]
        for ind_freData in freData:
            file.write(str(cnt_alias))
            file.write('\t')
            file.write(ind_processed_data['seg_name'])
            file.write('\t')
            file.write(str(ind_freData))
            file.write('\t')
            file.write(str(freData[ind_freData]))
            file.write('\n')
file.closed
```

Figure 4.5.2 Save data into file

```
TS visualizer.ts .../methylation_graph6_heatmap   api.py ×   manage.py   TS visualizer.ts .../methylation_graph3
                    url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                    api_item.folder,
                                    'stack_bar_data.csv'))
                flag_tabix=True
        except TabixAPI.DoesNotExist:
            flag_tabix=False

        try:

            api = UploadFile.objects.filter(non_tabix=True)
            for api_item in api:

                if option=="1":
                    url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                    api_item.mount_point.fkey.name,api_item.sample_upload_zip,
                                    'depth_cov'+str(read_num)+'.csv'))
                elif option=="2":
                    url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                    api_item.mount_point.fkey.name,api_item.sample_upload_zip,
                                    'depth_coverageQc2.csv'))
                elif option=="3":
                    url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                    api_item.mount_point.fkey.name,api_item.sample_upload_zip,
                                    'depth_coverageQc3_'+str(read_num)+'.csv'))
                elif option=="4":
                    url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                    api_item.mount_point.fkey.name,api_item.sample_upload_zip,
                                    'methylation_percentage.csv'))
                elif option=="7":
                    url_list.append(os.path.join(settings.BASE_DIR, 'db','public',
                                    api_item.mount_point.fkey.name,api_item.sample_upload_zip,
                                    'stack_bar_data.csv'))
                flag_upload=True
        except UploadFile.DoesNotExist:
            flag_upload=False
```

Figure 4.5.3 Read file as request

CHAPTER 4: RESULT DELIVERED

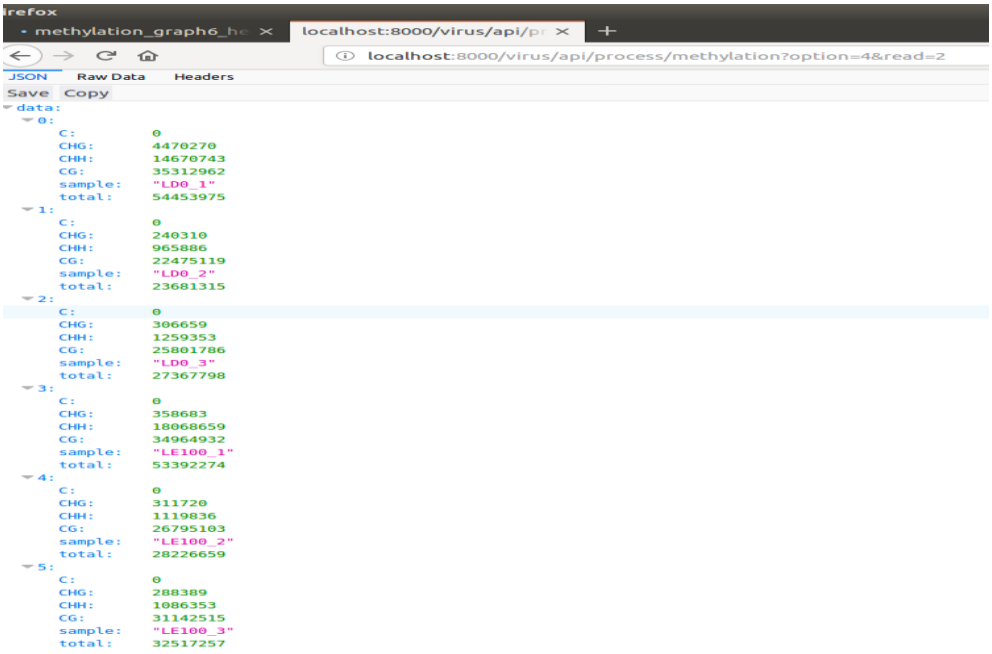


Figure 4.5.4 Result returned

CHAPTER 4: RESULT DELIVERED

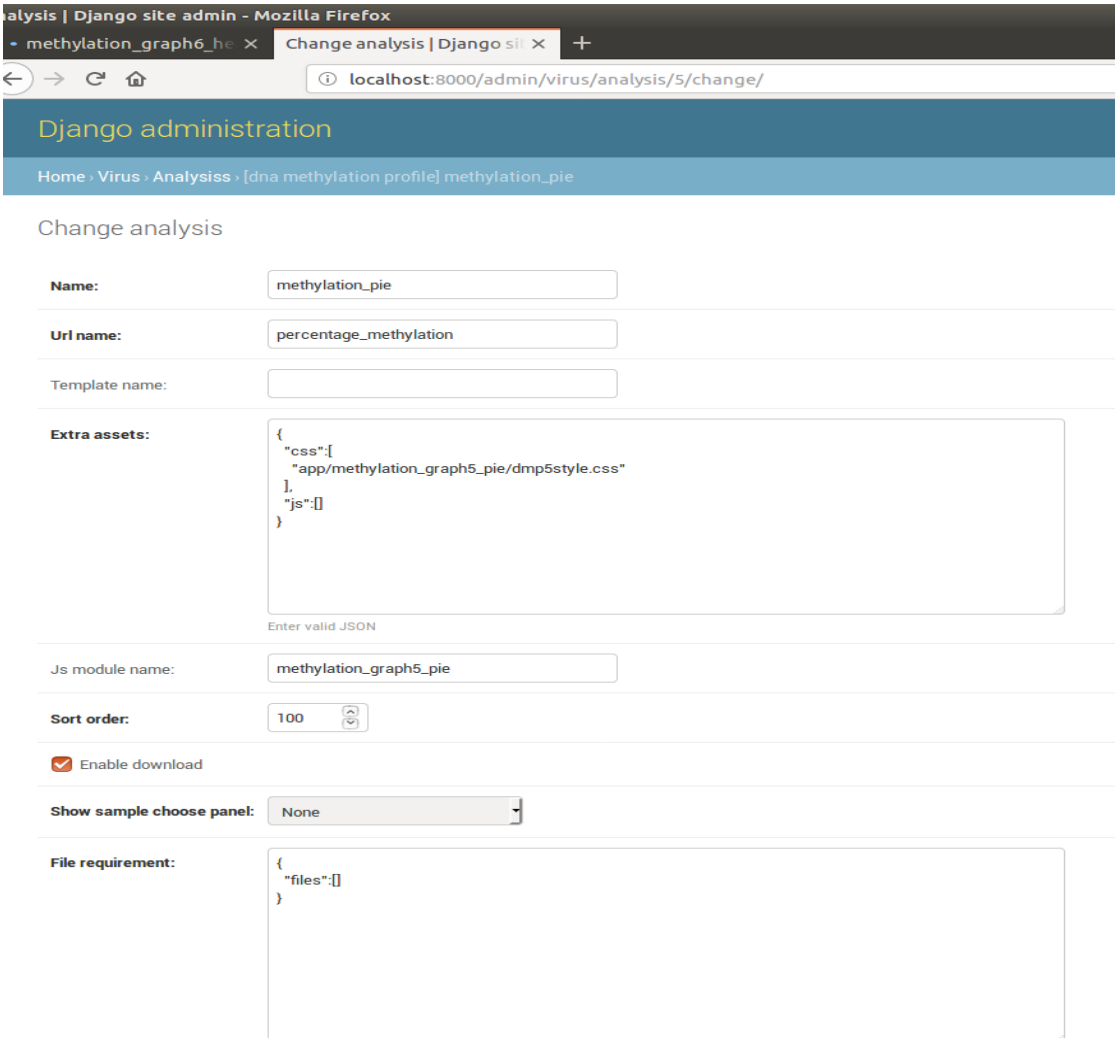


Figure 4.5.5 Setup of percentage of methylated cytosine

```
278
279
280     public run(): void {
281
282         this.data.count = 0
283         var total = 0
284         var percentageLocal = []
285         var percentagedict = {}
286         var list_j = ["CG", "CHH", "CHG"]
287
288         this.loadData("mCytosine", (d) => {
289             this.data.DepGraph = []
290             //Array
291             for (let j of d["data"]) {
292                 percentageLocal = []
293                 total = 0
294                 var sample_name = j["sample"]
295
296                 for (var test of list_j) {
297                     percentagedict = {}
298                     percentagedict["key"] = test
299                     percentagedict["value"] = parseInt(j[test])
300
301                     percentageLocal.push(percentagedict)
302
303
304                 }
305
306
307                 this.data.DepGraph[this.data.count] = <any>{}
308                 //need to perform deep copy----DONE
309                 var ycount = Math.floor((this.data.count) / Math.floor(this.option.svgWidth/300))
310                 var xcount = this.data.count % Math.floor(this.option.svgWidth/300)
311                 console.log(xcount,Math.floor(this.option.svgWidth/300) )
312                 this.data.DepGraph[this.data.count] = (
313                     {
314                         //X,Y position incorrect ----DONE
315                         xGraph: 300 * xcount + 100, yGraph: 300 * ycount + 100,
316                         percentageData: percentageLocal,
317                         sampleName: sample_name,
318                         total: j["total"]
319
320                     })
321
322                 this.data.count++
323             }
324
325
326
327         })
328
329         console.log(this.data.DepGraph)
```

Figure 4.5.6 Data is processed to get percentage of methylated cytosine.

Pie chart is drawn using d3.pie, path and arc. Text is placed inside the sector. The boundary of the text is calculated and checked whether the text is in the rectangle (within the pie sector). However, when the sector is too small to fit the text, the text will be moved out and displayed above the pie chart (Figure 4.5.9 & Figure 4.5.10). Interaction added for this graph is hovering (Figure 4.5.8). Div will be displayed when user hovers the pie chart. The radius of sector will be enlarged to make the hover effect significant as shown in Figure 4.5.7(a) and Figure 4.5.7(b).

```
this.addLayer("percentage", ["percentage"]).renderer(function ($size: Size, $data: Data, $op: Option, vz: Visualizer) {

    var width = 360
    var height = 360
    var radius = 100
    var color = d3.schemeCategory20c;
    for (var graphCnt = 0; graphCnt < $data.count; graphCnt++) {
        var key = function (d) { return d.key; };

        var values = Object.keys($data.DepGraph[graphCnt].percentageData).map(function (key) {
            return $data.DepGraph[graphCnt].percentageData[key]["value"] / $data.DepGraph[graphCnt].total * 100;
        });

        var percentage_pie = this
            .attr("id", graphCnt)
            .data([$data.DepGraph[graphCnt].percentageData])
            .attr("width", width)
            .attr("height", height)
            .append("svg:g")
            .attr("transform", "translate(" + ($data.DepGraph[graphCnt].xGraph) + "," + ($data.DepGraph[graphCnt].yGraph) + ")")

        var arc = d3.arc()
            .outerRadius(radius)
            .innerRadius(0);
        var arc_enlarge = d3.arc()
            .outerRadius(radius + 5)
            .innerRadius(0);

        var pie = d3.pie()
            .value(function (d) { return d["value"]; });

        var arcs = percentage_pie.selectAll("g.slice" + String(graphCnt))
            .data(<any>pie)
            .enter()
            .append("svg:g")
            .attr("class", "slice" + String(graphCnt));
        var tooltip_div = d3.select("body").append("div")
            .attr("class", "tooltip")
            .style("opacity", 0)
            .style("display", "inline-block");
```

(a)

(b)

Figure 4.5.7 Visualization of pie chart- initialize arc and slice for pie chart

```
arcs.append("svg:path")
    .attr("fill", function (d, i) { return color[i]; })
    .attr("d", arc)                    //this creates the actual SVG path using the associated data (pie) with the arc dra
    .on("mouseenter", function (d) {

        d3.select(this)
            .attr("stroke", "white")
            .transition()
            .duration(500)
            .attr("d", arc_enlarge)
            .attr("stroke-width", 2);
        tooltip_div.transition()
            .duration(200)
            .style("opacity", .9);
        tooltip_div.html((d["data"]["key"]) + "<br>" + (d["data"]["value"]) )
            .style("text-align", "left")
            .style("height", 45)
            .style("left", d3.event.pageX+10+"px")
            .style("top", d3.event.pageY-25+"px");
    })
    .on("mouseleave", function (d) {
        d3.select(this).transition()
            .attr("d", arc)
            .attr("stroke", "none");
        tooltip_div.transition()
            .duration(500)
            .style("opacity", 0);
    });
arcs.append("svg:text")
    .attr("transform", function (d) {
        d["innerRadius"] = 0;
        d["outerRadius"] = radius - 10;
        return "translate(" + arc.centroid(<any>d) + ")";        //this gives us a pair of coordinates like [50, 50]
    })
    .attr("opacity", 1.0)
    .attr("text-anchor", "middle")
    .text(function (d, i) { return (d["data"]["value"] / $data.DepGraph[graphCnt].total * 100).toFixed(2) + "%"; })
    .attr("x", -10) //Move the text from the start angle of the arc
    .attr("dy", 5) //Move the text from the start angle of the
```

Figure 4.5.8 Interaction for mouse enter and mouseleave

```
.attr("dy", 5) //move the text from the start angle of the
.each(function (d,i) {
    function pointIsInArc(pt, ptData, d3Arc) {

        //get the parameter from the arc
        var r1 = arc.innerRadius()(ptData),
            r2 = arc.outerRadius()(ptData),
            theta1 = arc.startAngle()(ptData),
            theta2 = arc.endAngle()(ptData);

        //get distance of point from circle
        var dist = pt.x * pt.x + pt.y * pt.y,
            angle = Math.atan2(pt.x, -pt.y);

        //get correct angle
        angle = (angle < 0) ? (angle + Math.PI * 2) : angle;

        //check whether text is in the (arc)circle or not.
        return (r1 * r1 <= dist) && (dist <= r2 * r2) &&
            (theta1 <= angle) && (angle <= theta2);
    }
    var textElem = <SVGTextElement>this
    var textBBox = textElem.getBBox();


    var center = arc.centroid(<any>d);

    var topLeft = {
        x: center[0] + textBBox.x,
        y: center[1] + textBBox.y
    };

    var topRight = {
        x: topLeft.x + textBBox.width,
        y: topLeft.y
    };

    var bottomLeft = {
        x: topLeft.x,
        y: topLeft.y + textBBox.height
    };

    var bottomRight = {
        x: topLeft.x + textBBox.width,
        y: topLeft.y + textBBox.height
    };
```

Figure 4.5.9 calculate boundary for text

```
    };
    var dist1 = topLeft.x * topLeft.x + topLeft.y * topLeft.y,
    dist2=topRight.x * topRight.x + topRight.y * topRight.y
    d["visible"] = pointIsInArc(topLeft, d, arc) &&
        pointIsInArc(topRight, d, arc) &&
        pointIsInArc(bottomLeft, d, arc) &&
        pointIsInArc(bottomRight, d, arc);

        console.log(Math.sqrt(dist1-dist2));

    if (d["visible"] == false) {
        console.log(parseInt(d3.select(this).attr("x")))
        var newx=-5+parseInt(d3.select(this).attr("x"))*2

        d3.select(this).attr("x", Math.sqrt(dist1-dist2)*d["index"]*2-100)
            .attr("dy", -55)
            .attr("fill", "black")

    }

})

this.append("text")
    .attr("id", graphCnt)
    .attr("text-anchor", "middle")
    .attr("transform", "translate(" + ($data.DepGraph[graphCnt].xGraph) + "," + (0 - 125 + $data.DepGraph[graphCnt].yGraph) +
    .text("Sample for " + $data.DepGraph[graphCnt].sampleName)

}
})

}
```

Figure 4.5.10 check text is in boundary or not

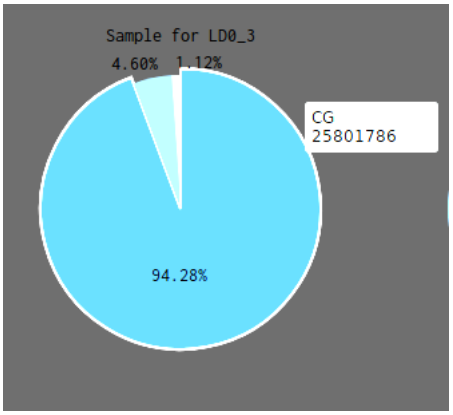


Figure 4.5.11 Pie chart for percentage of methylated cytosine in each context

From Figure 4.5.12, user can observe that sample 1 in day 0 (LE0_1) and day 100 (LD100_1) are having more CHH compared to others. The details of the context will be displayed through hovers the sector of pie chart.

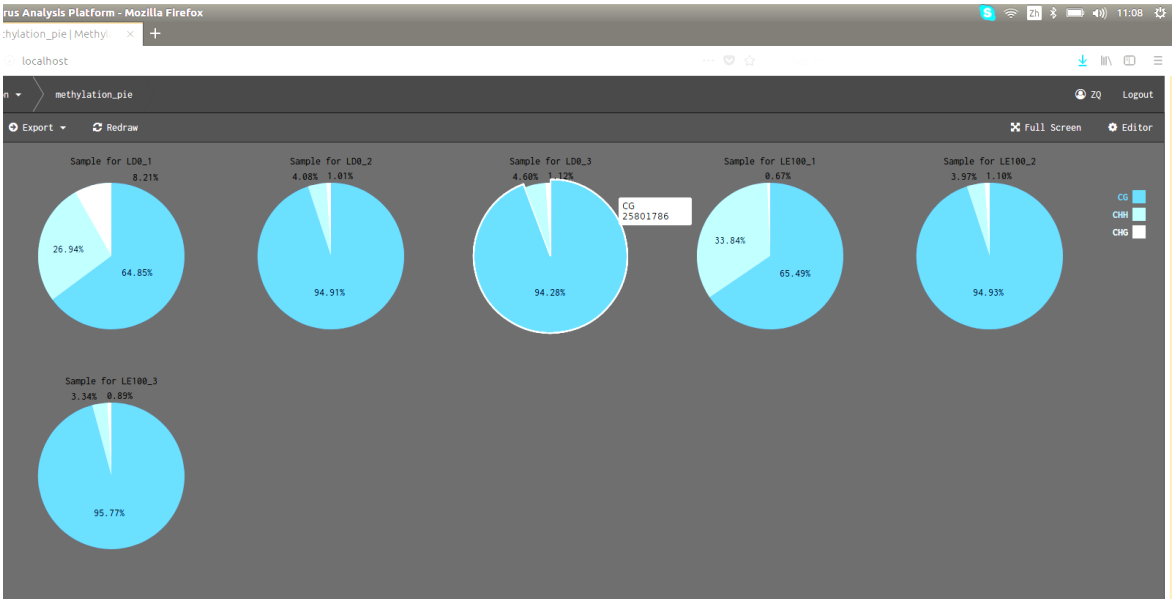


Figure 4.5.12 Complete percentage of methylated cytosine in each context.

SECTION 4.6- ELEMENT TARGET (BAR CHART)

Element target shows the fraction of CpG in low, intermediate and high. Each methylation ratio is compared and grouped. The graph helps user to identify the methylation pattern in each region.

Details of region that corresponding to the position of the chromosome are read as shown in Figure 4.6.1. Range of each region is read to detect the category of the position in the latter part of data processing. The sample within the position will be categorized as

low (<0.25), intermediate (0.25-0.75) and high (>0.75) according to Figure 4.6.2. Binary search is used to search whether the position is in range. Checking list of range in each region consumes a lot of time by using linear search (>O (*n*)). Thus, binary search is used to speed up the data retrieval process by spending only O (*log* (*n*)). Processed data is stored in stack_bar_data.csv. The data is loaded from api/process/methylation?option=7&read=0. Related information will be retrieved from the file- stack_bar_data.csv. Count of CpG for each level and different region are calculated.

```
        stack_bar_data={}
        #process_file_get_position for bar chart
        element_sum_file_name=os.path.join(settings.BASE_DIR, 'db','public',
                                "methylation","region",
                                "element.list")
        with open(element_sum_file_name, "r") as elements_file:
            for element_ind in elements_file:
                element=element_ind.split('\n')[0].split('\t')
                if not (element[0]=="#Element"):
                    element_file_name[element[0]]=element[1]
            print element_file_name

            for element_ind in element_file_name:
                    element_small_file_name=os.path.join(settings.BASE_DIR, 'db','public',
                                    "methylation","region",
                                    element_file_name[element_ind])
                    print "reading file"+element_small_file_name
                    with  open(element_small_file_name, "r") as element_line:
                            range_list[element_ind]=[]
                            for line in element_line:
                                temp=line.split('\n')[0].split('\t')
                                range_list[element_ind].append((temp[1].strip().split(' ')[0],temp[2].strip().split(' ')[0]))
                    sorted(range_list[element_ind])
                    stack_bar_data[element_ind]={}
                    stack_bar_data[element_ind].setdefault("0.25-0.75",0)
                    stack_bar_data[element_ind].setdefault("0.25",0)
                    stack_bar_data[element_ind].setdefault("0.75",0)
                    print element_ind
                    element_line.close()


        for ratio_fre_Cnt in range(0,10):
            ratio_frequency.append({})

        for index in list_cytosine:
            for read_ind in read_list:
                count[index+str(read_ind)]=0
                total[index+str(read_ind)]=0
            for ratio_fre_Cnt in range(0,10):
                ratio_frequency[ratio_fre_Cnt][index]={}
            context_percentage[index]=0
        context_percentage["total"]=0


        row_return['read_num']=read_list
        row_return['seg_name']=''
```

Figure 4.6.1 Get the range for each region in stacked bar chart

```
#stack_bar7

for element_ind_range in range_list:

    if (self.binarysearch(range_list[element_ind_range],row["pos"])!=-1):
        if float(row["ratio"])<0.25:
            stack_bar_data[element_ind_range]["0.25"]+=1
        elif float(row["ratio"])>0.75:
            stack_bar_data[element_ind_range]["0.75"]+=1
        else:
            stack_bar_data[element_ind_range]["0.25-0.75"]+=1
        break
```

Figure 4.6.2 Data processing-categorize methylation ratio of the sample.

```
1071
1072        @staticmethod
1073        def bindtofile(self,processed_data,type,cnt_alias=0):
1074            if type==0:
1075                file_name=  self.sample_upload_tabix
1076            else:
1077                file_name=  self.sample_upload_zip
1078            list_file=[1,2,5,10,100]
1079            #stack_bar7
1080            completeName = os.path.join(settings.BASE_DIR, 'db','public',
1081                                        self.mount_point.fkey.name,file_name,
1082                                        'stack_bar_data.csv')
1083            with open(completeName,'ab+') as file:
1084                for ind_processed_data in processed_data:
1085                    elementData=ind_processed_data["stack_bar_data"]
1086                    for ind_element_data in elementData:
1087                        for ind_percentage_data in elementData[ind_element_data]:
1088                            detail_data= elementData[ind_element_data][ind_percentage_data]
1089                            file.write(str(cnt_alias))
1090                            file.write('\t')
1091                            file.write(ind_element_data)
1092                            file.write('\t')
1093                            file.write(ind_percentage_data)
1094                            file.write('\t')
1095                            file.write(str(detail_data))
1096                            file.write('\n')
1097            file.closed
```

Figure 4.6.3 Save processed data into corresponding file


```
• methylation_graph6_he ×   localhost:8000/virus/api/p ×   +
localhost:8000/virus/api/process/methylation?option=7&read=0
JSON   Raw Data   Headers
Save   Copy
data:
  0:
    5-UTR:
      0.25:            "1323036"
      0.25-0.75:       "796097"
      0.75:            "1479041"
    PseudoGene:
      0.25:            "526969"
      0.25-0.75:       "315673"
      0.75:            "569050"
    intron:
      0.25:            "3247022"
      0.25-0.75:       "2097058"
      0.75:            "3688806"
    sample:            "LD0_1"
    transposons:
      0.25:            "5963590"
      0.25-0.75:       "3635288"
      0.75:            "6536789"
    tandem_repeats:
      0.25:            "87816"
      0.25-0.75:       "47998"
      0.75:            "93348"
    mRNA:
      0.25:            "1345910"
      0.25-0.75:       "832490"
      0.75:            "1487976"
    CDS :
      0.25:            "13449"
      0.25-0.75:       "7901"
      0.75:            "14480"
    3-UTR:
      0.25:            "111238"
      0.25-0.75:       "64547"
      0.75:            "117792"
    ncRNA:
      0.25:            "279689"
      0.25-0.75:       "175884"
      0.75:            "317905"
  1:
    5-UTR:
      0.25:            "94575"
      0.25-0.75:       "480857"
```


Figure 4.6.4 Returned data for each region in sample

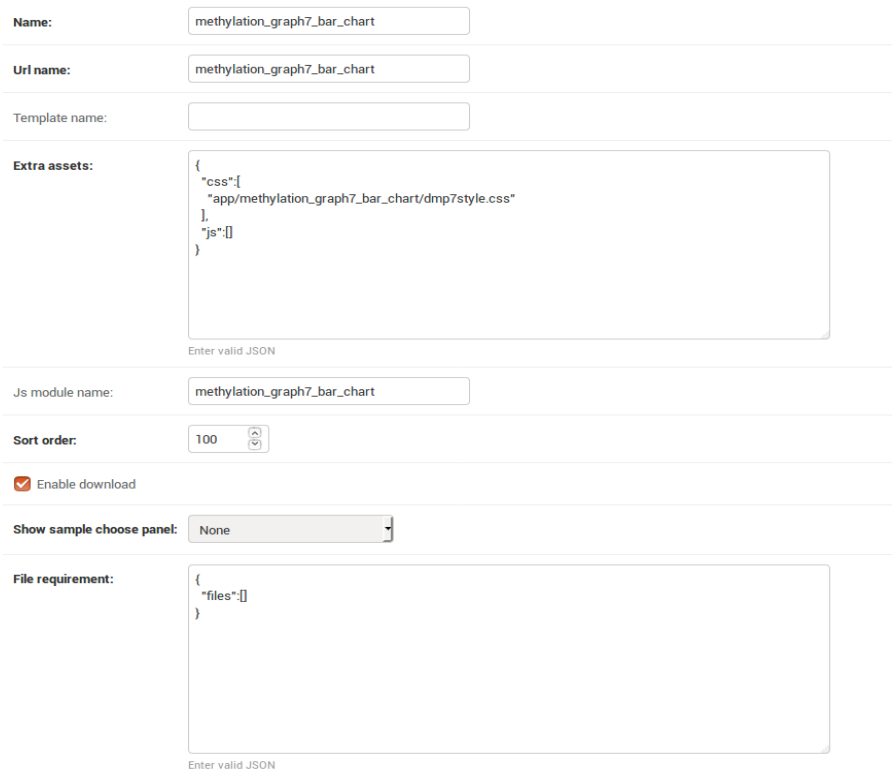


Figure 4.6.5 Setup for element target in bar chart.

A stacked bar chart is visualized to determine the percentage of CpG in each region. Count is divided by total count in respective region as calculated in Figure 4.6.6 (a). Sum for each column in the bar chart is 1. The number of stacked bar chart per row is based on the size of svg (Figure 4.6.6 (b)). Smaller window size will have lesser chart in a row.

```
public run(): void {

    this.data.count = 0
    var total = 0
    var percentageLocal = []
    var percentagedict = {}

    this.loadData("mCytosine", (d) => {

        this.data.DepGraph = []
        //Array
        /*
        var bar_data = [{ key: "5-UTR", "0.25": 2000, "0.5-0.75": 3000, "0.75": 1000 }, { key: "PseudoGene", "0.25": 2000, "0.5-0.75": 3000, "
        { key: "ncRNA", "0.25": 2000, "0.5-0.75": 3000, "0.75": 1000 }
            , { key: "mRNA", "0.25": 2000, "0.5-0.75": 3000, "0.75": 1000 }, { key: "intron", "0.25": 2000, "0.5-0.75": 3000, "0.75": 1000 },
        { key: "CDS", "0.25": 2000, "0.5-0.75": 3000, "0.75": 1000 }]*/
        for (var temp_d of d["data"]) {

            var sample_name=temp_d["sample"]
            var data = [];

            for (var test_case in temp_d) {
                if(test_case!="sample"){

                var total = parseInt(temp_d[test_case]["0.25"]) + parseInt(temp_d[test_case]["0.25-0.75"]) + parseInt(temp_d[test_case]["0.75"

                let data_part = {
                    key: test_case,
                    "0.25": parseInt(temp_d[test_case]["0.25"]) / total,
                    "0.25-0.75":  parseInt(temp_d[test_case]["0.25-0.75"]) / total,
                    "0.75":  parseInt(temp_d[test_case]["0.75"]) / total,
                    "total": 1,
                    "int_total": total
                }
                data.push(data_part);
            }
            };
            var keys = ["key", "0.25", "0.5-0.75", "0.75"].slice(1);

            this.data.DepGraph[this.data.count] = <any>{}
            //need to perform deep copy----DONE
            var ycount = Math.floor((this.data.count) / Math.floor(this.option.svgWidth / Visualizer.width ))
            var xcount = this.data.count % Math.floor(this.option.svgWidth / Visualizer.height)
            this.data.DepGraph[this.data.count] = (
                {
                    //X,Y position incorrect ----DONE
                    xGraph: (Visualizer.width+50) * xcount , yGraph: (Visualizer.height+180) * ycount ,
                    percentageData: data,
                    sampleName: sample_name

                })
```

(a)

(b)

Figure 4.6.6 Data processing –Calculate fraction for each level. Dynamic number of item per row is set.

Visualization of stacked bar chart is formed by axis and rectangle. Data will be passed into a group and a rectangle will be appended into the graph for every row of data. Hover interaction is added to the graph. A div will be displayed when user hovers the graph. Details information such as number of count, total number of count and the fraction of CpG are displayed in the div as shown in Figure 4.6.9. Variable z stores the color with keys (0.25, 0.25-0.75, 0.75) as domain. The first 3 color in z will present the keys.

```
119         this.addLayer("percentage", ["percentage"]).renderer(function ($size: Size, $data: Data, $op: Option, vz: Visualizer) {
120
121             var width = 400
122             var height = 400
123             var color = d3.schemeCategory20b;
124             for (var graphCnt = 0; graphCnt < $data.count; graphCnt++) {
125
126
127                 var
128                     margin = { top: 20, right: 20, bottom: 30, left: 40 },
129                     g = this.append("g") .attr("transform", "translate(" + ($data.DepGraph[graphCnt].xGraph ) + "," + ($data.DepGraph[graphCnt].y
130
131
132                 var x = d3.scaleBand()
133                     .rangeRound([0, width])
134                     .paddingInner(0.05)
135                     .align(0.1);
136
137                 var y = d3.scaleLinear()
138                     .rangeRound([height, 0]);
139                 var keys = ["key", "0.25", "0.25-0.75", "0.75"].slice(1);
140                 var z = d3.scaleOrdinal(d3["schemeCategory20c"]);
141                 $data.DepGraph[graphCnt].percentageData.sort(function (a, b) { return b["total"] - a["total"]; });
142                 x.domain($data.DepGraph[graphCnt].percentageData.map(function (d) { return d["key"]; }));
143                 y.domain([0, d3.max($data.DepGraph[graphCnt].percentageData, function (d) { return d["total"]; })]).nice();
144                 z.domain(keys);
145
146                 var tooltip_div = d3.select("body").append("div")
147                 .attr("class", "tooltip")
148                 .style("opacity", 0)
149                 .style("display", "inline-block");
150                 g.append("g")
151                     .selectAll("g")
152                     .data(d3.stack().keys(keys)($data.DepGraph[graphCnt].percentageData))
153                     .enter().append("g")
154                     .attr("fill", function (d) { return z(d["key"]); })
155                     .selectAll("rect")
156                     .attr("class", function (d) { return d["key"]; })
157                     .data(function (d) { return d; })
158                     .enter().append("rect")
159                     .attr("x", function (d) { return x(String(d["data"]["key"])); })
160                     .attr("y", function (d) { return y(d[1]); })
161                     .attr("height", function (d) { console.log(d);return y(d[0]) - y(d[1]); })
162                     .attr("width", x.bandwidth())
163                     .on("mouseenter", function (d) {
```

(a)

Figure 4.6.7 Visualization of stacked bar chart. Append rectangle on graph to form stacked bar chart

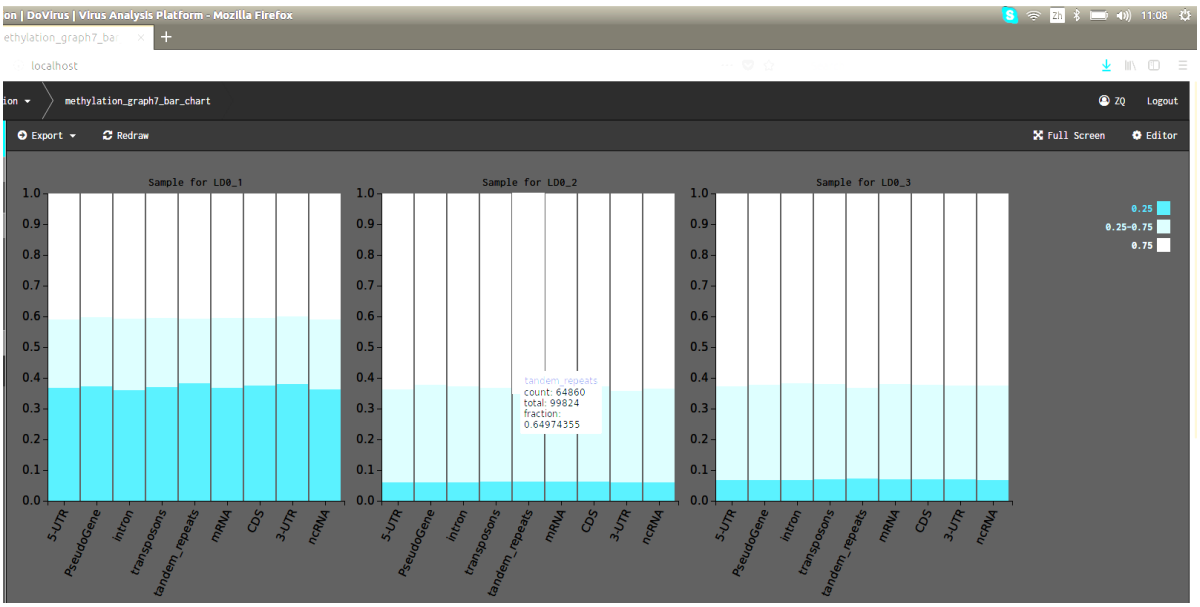


Figure 4.6.9 Partial stacked bar chart

From figure 4.6.9, user can find that the percentage of methylated CG in sample 1 have a significant difference compare to other. Sample 2 and Sample 3 have a similar percentages of methylated CG.

SECTION 4.7- CLUSTERING AND PCA ANALYSIS OF METHYLATION OF CPG SITES ACROSS SAMPLES

Data used in heatmap and dendrogram does not pre-process when user uploads the sample file. This is because the data that is needed to be stored is too big. Thus, the data is processed only when user requests.

The data is retrieved form api/process/methylation?option=5&read=0. The sample file from gzip and tabix will be read. From figure 4.7.1(a), the sample is grouped by dividing the position by 10k and methylation ratio of the group will be summed up. Figure 4.7.1(b) shows the program accepts dynamic arrangement for input file. However, performance of Figure 4.7.1(b) is much slower compared to Figure 4.7.1(a). Thus, Figure 4.7.1(b) is comment out and Figure 4.7.1(a) is used for testing. For instance, [10000:0.5, 10002:1] will be grouped under [1] and form array of [1:1.5]. The count of methylation ratio added to the position that form [1:1.5] is recorded in the form of [1:2]. After all of the positions in the sample are grouped, the value of positions is divided by count of methylation ratio in their positions-[1: 1.5/2] to get the mean of [1:0.75].

The positions of a sample are compared with other samples to ensure all of them will having same amount of position in the sample. The similar set between samples is retrieved as Figure 4.7.2 (a). Similar set from sample 1 and 2 is compared and added to `common_set`. `common_set` will be compared with next common set retrieved from similar set from sample 2 and sample 3. The common set of each sample is taken out and replaces the original set as shown in Figure 4.7.2(b). If A have 1000 positions and B have 1000 positions, A have 500 different position compared to B. These 500 positions will be taken out from A and B and form the samples with 500 positions.

```
681            with gzip.open(api_item.file.name) as f:
682                    with io.BufferedReader(f) as gz_read:
683                        for line in gz_read:
684                            row_split=line.split("\t")
685                            if (row_split[6])=='CG':
686                                if row_split[5]!='NA':
687                                    ind=int(math.floor(int(row_split[1])/10000))
688                                    methylation_ratio_val=methylation_ratio_dict.setdefault(ind,0)+float(row_split[5])
689                                    m_count=count_dict.setdefault(ind,0)+1
690                                    methylation_ratio_dict[ind]=methylation_ratio_val
691                                    count_dict[ind]=m_count
692
693
694                            '''
695                            col_count = -1
696                            row={}
697                            row_split=entry.split("\t")
698                            if col_count < 0:
699                                col_count = len(row_split)
700                            for i in xrange(col_count):
701                                col = row_split[i]
702                                col_name = col_list[i]
703                                if col_name == '_': continue
704                                row[col_name] = col
705                            if (row["contexttype"])=='CG':
706                                if row["ratio"]!='NA':
707                                    methylation_ratio[row["position"]] = methylation_ratio.get(row["position"], 0) + float(row["ratio"])
708                                    methylation_ratio_count[row["position"]] = methylation_ratio_count.get(row["position"], 0) + 1
709                                '''
710
711            row_return["sample"]=api_item.sample_upload_zip
712            row_return["methylation_ratio"]={k:methylation_ratio_dict[k]/count_dict[k] for k in methylation_ratio_dict}
713            result.append(row_return)
714
```

(a)

(b)

Figure 4.7.1 Data processing-Group and calculate mean of each grouped position

```
723
724    common_set=set()
725
726    #get common set between set
727    for row_index in range(0,len(result)-1):
728        if(len(common_set)>0):
729            common_set=common_set&(set(result[row_index]["methylation_ratio"])&set(result[row_index+1]["methylation_ratio"]))
730        else:
731            common_set=(set(result[row_index]["methylation_ratio"])&set(result[row_index+1]["methylation_ratio"]))
732    for row_index in range(len(result)):
733
734        sample_name={str(row_index):result[row_index]["sample"]}
735        sample_name_list.append(sample_name)
736        common_1={}
737        for com_i in common_set:
738            common_1[com_i]=result[row_index]["methylation_ratio"][com_i]
739        print "first_sorting"
740        sorted_x_1 = sorted(common_1.items(), key=lambda x: (x[1],int(x[0])))
741        arr= [{"pos":int(item[0]), "value":(item[1]),"sample":row_index} for item in sorted_x_1]
742        #make sure the original set updated
743        result[row_index]["methylation_ratio"]=arr
744
745
746    for item in result:
747        methylation_ratio_total+=item["methylation_ratio"]
748
749
750    print "sorting"
751    result_update["methylation_ratio"]=methylation_ratio_total
752    result_update["sample"]=sample_name_list
753
754
755    return result_update
756
757
```

(a)

(b)

Figure 4.7.2 Data processing- Get the similar set for all the samples

SECTION 4.7.1- 3D CUBE FOR PCA ANALYSIS

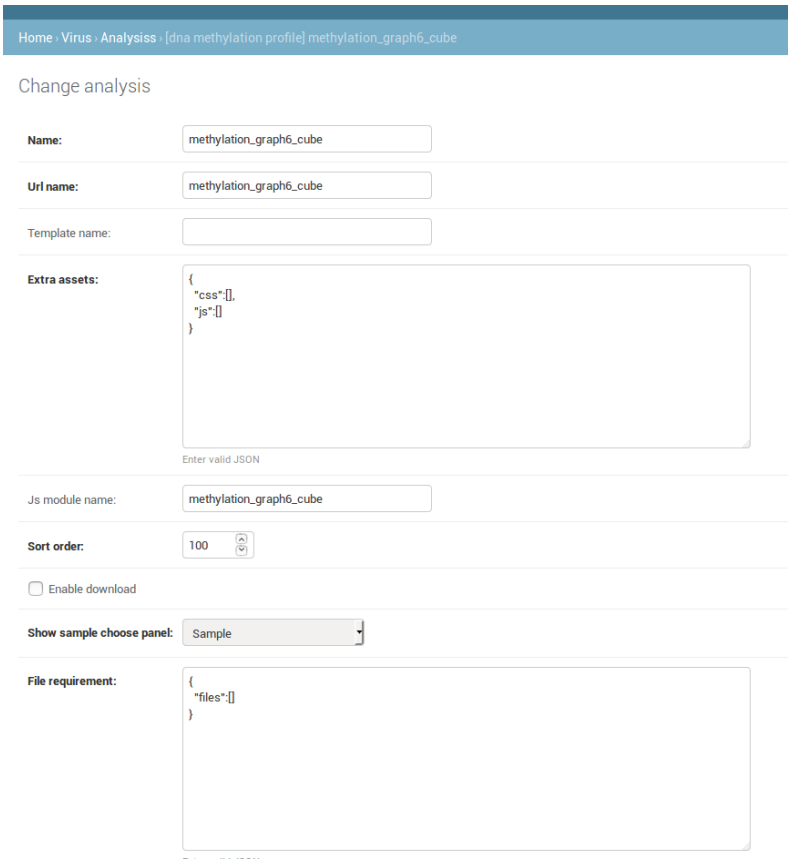


Figure 4.7.1.1 Setup for 3D cube for pca

Data returned from api is mapped to data structure in Figure 4.7.1.3 (a) and Figure 4.7.1.3 (b). Color of samples inside 3D scatter plot is grouped by name of sample as shown in Figure 4.7.1.2. Default color for the sample will be yellow (#EFEF66). For those samples that start with LE, color for these samples will be purple (#F0FF0). For those samples that start with LD, color for these samples will be light blue (#7491C1). Data is passed into PCA to form feature vectors that helps to predict the score for PCA. PCA is used to reduce the dimensionality of the graph. The plugin package used helps to calculate the eigenvectors and eigenvalues of the data. The score for the sample will be returned through `pca.predict`. Data that is used to visualize sample with PCA scoring is stored as Figure 4.7.1.3(e).

```
for (var i of Name) {

        var colorstr = this.data.sample_list[ i["key"]][ i["key"]].substring(0, 2);

    console.log(colorstr)
        var clr = "0xefef66"

        if (colorstr === "LE") {
            clr = "0xcc0ff0"
        }
        else if (colorstr === "LD") {
            clr = "0x7491c1"
        }
        colorarr.push(clr)
```

Figure 4.7.1.2 Color grouping for sample

```
TS visualizer.ts .../methylation_graph6_heatmap   api.py ●   manage.py   TS visualizer.ts .../methylation_graph3   TS visualizer.ts .../methylation_
442                 var test = {
443                     "sample": i["key"], ["values"]: d3.nest()
444                         .key(function (d) { return d["pos"]; })
445                         .entries(i["values"])
446                         .map(function (group) {
447                             return {
448                                 "pos": group.key,
449                                 "mean": d3.mean(group.values, function (d) { return d["value"]; }),
450                                 "values": group.values
451                             }
452                         })
453                 };
454
455                 let newObj = JSON.parse(JSON.stringify(test["values"].map(function (group) {
456                     return {
457                         "pos": group["pos"],
458                         "value": group["mean"],
459                         "sample": i["key"]
460                     }
461                 })));
462                 /*newObj.sort(function (a, b) {
463                     return (a["value"] - b["value"]);
464                 })*/
465                 this.data.key_list.push(newObj.map(item => item["pos"])
466                     .filter((value, index, self) => self.indexOf(value) === index));
467                 this.data.dendrogramData.push({ "sample": i["key"], "value_list": newObj.map(a => a["value"])
468
469                 data.push(newObj);
470             }
471             var merged = [].concat.apply([], data);
472             merged.sort(function (a, b) {
473                 return (a["value"] - b["value"]);
474             })
475             this.data.heatmapGraphData = merged
476
477             const PCA = require('ml-pca');
478             // dataset is a two-dimensional array where rows represent the samples and columns the features
479
480             this.data.dendrogramData = this.data.dendrogramData.map(d => d["value_list"]);
481             console.log(this.data.dendrogramData);
482             const pca = new PCA(this.data.dendrogramData);
483
484
485             var pcascore = pca.predict(this.data.dendrogramData)
486             console.log(pcascore);
487             var score_index;
488             this.data.unfiltered = pcascore.map(function (item, i) {
489                 return {
490                     x: item[0],
491                     y: item[1],
492                     z: item[2],
493                     color:colorarr[i]
```

(a) (b) (c) (d) (e)

Figure 4.7.1.3 Data processing before visualization- Calculate PCA scoring for visualization

D3.js does not support 3D visualization. Thus, to draw a 3D scatter plot, three.js is used. Renderer, screen and camera are three important components in visualization of 3D object. Camera is a very important component that controls the view of angle in 3D

graph. Perspective camera that is used in the project makes the scatter plot look natural. Figure 4.7.1.4 shows the views of different cameras. Alpha attribute in renderer is set to true so a transparent background can be displayed. Figure 4.7.1.5 (a) and Figure 4.7.1.5 (b) shows the initialization of camera and renderer for 3D scatter plot. Object3D will be used to visualize 3D scatter plot. All the spheres will be visualized inside the object.

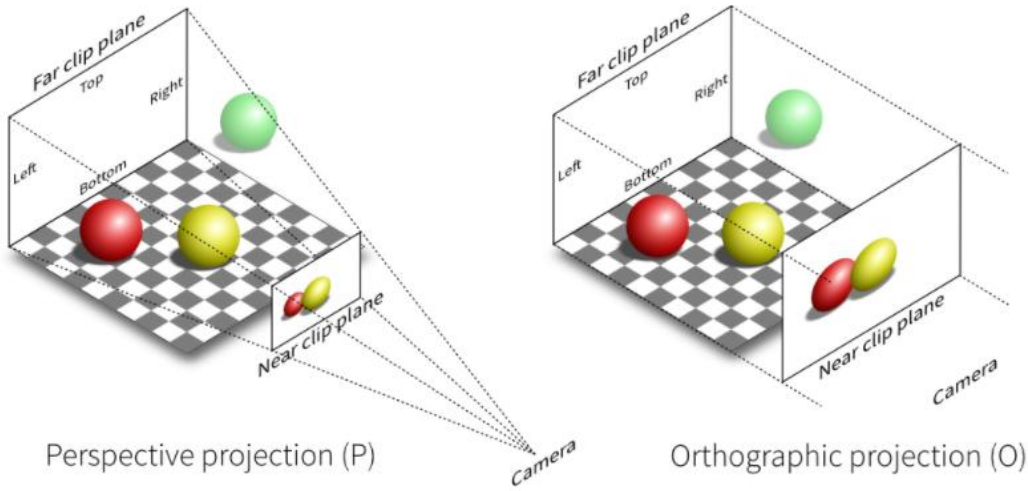


Figure 4.7.1.4 Perspective Camera and Orthographic camera that always be used in 3D visualization by three.js

```
public init(op?: Option) {

    super.initVisualizer(this, op)
    this.size = new Size(this.option.svgWidth)
    var margin = { top: 18, right: 18, bottom: 18, left: 18 }
    var text_margin = 120

    this.addLayer("cube", ["cube"]).renderer(function ($size: Size, $data: Data, $op: Option, vz: Visualizer) {
        if (document.getElementById("cube_scatter") != null) { document.getElementById("cube_scatter").remove(); }
        var elem = d3.select("body").append('div');
        elem.attr("id", "cube_scatter")
            .style("position", "absolute")
            .style("width", "100%")
            .style("height", "100%")
            .style("opacity", "1")
            .style("z-index", "100")


        const WIDTH = window.innerWidth;
        const HEIGHT = window.innerHeight;



        // Get the DOM element to attach to
        const container =
            document.querySelector('#cube_scatter');
        var renderer = new THREE.WebGLRenderer({
            antialias: true,
            alpha: true
        });

        var w = window.innerWidth;
        var h = window.innerHeight;
        renderer.setSize(w, h);
        container.appendChild(renderer.domElement);
        var camera = new THREE.PerspectiveCamera(45, w / h, 1, 1000);
        camera.position.z = 180;
        camera.position.x = -10;
        camera.position.y = 70;

        var scene = new THREE.Scene();

        var scatterPlot = new THREE.Object3D();
        scene.add(scatterPlot);

        scatterPlot.rotation.y = -0;
```

(a)

(b)

Figure 4.7.1.5 Initialize basic component for 3D visualization

Figure 4.7.1.6 (a) gets the min and max of the data so a cube with dynamic scale is visualized. Figure 4.7.1.6 (b) shows the visualized the wireframe of a cube. The wireframe of the cube is formed by connecting one point to another point without any changes on axis. The formation of wireframe of 3D scatter plot is shown in Figure 4.7.1.7. A 2D pane is drawn first and connected to another 2D pane. Combination of Figure 4.7.1.7 (b) and (c) forms a wireframe for 3D scatter plot. Changing of axis when connecting the point will result to Figure 4.7.1.8 which is different from expected.

```
function v(x, y, z) {
    return new THREE.Vector3(x, y, z);
}
renderer.setClearAlpha(0xffffff)
renderer.setClearColor(0xffffff, 0);
renderer.render(scene, camera);

var xExent = d3.extent($data.unfiltered, function (d) { return d["x"]; }),
    yExent = d3.extent($data.unfiltered, function (d) { return d["y"]; }),
    zExent = d3.extent($data.unfiltered, function (d) { return d["z"]; });
console.log(xExent, yExent, zExent)
var vpts = {
    xMax: xExent[1],
    xMin: xExent[0],
    yMax: yExent[1],
    yMin: yExent[0],
    zMax: zExent[1],
    zMin: zExent[0]
}

var colour = d3.scaleOrdinal(d3.schemeCategory10);
var xScale = d3.scaleLinear()
    .domain(xExent)
    .range([-50, 0]);
var yScale = d3.scaleLinear()
    .domain(yExent)
    .range([-50, 0]);
var zScale = d3.scaleLinear()
    .domain(zExent)
    .range([-50, 0]);

var lineGeo = new THREE.Geometry();

lineGeo.vertices.push(
    v(xScale(vpts.xMax), yScale(vpts.yMin), zScale(vpts.zMin)), v(xScale(vpts.xMax), yScale(vpts.yMax), zScale(vpts.zMin)),
    v(xScale(vpts.xMin), yScale(vpts.yMax), zScale(vpts.zMin)), v(xScale(vpts.xMin), yScale(vpts.yMin), zScale(vpts.zMin)), v(xScale(
    v(xScale(vpts.xMax), yScale(vpts.yMin), zScale(vpts.zMax)), v(xScale(vpts.xMax), yScale(vpts.yMax), zScale(vpts.zMax)),
    v(xScale(vpts.xMin), yScale(vpts.yMax), zScale(vpts.zMax)), v(xScale(vpts.xMin), yScale(vpts.yMin), zScale(vpts.zMax)),
    v(xScale(vpts.xMax), yScale(vpts.yMin), zScale(vpts.zMax)), v(xScale(vpts.xMax), yScale(vpts.yMax), zScale(vpts.zMax)),


    v(xScale(vpts.xMax), yScale(vpts.yMax), zScale(vpts.zMin)), v(xScale(vpts.xMax), yScale(vpts.yMax), zScale(vpts.zMax)),
    v(xScale(vpts.xMax), yScale(vpts.yMin), zScale(vpts.zMax)), v(xScale(vpts.xMax), yScale(vpts.yMin), zScale(vpts.zMin)), v(xScale(
    v(xScale(vpts.xMin), yScale(vpts.yMax), zScale(vpts.zMin)), v(xScale(vpts.xMin), yScale(vpts.yMax), zScale(vpts.zMax)),
    v(xScale(vpts.xMin), yScale(vpts.yMin), zScale(vpts.zMax)), v(xScale(vpts.xMin), yScale(vpts.yMin), zScale(vpts.zMin)),

);
```

(a)

(b)

Figure 4.7.1.6 Visualization- Formation of 3D scatter plot

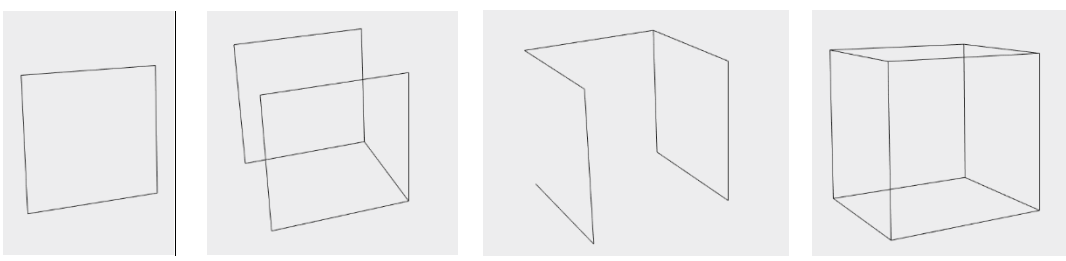


Figure 4.7.1.7 Formation of wireframe for 3D scatter plot

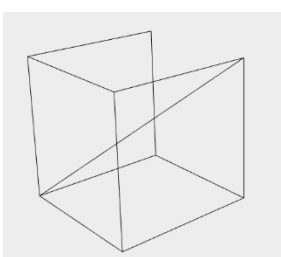


Figure 4.7.1.8 Connect point to point on different axis- Wrong visualization example

Text will be added to the edge of the 3D scatter plot through createText2D and createTextCanvas. Because this graph visualizing is a 3D scatter plot, it is hard to add 2D object like text. Thus, 2D text can be visualized through drawing of canvas and filling of text with specific color. Figure 4.7.1.9(a) shows the function that is used to create a

canvas and figure 4.7.1.10 (a) shows how to place a canvas on to three.js element to allowed double sided text. In figure 4.7.1.10 (b), minimum and maximum for axis x, y and z are drawn and placed to the edge of the 3D scatter plot.

```
var lineMat = new THREE.LineBasicMaterial({
    color: 0x0000ff,
    linewidth: 1
});
var line = new THREE.Line(lineGeo, lineMat);

scatterPlot.add(line);

function createTextCanvas(text, color, font, size) {
    size = size;
    var canvas = document.createElement('canvas');
    var ctx = canvas.getContext('2d');
    var fontStr = (size + 'px ') + (font);

    var w = ctx.measureText(text).width;
    var h = Math.ceil(size + 1);

    canvas.width = 70;
    canvas.height = h;
    ctx.font = fontStr;
    ctx.fillStyle = color;
    ctx.fillText(text, 0, size);
    return canvas;
}
```

(a)

Figure 4.7.1.9 Visualization (CreateTextCanvas) - helps to create text

virus - Visual Studio Code

TS visualizer.ts .../methylation_graph6_heatmap | api.py | manage.py | TS visualizer.ts .../methylation_graph3 | TS visualizer.ts .../methylation_graph6_cube

```
function createText2D(text, option = 0, color = 'white', font = 'Arial', size = 16, segW = 0, segH = 0) {

    var canvas;
    if (option === 0)
        canvas = createTextCanvas(text, color, font, size);
    else
        canvas = createTextCanvas2(text, color, font, size);
    var plane = new THREE.PlaneGeometry(canvas.width, canvas.height, segW, segH);
    var tex = new THREE.Texture(canvas);
    tex.needsUpdate = true;
    var planeMat = new THREE.MeshBasicMaterial({
        map: tex,
        color: 0xffffff,
        side: THREE.DoubleSide,
        transparent: true
    });
    var mesh = new THREE.Mesh(plane, planeMat);
    mesh.scale.set(0.2, 0.2, 0.2);
    return mesh;
}
var format = d3.format("+.3f");
var margin_text = 2;
var valueX = createText2D(format(xExent[0]));
valueX.position.x = xScale(vpts.xMin) - 3 * margin_text,
    valueX.position.y = yScale(vpts.yMin);
scatterPlot.add(valueX);

var valueX = createText2D(format(xExent[1]));
valueX.position.x = xScale(vpts.xMax) + 4 * margin_text,
    valueX.position.y = yScale(vpts.yMin);
scatterPlot.add(valueX);



var valueY = createText2D(format(yExent[1]));
valueY.position.x = xScale(vpts.xMin),
    valueY.position.y = yScale(vpts.yMax) + margin_text,
    valueY.position.z = zScale(vpts.zMin);
scatterPlot.add(valueY);


var valueY = createText2D(format(yExent[0]));
valueY.position.x = xScale(vpts.xMin) - 3 * margin_text,
    valueY.position.y = yScale(vpts.yMin),
    valueY.position.z = zScale(vpts.zMin);
scatterPlot.add(valueY);

var titleZ = createText2D(format(zExent[0]));
titleZ.position.z = zScale(vpts.zMin),
```

(a)

(b)

Figure 4.7.1.10 Visualization- Create text on the edge of x, y and z

For each sample like in Figure 4.7.1.11(a), a sphere that represents the sample is added to the 3DObject (scatterplot). The color of the sphere is based on the color of the sample that we grouped earlier in the data loading phase. MeshBasicMaterial is a material that will not be affected by light. The material of sphere for the sample will be MeshBasicMaterial with its respective color. Sample name that is drawn by 2D canvas is shown in Figure 4.7.1.11(a). Different material is used to draw the sample name, thus the drawing text function that is used in visualizing text for maximum and minimum will not be used. Sample name for the sample will located at position (x+20). Interaction that rotate 3D scatter plot and viewing the object from different views is added on mouse move as shown in Figure 4.7.1.12 (a). 3D scatter plot that visualized PCA score for the samples is shown in Figure 4.7.1.13 and Figure 4.7.1.14 (rotated view).

```
var pointCount = $data.unfiltered.length;
for (var i = 0; i < pointCount; i++) {
    var x = xScale($data.unfiltered[i]["x"]);
    var y = yScale($data.unfiltered[i]["y"]);
    var z = zScale($data.unfiltered[i]["z"]);
    console.log(x, y, z)
    var geometry = new THREE.SphereGeometry(1, 32, 32);
    console.log($data.unfiltered[i])
    var material = new THREE.MeshBasicMaterial();
    console.log(parseInt($data.unfiltered[i]["color"], 16))
    material.color.set(new THREE.Color(parseInt($data.unfiltered[i]["color"], 16)));

    var meshpoint = new THREE.Mesh(geometry, material);

    meshpoint.position.set(x, y, z);


    var canvas1 = document.createElement('canvas');
    var context1 = canvas1.getContext('2d');
    context1.font = "Bold 20px Arial";
    context1.fillStyle = "rgba(255,255,255,1)";
    context1.fillText($data.sample_list[i][i], 0, 100);

    // canvas contents will be used for a texture
    var texture1 = new THREE.Texture(canvas1)
    texture1.needsUpdate = true;

    var material1 = new THREE.MeshBasicMaterial({ map: texture1, side: THREE.DoubleSide });
    material1.transparent = true;

    var mesh1 = new THREE.Mesh(
        new THREE.PlaneGeometry(50, 20),
        material1
```

(a)

(b)

Figure 4.7.1.11 Visualization of sample inside 3D scatter plot by using sphere

dovirus - Visual Studio Code

TS visualizer.ts .../methylation_graph6_heatmap | api.py | manage.py | TS visualizer.ts .../methylation_

```
318                 new THREE.PlaneGeometry(50, 20),
319                 material1
320             );
321             mesh1.position.set(x + 20, y, z);
322             scatterPlot.add(mesh1);
323             scatterPlot.add(meshpoint);
324
325         }
326         renderer.render(scene, camera);
327         var paused = false;
328         var last = new Date().getTime();
329         var down = false;
330         var sx = 0,
331             sy = 0;
332
333         window.onmousedown = function (ev) {
334             down = true;
335             sx = ev.clientX;
336             sy = ev.clientY;
337         };
338         window.onmouseup = function () {
339             down = false;
340         };
341         window.onmousemove = function (ev) {
342             if (down) {
343                 var dx = ev.clientX - sx;
344                 var dy = ev.clientY - sy;
345                 scatterPlot.rotation.y += dx * 0.01;
346                 camera.position.y += dy;
347                 sx += dx;
348                 sy += dy;
349             }
350         }
```

(a)

(b)

Figure 4.7.1.12 Interaction that rotate the 3D scatter plot on move

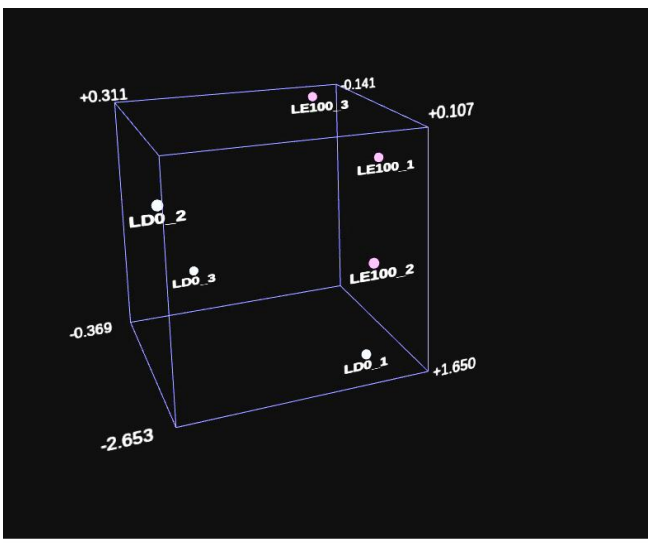


Figure 4.7.1.13 Interactive cube for PCA clustering

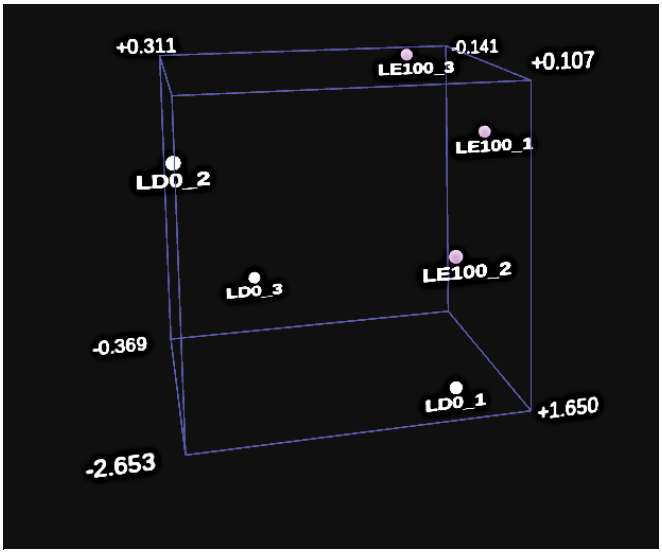


Figure 4.7.1.14 Interactive cube for pca clustering-Rotated view

SECTION 4.7.2- HEATMAP AND DENDROGRAM （CLUSTERING）

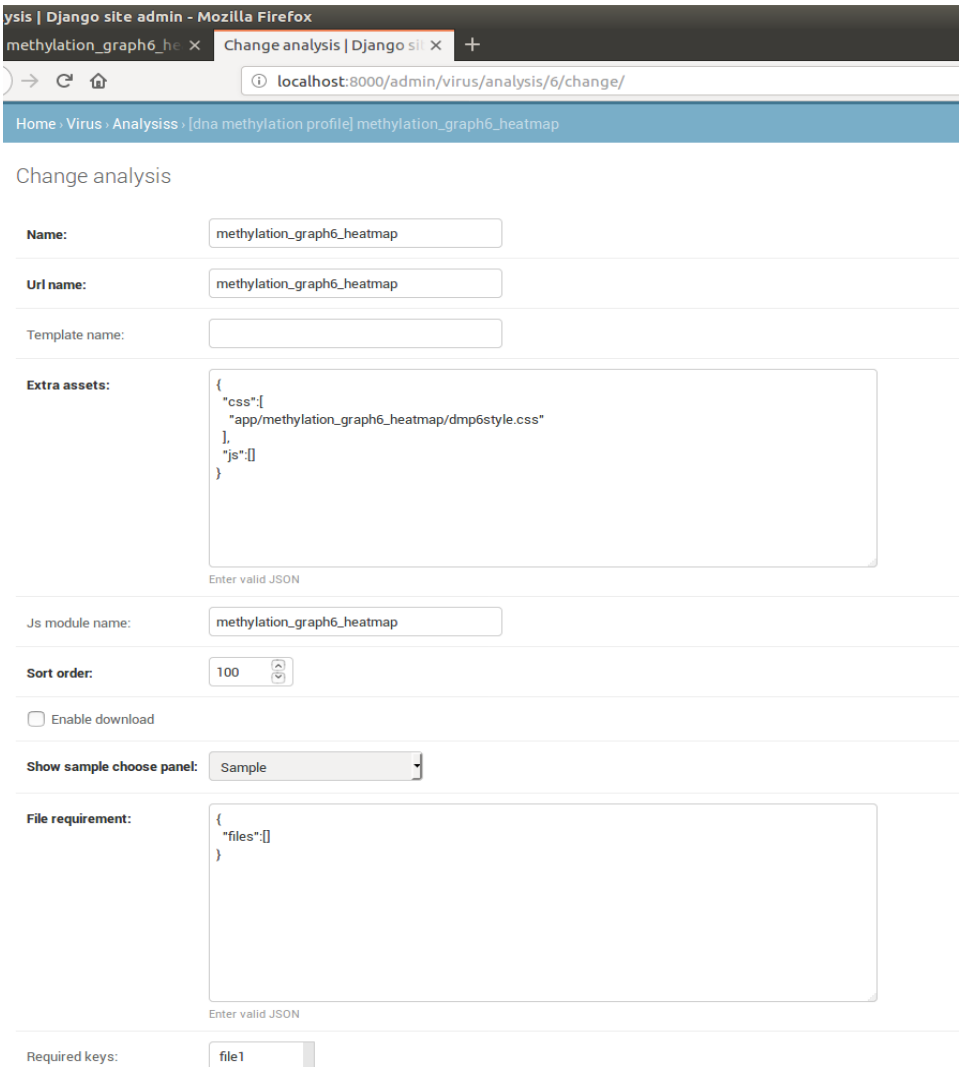


Figure 4.7.2.1 Setup of heatmap for methylation ratio

Data is grouped by sample and sorted by methylation ratio for data in each sample. To calculate the distance metric for the samples, the values for each sample is stored in dendrogramData and used to calculate the distance between each sample using Euclidean distance. The linkage of dendrogram is formed by getting the minimum between distance matrixes and forming a new distance matrixes. Euclidean distance of the samples is calculated by a plugin package.

Visualization of this sector is divided into two parts which is the visualization of heatmap and visualization of dendrogram. Heatmap is formed by rectangle. For every row in the data, a rectangle will be drawn. Data for heatmap is a matrix of sorted

methylation ratio in the position. Data structure of the data passed into heatmap is displayed in Figure 4.7.2.2 (a). Figure 4.7.2.3 (a) illustrates how the smooth color is declared for heatmap. The domain of the colors that is stored in linear scale is evenly distributed. Dendrogram is visualized using path. For every row in the data, a path will be drawn. Hover function is called when user moves the mouse onto the heatmap. Hover box that shows the extra information will be shown as Figure 4.7.2.4 (b). Hover function applied for heatmap will be similar to pie chart in the previous result. The colors go from blue to black and black to red. The color red in the heatmap means the methylation ratio is nearer to 1 while blue in the heatmap represents that the methylation ratio is closer to 0.

```
this.loadData("mCytosine", (d) => {

this.data.key_list = []
this.data.dendrogramData = []
var heatmapGraphLocal = d["data"]["methylation_ratio"]
this.data.sample_list = d["data"]["sample"]
console.log(heatmapGraphLocal)
var name = d3.nest()
    .key(function (d) { return d["sample"]; })
    .entries(heatmapGraphLocal)

var data = [];
console.log(name)
for (var i of name) {



  var info = {
      "sample": i["key"], ["values"]: d3.nest()
          .key(function (d) { return d["pos"]; })
          .entries(i["values"])
          .map(function (group) {
              return {
                  "pos": group.key,
                  "mean": d3.mean(group.values, function (d) { return d["value"]; }),
                  "values": group.values
              }
          })
  };

  let newObj = JSON.parse(JSON.stringify(info["values"].map(function (group) {
      return {
          "pos": group["pos"],
          "value": group["mean"],
          "sample": i["key"]
      }
  })));
  newObj.sort(function (a, b) {
      return (a["value"] - b["value"]);
  })
  this.data.key_list.push(newObj.map(item => item["pos"])
      .filter((value, index, self) => self.indexOf(value) === index));
  this.data.dendrogramData.push({ "sample": i["key"], "value_list": newObj.map(a => a["value"]) })

  data.push(newObj);
}
var merged = [].concat.apply([], data);
merged.sort(function (a, b) {
    return (a["value"] - b["value"]);
})
this.data.heatmapGraphData = merged
```

(a)

Figure 4.7.2.2 Data processing to visualize heatmap and dendrogram

```
var heatmapsvg = svg.append("g")
.attr('transform', 'translate(100 ,150)');

// var colours=["#8ad","#68b","#469","#247","#025", "#000000","#080000", "#100000","#160000","#200000","#240000]
 var colours=["#87ceeb","#79c","#79a","#68b","#57a","#469","#358","#247","#136","#025","#0000cc","#000099","#000066","#0000CC","#0000FF","#00000A","#000000","#0F0000",
 "#200000","#240000","#280000","#300000","#400000","#450000","#600000","#660000","#700000","#750100","#8e1a00","#990000","#CC0000","#db0000","#FF0000"]

var domainlist=[0]
var maxrange=1 / (colours.length)
var last=0
 for (var d_i=colours.length;d_i>=1;d_i--)
 {


     last+=maxrange
         domainlist.push(last)

 }
 var colorScale = d3.scaleLinear<string>()
 .domain(domainlist)
 .range(colours)
```

(a)

Figure 4.7.2.3 Visualization of heatmap- Initilize color

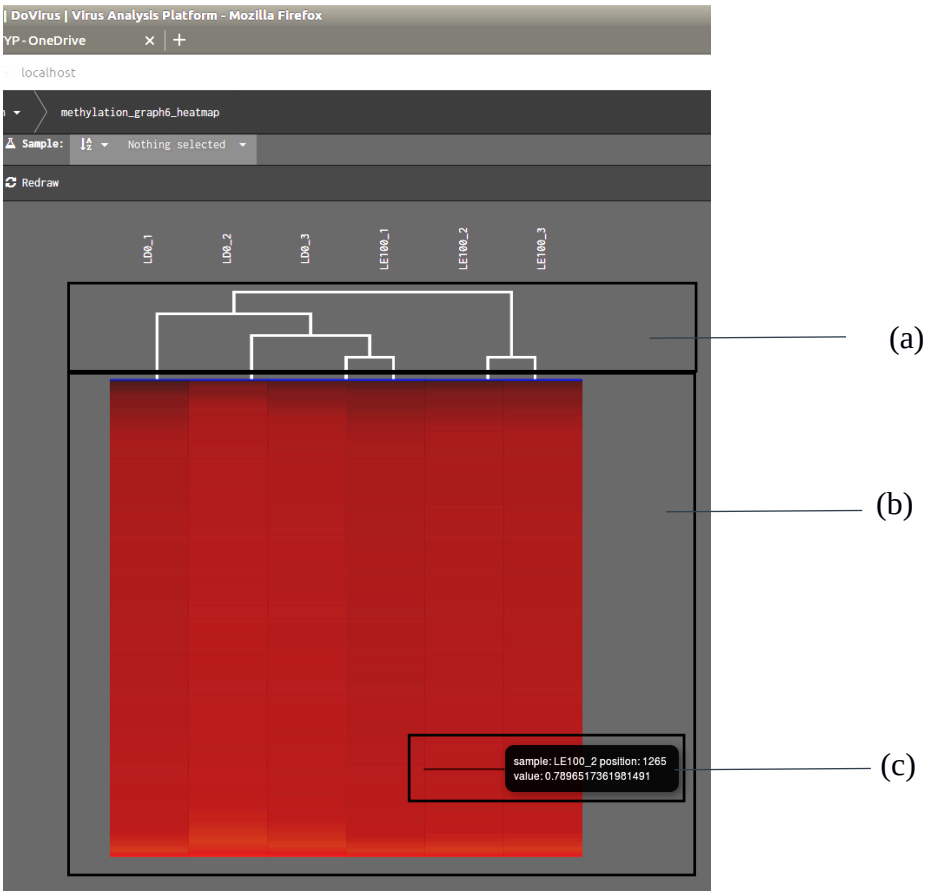


Figure 4.7.2.4 Heatmap of methylation ratio (a) dendrogram (b) heatmap (c) hover box

SECTION 4.8- IMPLEMENTATION ISSUES AND CHALLENGES

One of the implementation issues faced in the project is large input file. Browser cannot handle a huge input file. The file is tried to filter before it load. Only the columns that are going to be used in the graph should be read in each visualization for DNA methylation. However, filtering of the data in d3.tsv does not help the issue. D3.tsv read and store everything in data for call back function. The data filtering only will be done after call back function called. In short, d3.tsv is not an appropriate way to handle large data file. In contrast, Tabix is then found as a genomic indexed tool that index the bgzip file efficiently. It is useful in retrieving large data file with high speed. User can upload a brunch of bgzip file to the module. System is required to generate the tabix-indexed files.

However, tabix should be implemented in C, Java, Perl or Python. The visualization of the graph need to get data from tabix API that is implemented in Python.

Moreover, another implementation issues in the current project is SVG does not support html tables. However, the quality control table need to be visualize in table form. Therefore, html table template need to be added into the html part directly. The data will be handled in another JavaScript file instead of visualize.ts. Jquery.ajax() will be used to perform Ajax request and retrieve the data.

The platform is huge and required times to understand it well. It takes a lot of time to study about the biology concept and coding we have not previously learnt before. Interactive visualization for the DNA methylation required time to make it user friendly and good looking in design. It is a big challenge to design a good interface.

Besides, d3 version 4 is used in this project. Compare to d3 version 3 that install all of the d3 related module into the project, d3 version 4 only install small module that is needed. However, some of the functions such as brush.empty() is missing in d3 version 4.

D3.js is the main library that is used in the project. However, 3d cube cannot be visualized using d3.js. D3.js is only powerful to create interaction between user and 2D object. To solve this problem, three.js is used. Three.js is a powerful library that is always used in simulation and 3D model drawing. It provides some useful function to create interactive 3D graph including 3D scatter plot.

In addition, heatmap is drawn by a huge amount of data. The number of pixels of the rectangle is larger than the pixels that the browser can support. In order to draw heatmap, the data is grouped before visualization. Every position in the data is divided by 10k and the average position is retrieved to draw the heatmap.

CHAPTER 5:

CONCLUSION

As more and more people are concerned about health issues, researches on human health has received a lot of focus. DNA methylation is one main area that researchers focus on when they study about cancer. DNA methylation is a process of adding methyl groups to DNA molecule and form 5-methlcytosine (5mC). The relationship between DNA methylation and mutation is heavily studied. More and more researchers study DNA methylation for its connection with some chronic diseases that are caused by defective imprinting mechanisms. This resulted in rapid increase in DNA methylation data. It is becoming difficult for researchers to analyze the data, and visualization tool in DNA methylation can help with their research productivity, as well as help communicate their research output. The project develops an interactive online tool that visualize DNA methylation data at the same time provide publishable quality graphics and figure to researchers.

Analysis included in this project is stated as below:

a) Quality control by showing percentage of coverage cytosine
b) Quality control by showing distribution of different reads in coverage cytosine
c) Quality control by showing distribution of methylation ratio
d) Pie chart shows the percentage of methylated context for each sample
e) Stacked bar chart shows distribution of CG in low, intermediate and high level based on region
f) Heatmap and dendrogram visualize the overview of methylation ratio in each sample and dendrogram clusters samples
g) 3D scatter plot perform PCA and visualize the sample based on PCA score in 3D mode

There are some possible ways to improve the project. More important analysis can be added. Analysis like differential methylated region can be added to perform a completed analysis. Interaction like hover or zoom can be added to graph like PCA. It can also be visualized using 3D scatter plot or heatmap.

CHAPTER 6:

REFERENCE

1. AskUbuntu (2012). How do I install the latest Python 2.7.X or 3.X on Ubuntu? Available at: https://askubuntu.com/questions/101591/how-do-i-install-the-latest-python-2-7-x-or-3-x-on-ubuntu (Accessed 5 November 2017).
2. Bock,C., Reither,S., Mikeska, T., Paulsen, M., Walter, J., Lengauer,T.(2005) BiQ Analyzer: visualization and quality control for DNA methylation data from bisulfite sequencing, Bioinformatics[online], Volume 21, Issue 21, Pages 4067–4068, Available from: https://doi.org/10.1093/bioinformatics/bti652 (Accessed 5 November 2017).
3. Cardenas, A., et al., (2017). Persistent DNA methylation changes associated with prenatal mercury exposure and cognitive performance during childhood, Scientific Reports 7 , p. 288. Available from: https://www.nature.com/articles/s41598-017-00384-5 (Accessed 5 November 2017).
4. Carr, I. M., Valleley, E. M. A., Cordery, S. F., Markham, A.F., Bonthron, D. T. (2007). Sequence analysis and editing for bisulphite genomic sequencing projects, Nucleic Acids Research[online], Volume 35, Issue 10, Pages e79, Available from: https://doi.org/10.1093/nar/gkm330 (Accessed 5 November 2017).
5. Chen,P.Y. et al., (2013) Intrauterine calorie restriction affects placental DNA methylation and gene expression , Physiological Genomics [online], Volume 45, Issue 14, Pages 565-576. Available from: http://physiolgenomics.physiology.org/content/45/14/565 (Accessed 5 November 2017).
6. 'DNA methylation'. Wikipedia, Wikimedia Foundation. [Online]. Available from: https://en.wikipedia.org/wiki/DNA_methylation (Accessed 5 November 2017).

7. Justin Ellingwood (2016). How to use PostgresSQL with your Django application on Ubuntu 16.04. Available from: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-16-04 (Accessed 5 November 2017).
8. K., R. (2016). How to Install Latest Node.js on Ubuntu via Apt-Get - TecAdmin. Available at: https://tecadmin.net/install-latest-nodejs-npm-on-ubuntu/# (Accessed 5 November 2017).
9. Kumaki,Y., Oda,M., Okano,M.(2008) QUMA: quantification tool for methylation analysis, Nucleic Acids Research [online], Volume 36, Issue suppl_2, Pages W170–W175, Available from: https://doi.org/10.1093/nar/gkn294 (Accessed 5 November 2017).
10. Li, H. (2011). Tabix: fast retrieval of sequence features from generic TAB-delimited files [online]. Bioinformatics, 27(5), 718–719. Available from: http://doi.org/10.1093/bioinformatics/btq671 (Accessed 5 November 2017).
11. Li, S.C., Jia, W.L., Li, H.C. (2017). An Integrated Oncovirus Analysis Platform [Online]. Available from: http://www.dovirus.com (Accessed 5 November 2017) .
12. Lister, R., Pelizzola, et al., (2017). Human DNA methylomes at base resolution show widespread epigenomic differences. Nature. 462(7271): 315-22. Available from: https://www.ncbi.nlm.nih.gov/pubmed/19829295 (Accessed 11 November 2017).
13. Mallona, I., Díez-Villanueva, A. and Peinado, M. (2014). Methylation plotter: a web tool for dynamic visualization of DNA methylation data, Biology and Medicine [online] , Volume 9, Issue 11, Available from: https://doi.org/10.1186/1751-0473-9-11 (Accessed 5 November 2017).
14. National Center for Biotechnology. (n.d.). GenBank:GenBank Overview [online]. Available from: https://www.ncbi.nlm.nih.gov/genbank/(Accessed 12 August 2017).

15. Palani, G.S. (2011). Summary of web application testing methodologies and tools [Online]. Available from: https://www.ibm.com/developerworks/library/wa-webapptesting/ (Accessed 5 November 2017).
16. Pardo, C.E., Carr, I.M., Hoffman, C.J., Darst, R.P., Markham, A.F., Bonthron, D.T., Kladde, M.P. (2011). MethylViewer: computational analysis and editing for bisulphite sequencing and Methyltransferase Accessibility Protocol for Individual Templates (MAPit) projects, Nucleic Acids Research [online], Volume 39, Issue 1, Pages e5, Available from: https://doi.org/10.1093/nar/gkq716 (Accessed 5 November 2017).
17. Phil-pedruco's Block. 10 March 2018. 3D scatter plot using three.js. Available from: http://bl.ocks.org/phil-pedruco/9852362 / (Accessed 18 April 2018).
18. SaltyCrane Blog. 15 Febuary 2010. How to install pip on Ubuntu. Available from: https://www.saltycrane.com/blog/2010/02/how-install-pip-ubuntu/ (Accessed 5 November 2017).
19. Shao, C. et al, (2014). Epigenetic modification and inheritance in sexual reversal of fish. Genome Research [online], 24(4), 604–615. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3975060/pdf/604.pdf (Accessed 25 November 2017).
20. Sproul, D., Meehan, R.R. (2013). Genomic insights into cancer-associated aberrant CpG island hypermethylation. Briefings in Functional Genomics [online], Volume 12, Issue 3, Pages 174–190, Available from: https://doi.org/10.1093/bfgp/els063 (Accessed 5 November 2017).
21. Xin, Y., Chanrion, B., O'Donnell, A. H., Milekic, M., Costa, R., Ge, Y., & Haghighi, F. G. (2012). MethylomeDB: a database of DNA methylation profiles of the brain. Nucleic Acids Research [online], 40(Database issue), D1245–D1249. Available from: http://doi.org/10.1093/nar/gkr1193 (Accessed 5 November 2017).
22. Zhang, Y., et al., (2011). QDMR: a quantitative method for identification of differentially methylated regions by entropy. Nucleic Acids Research [online], 39 (9): e58. Available from:

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3089487/ (Accessed 25 November 2017).

APPENDICES:

APPENDICES A

WEEKLY REPORT

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 2, 3	**Study week no.: 2**
Student Name & ID: CHEAH ZHAO QIN 140ACB2224	
Supervisor:DR. NG YEN KAOW	
Project Title: INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

a) Modified UploadFile class and allow user to upload bgzip file. The bgzip file will be processed and tabix indexed file will be generated. The program will retrieve the file information through tabix indexed file. User can upload multiple file into one sample directory.
b) The bgzip uploaded file details is added into TabixAPI class for reference.
c) Data for table 1-quality control is generated and stored into 5 file (2x, 5x, 10x, 100x) when file is uploaded. When file is uploaded into sample directory, the details will be updated into the file generated.

2. WORK TO BE DONE

a) Visualization of table 1-quality control
   - Display quality of uploaded file in a table.
b) Data processing for second graph (Distribution of depth coverage)
   - Process data that used for visualization of second graph (Distribution of depth coverage).

3. PROBLEMS ENCOUNTERED

a) The input file that received from current researcher is too large. It used 1 hour to process a single sample file.

4. SELF EVALUATION OF THE PROGRESS

a) Solved the problem of processed input file. Every graph that visualized in the project depends on the data processing using python.
b) Finished processed data for table 1-quality control.
c) Progress is slower than expected

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 2, 3	**Study week no.: 3**
Student Name & ID: CHEAH ZHAO QIN 140ACB2224	
Supervisor:DR. NG YEN KAOW	
Project Title: INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

a) TabixAPI is checked when the uploaded is deleted. If no more sample file in TabixAPI, the TabixAPI model will be deleted.
b) Table 1 is visualized by using DataTable function of javascript. Selection for number of reads is added.

2. WORK TO BE DONE

a) Data processing for second graph (Distribution of depth coverage)
b) Visualization and Interaction of second graph (Distribution of depth coverage)

3. PROBLEMS ENCOUNTERED

-

4. SELF EVALUATION OF THE PROGRESS

a) Redundancy Tabix API model is deleted when uploaded file is deleted.
b) Table that shows the quality of the sample is visualized successfully.
c) Progress is slower than expected

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 2, 3	**Study week no.: 4**
Student Name & ID: CHEAH ZHAO QIN 140ACB2224	
Supervisor:DR. NG YEN KAOW	
Project Title: INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

a) Data processing of second graph (Distribution of depth coverage). Data is processed when sample file is uploaded. Processed data is saved into a file that will be used in visualization.
b) Visualization of second graph (Distribution of depth coverage) is improved based on the visualization that done before. Last semester, second graph (Distribution of depth coverage) is visualized with incorrect data loading process.
c) Interaction of second graph (Distribution of depth coverage) is added. Hover box that show the detailed information of the graph will be shown when hovering. Click on legend will show only the selected graph. For example if only frequency graph to be shown then click on the color that c

2. WORK TO BE DONE

a) Data processing of third graph- Distribution of methylation ratio
b) Visualization of third graph- Distribution of methylation ratio
c) Interaction of third graph- Distribution of methylation ratio

3. PROBLEMS ENCOUNTERED

a) Zoom function cannot be applied on second graph (Distribution of depth coverage) due to multiple graphs are appended to svg.

4. SELF EVALUATION OF THE PROGRESS

a) Data processing of second graph (Distribution of depth coverage) is completed.
b) Visualization and Interaction of second graph (Distribution of depth coverage) are completed.

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 2, 3	**Study week no.: 5**
Student Name & ID: CHEAH ZHAO QIN 140ACB2224	
Supervisor:DR. NG YEN KAOW	
Project Title: INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

a) Data processing of third graph- Distribution of methylation ratio. Data is processed when sample file is uploaded. Processed data is saved into a file that will be used in visualization.
b) Visualization of third graph is improved based on the visualization that done before. Last semester, third graph is visualized with incorrect data loading process. Color of graph3 is changed.
c) Interaction of third graph is added. Hover box that show the detailed information of the graph will be shown when hovering. Circle and line will be shown based on the mouse position. User can select number of reads, n and filter out the distribution for number of reads smaller than n.

2. WORK TO BE DONE

a) Data processing for fourth graph- percentage of methylated context in each sample
   - Process data that used for visualization of fourth graph.

b) Visualization and interaction of fourth graph.
   - Pie chart will be visualized for fourth graph and hovering interaction will be added.

3. PROBLEMS ENCOUNTERED

a) Zoom function cannot be applied on third graph (Distribution of methylation level) due to multiple graphs are appended to svg.

4. SELF EVALUATION OF THE PROGRESS

a) Data processing of third graph (Distribution of methylation level) is completed.
b) Visualization and Interaction of third graph (Distribution of methylation level) are completed.

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 2, 3	**Study week no.: 6**
Student Name & ID: CHEAH ZHAO QIN 140ACB2224	
Supervisor:DR. NG YEN KAOW	
Project Title: INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

a) Data processing of fourth graph- percentage of methylated context in each sample
b) Visualization and interaction of fourth graph. Fourth graph is visualized using pie chart and interaction is added on hovering. When user hover on the pie chart, the radius for sector of the pie chart will be 5px larger. Detail information will show in hover box.

2. WORK TO BE DONE

a) Data processing for fifth graph- heatmap that show methylation level
   - Process data that used in visualization of fifth graph.

b) Visualization and interaction of fifth graph.
   - heatmap will be visualized for fifth graph to show then overall methylation level and hovering interaction will be added.

3. PROBLEMS ENCOUNTERED

a) The matrix size that will be used in fifth graph is too large. Browser cannot support and browser will be not responding.

4. SELF EVALUATION OF THE PROGRESS

a) Data processing for fourth graph is done.
b) Visualization and interaction for fourth graph is completed as expected.
c) Data processing for fifth graph- heatmap is still processing.

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 2, 3	**Study week no.: 7**
Student Name & ID: CHEAH ZHAO QIN 140ACB2224	
Supervisor:DR. NG YEN KAOW	
Project Title: INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

a) Data processing of fifth graph-heatmap is completed.
   - Matrix size of data that used to visualize heatmap is reduced by dividing with a factor. The factor used to reduce matrix size is 10000.

b) Visualization and interaction of heatmap is completed. Hover box will be shown when hovering.

2. WORK TO BE DONE

a) Data processing for fifth graph-dendrogram
b) Visualization of fifth graph-dendrogram

3. PROBLEMS ENCOUNTERED

a) Although heatmap is successfully visualized, the performance of browser will slow down significantly.

4. SELF EVALUATION OF THE PROGRESS

a) Data processing of fifth graph- heatmap is completed.
b) Visualization and interaction of fifth graph- heatmap is completed.

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 2, 3	**Study week no.: 8**
Student Name & ID: CHEAH ZHAO QIN 140ACB2224	
Supervisor:DR. NG YEN KAOW	
Project Title: INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

a) Data processing of fifth graph-denderogram is completed.
   - Distance matrix is calculated using Euclidean distance.

b) Visualization of denderogram is completed. Text of samples are added above the dendrogram.

2. WORK TO BE DONE

a) Data processing for sixth graph-3D scatter plot for PCA
b) Visualization of sixth graph-3D scatter plot for PCA

3. PROBLEMS ENCOUNTERED

a) Although denderogram and heatmap is successfully visualized, the performance of browser will slow down significantly.

4. SELF EVALUATION OF THE PROGRESS

a) Data processing of fifth graph- denderogram is completed.
b) Visualization and interaction of fifth graph- denderogram is completed.

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 2, 3	**Study week no.: 9**
Student Name & ID: CHEAH ZHAO QIN 140ACB2224	
Supervisor:DR. NG YEN KAOW	
Project Title: INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

a) Data processing of sixth graph-3D scatter plot for PCA is completed.
   - PCA scoring of the samples is calculated

b) Visualization and interaction of sixth graph-3D scatter plot for PCA is completed. User can drag to rotate the 3D scatter plot.

2. WORK TO BE DONE

a) Data processing for seventh graph-Fraction of CG in each region

b) Visualization of seventh graph- Fraction of CG in each region

3. PROBLEMS ENCOUNTERED

a) D3.js that is used for all the visualization before does not support 3D visualization. A new library is used and studied.

4. SELF EVALUATION OF THE PROGRESS

a) Project is going as expected.

_____ _____

Supervisor's signature Student's signature

–

Trimester, Year: 2, 3	**Study week no.: 10**
Student Name & ID: CHEAH ZHAO QIN 140ACB2224	
Supervisor:DR. NG YEN KAOW	
Project Title: INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

a) Modified UploadFile class and allow user to upload gzip file. Gzip file will be processed and related files for analysis are generated. User can only upload one file for one sample.
b) Data for all the existing is generated and stored respective file when file is uploaded. When file is uploaded into sample directory, the details will be updated into the file generated.

2. WORK TO BE DONE

a) Data processing for seventh graph-Fraction of CG in each region
b) Visualization of seventh graph- Fraction of CG in each region

3. PROBLEMS ENCOUNTERED

4. SELF EVALUATION OF THE PROGRESS

a) Unexpected changes in input file style.
b) A new input reading file method slow down the progress of the project

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I)

Trimester, Year: 2, 3	**Study week no.: 11**
Student Name & ID: CHEAH ZHAO QIN 140ACB2224	
Supervisor:DR. NG YEN KAOW	
Project Title: INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

a) Make changes on 3D scatter plot for PCA. Text for axis and samples are added.
b) Data processing of fifth graph- Heatmap and 3D scatter plot is changes.

2. WORK TO BE DONE

a) Data processing for seventh graph-Fraction of CG in each region
b) Visualization of seventh graph- Fraction of CG in each region

3. PROBLEMS ENCOUNTERED

4. SELF EVALUATION OF THE PROGRESS

c) Finished processed data for fifth graph-heatmap and dendrogram.
d) Progress is slower than expected

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 2, 3	**Study week no.: 12**
Student Name & ID: CHEAH ZHAO QIN 140ACB2224	
Supervisor:DR. NG YEN KAOW	
Project Title: INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

a) Data processing for seventh graph-Fraction of CG in each region
b) Visualization and interaction for seventh graph-Fraction of CG in each region

2. WORK TO BE DONE

a) Report for FYP2

3. PROBLEMS ENCOUNTERED

4. SELF EVALUATION OF THE PROGRESS

e) 20% of the report is done

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 2, 3	**Study week no.: 13**
Student Name & ID: CHEAH ZHAO QIN 140ACB2224	
Supervisor:DR. NG YEN KAOW	
Project Title: INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES	

1. WORK DONE [Please write the details of the work done in the last fortnight.] a) Report is done.
2. WORK TO BE DONE a) Presentation and slide need to be prepared
3. PROBLEMS ENCOUNTERED
4. SELF EVALUATION OF THE PROGRESS a) Although some of the graph still need to be improved, the time to submit the report is just around the corner. Thus, some of the visualization cannot be done.

_____ _____

Supervisor's signature Student's signature

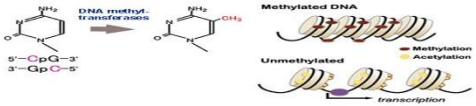
POSTER:

INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES

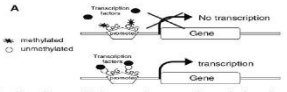
Cheah Zhao Qin | Supervised by: Dr Ng Yen Kaow

Introduction

- As more and more people concern about health issues, research based on human health is focused. DNA methylation is one main concept that researchers focus on when they study about cancer. DNA methylation is a process of adding methyl groups to DNA molecule and form 5-methlcytosine (5mC).



- Some of the DNA methylation may cause cancer. All cells have similar DNA sequences. Each cell have a unique methylation method express certain part of gene and perform different function. Promoter region contain regulatory element that control transcription of gene. DNA methylation status has a strong inverse correlation with gene expression. DNA methylation pattern changes in cancer cell. The ratio of the methylation and DNA demethylation are deeply balanced in normal cell. However, the regulation is damaged in cancer cell.



- In normal cells, there will be an absence of methylated cytosine in the promoter region. On the other hand, the cytosine in promoter region of cancer cell is methylated and results in no transcription of gene. Some of the transcription helps to repair mutation of the cell. Due to transcription of gene silencing in tumor gene promoter, mutation in cancer cell increased.
- The increase in projects on DNA methylation has led to an increase in available genomic and epigenetic data. Thus, interactive tool in visualizing the data is needed.

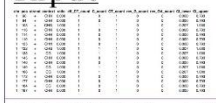
Problem

- However, lack of available tools to visualize huge genomic data and display interesting interfaces slows down the researcher's work and degrade the presentation of the researcher.

Method and Result

- Develop an interactive online tool for methylation study using d3.js, typescript and SASS that help users control the quality of the input, display the depth of methylated Cytosine, show overview of DNA methylation and visualize clustering and PCA
- The visualization graph will help user discover the methylation pattern and the differential methylation pattern. Thus, the graph help user to understand the relationship between DNA methylation and the regulation of gene.

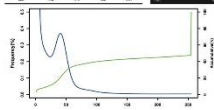
Input

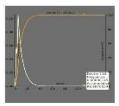


The interactive tool required a BSMAP standard table as input. BSMAP standard output shows the mapping of bisulphite sequencing in DNA methylation. The BSMAP standard input should be uploaded in bgzip file.

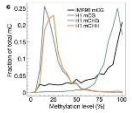
SAMPLE AND RESULT (Quality Control)

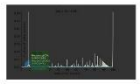
Three ratio of coverage depth-
The ratio of effective cytosine count for each context versus total methylated cytosine shows the quality of the sample. If the sample have a poor segmentation and result in too much repair on the segment, the ratio of coverage depth of the sample will be low.





Frequency of such depth appears in the sample.-
The frequency of depth in the sample shows the effective coverage in each sequence of sample.

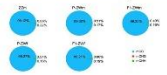




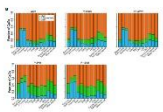
Distribution of methylation level in each sequence context-
It used the ratio of the count of specific methylation level and total effective cytosine to generate the graph. The graph give user an overview of the methylation level in each ratio for every sequence data in different context.

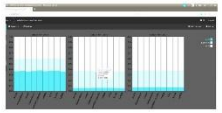
Ratio of the coverage cytosine, distribution of the coverage depth of cytosine and distribution of methylation level in each sequence context determine the quality of input. Poor sample will show low covered cytosine and poor analytical result.

SAMPLE AND RESULT(Overview of DNA Methylation)



Percentage of methylated cytosine in mCG, mCHG and mCHH-
It visualized by using the number of cytosine in reads/ effective coverage cytosine. In the sequence, all the remaining unmethylated cytosine will be converted to Thymine. Thus, using the number of cytosine in each reads will tell user methylation condition of each context.

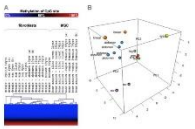




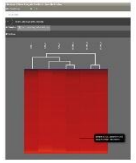
Fraction of CpG in low (<0.25), intermediate (> 0.25 and <0.75), and high (> 0.75) methylation levels in different genomic elements. The position of the genomic element will be provided and methylation level of each genomic element will be calculated and classified into low, intermediate and high.

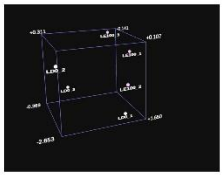
Overview of DNA methylation shows the overview of methylation status, methylation level, methylcytosine density for the sample. The overview helps to determine the contribution of DNA methylation to variability of cell and phenotypes.

SAMPLE AND RESULT (CLUSTERING AND PCA)



Clustering and PCA analysis of CpG island.
PCA analysis shows the scoring of the sample in the right figure. CpG island required 2 samples and above to do the clustering. Methylation pattern is identified and cluster using PCA analysis. From that, user can identified the characteristics of methylation pattern in each sample.





PCA analysis and heatmap for the sample are visualized in the project.

Discussion

- The project developed a tool that will generate complete analysis graph that visualize the huge data of DNA methylation. The graphs help in explaining how DNA methylation affects the regulation of gene expression and how it influence the phenotype.
- Interactive graphs are the product of the project. Interactive graph shows the genomic data neatly. Static graph visualize the genomic data in multiple graph and make the visualization look messy.

BCS (Hons) Computer Science
Faculty of Information And Communication Technology (Perak Campus), UTAR.

PLAGIARISM CHECK RESULT

INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES

ORIGINALITY REPORT

11%	10%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	eprints.utar.edu.my Internet Source	8%
2	www.cse.dmu.ac.uk Internet Source	<1%
3	M. Okano. "QUMA: quantification tool for methylation analysis", Nucleic Acids Research, 05/19/2008 Publication	<1%
4	Sproul, D., and R. R. Meehan. "Genomic insights into cancer-associated aberrant CpG island hypermethylation", Briefings in Functional Genomics, 2013. Publication	<1%
5	genome.cshlp.org Internet Source	<1%
6	University of Tennessee, Knoxville Publication	<1%
7	www.nfpa.org Internet Source	<1%

Figure I Plaglarism result

INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES

by Cheah Zhao Qin

Submission date: 18-Apr-2018 01:12AM (UTC+0800)
Submission ID: 948522691
File name: fypReport22_1.pdf (9.63M)
Word count: 17879
Character count: 97207

Figure II Plaglarism result

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	CHEAH ZHAO QIN
ID Number(s)	14ACB02224
Programme / Course	BACHELOR OF COMPUTER SCIENCE (HONS)
Title of Final Year Project	INTERACTIVE ONLINE TOOL FOR METHYLATION STUDIES

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 11 % **Similarity by source** Internet Sources: 10 % Publications: 8 % Student Papers: - %	
Number of individual sources listed of more than 3% similarity: 1	

Parameters of originality required and limits approved by UTAR are as Follows:

(i) Overall similarity index is 20% and below, and
(ii) Matching of individual sources listed must be less than 3% each, and
(iii) Matching texts in continuous block must not exceed 8 words

Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

_____	_____
Signature of Supervisor	Signature of Co-Supervisor
Name: _____	Name: _____
Date: _____	Date: _____

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (PERAK CAMPUS)

CHECKLIST FOR FYP2THESIS SUBMISSION

Student Id	1402224
Student Name	CHEAH ZHAO QIN
Supervisor Name	DR. NG YEN KAOW

TICK (√)	**DOCUMENT ITEMS** Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Title Page
	Signed form of the Declaration of Originality
	Abstract
	Table of Contents
	List of Figures (if applicable)
	List of Tables (if applicable)
	List of Symbols (if applicable)
	List of Abbreviations (if applicable)
	Chapters / Content
	Bibliography (or References)
	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
	Poster
	Signed Turnitin Report (Plagiarism Check Result – Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report. _____ (Signature of Student) Date:	Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction. _____ (Signature of Supervisor) Date: