Developing a ChIP-seq pipeline that analyzes the human genome and its repetitive sequences

Helena Ishak

Degree project in bioinformatics, 2017
Examensarbete i bioinformatik 45 hp till masterexamen, 2017
Biology Education Centre, Uppsala University, and Department of Medical Biochemistry and Biophysics, Karolinska Institutet/SciLifeLab Stockholm
Supervisor: Simon Elsässer
Abstract

Repetitive sequences make up a large part of the human genome but for a long time they have been misunderstood as nonfunctional ‘junk’. Accumulating evidence suggest that they may contain useful functional information for the cell which we are only beginning to understand. Transposable elements (TEs), which constitute most of the repetitive families, are mobile ‘parasitic’ genetic elements that create mutations and insertions in the host genome, hence modifying the epigenetic and genetic landscape of the cell, and creating genomic instability. The eukaryotic cell has therefore evolved a variety of mechanism to silence and restrict the activity of TEs, which are believed to be the basis for known epigenetic gene regulatory mechanisms such as histone modification.

Chromatin immunoprecipitation followed by sequencing (ChIP–seq) is used to analyze the relationship between repetitive families and different proteins, such as histones. Many time-consuming steps then take place to analyze the ChIP-seq data. Also, programming skills and bioinformatic skills may be required for this analysis. Standard programs have been developed for analyzing the epigenome over genes and non-repetitive regions of the genome, while repetitive regions are typically excluded from those analyses. Therefore, the goal of this project is to develop an easy-to-use pipeline that makes the automated ChIP-seq analysis for repetitive sequences possible for a broad audience. The pipeline both analyses the enrichment of repetitive families in experiments containing different histone modifications, but also performs a standard analysis on the entire genome.

The pipeline is built to be run on a server called UPPMAX which provides a large processing power and memory space, as well as having all the programs necessary for the pipeline available. The pipeline was developed in the scripting language bash and consist of five scripts that together collect the data from the NIH Epigenomic Roadmap project based on the client interest. The pipeline processes the data and maps the experiments to the human genome including the repetitive parts, as well as to a metagenome containing a complete set of repetitive sequences. It then performs different robust analysis algorithms to create tables with the enrichment values of the repetitive families for each dataset.

The pipeline was tested and demonstrated on published ChIP-seq data in order to probe the correlation between transposable elements and histone modifications. Samples of different tissues and cell types containing four different histone modifications was used to test the pipeline. Two of these histone modifications are related to gene activation; H3K4me3 and H3K27ac, and the other two selected histone modifications are related to the repression of gene expression; H3K9me3 and H3K27me3. The tested pipeline generated quality scores for each of the datasets telling if the data was of good quality, as well as heatmaps and average enrichment profiles showing the overall enrichment for each experiment. It also generated more detailed enrichment tables which show the enrichment of each repetitive family in the analyzed experiments. This was then shown as a heatmap to show easy-analyzed correlations between histone modifications and repetitive families.
Developing a ChIP-seq pipeline that analyzes the human genome and its repetitive sequences

Popular Science Summary

Helena Ishak

DNA carries the genetical information (genome) for almost all organisms. The DNA has similar patterns between same organisms, but is unique for every living organism. However, all cells in one human body contain the same DNA. The genome contains genes that can be translated into proteins which have different functions in the human body. It is these proteins that build up all the features of a human. The genome is packed around histone proteins, forming chromatin, which are then packed together into a chromosome. Depending on how the histone proteins pack the genome, it affects the translation of genes; this is referred to as gene expression.

Transposable elements (TEs) are repetitive sequences that cover a large part of the human genome. These elements are capable to mobilize in the genome. Because of this, they are affecting nearby gene’s expression, genomic function and genomic structure. They also cause a genomic instability which can lead to genetic abnormalities and diseases. As a defense to rampant genetic instability, eukaryotic cells have developed an epigenetic mechanism such as histone modification to silence the activation of TEs. This means that a modification on the histone proteins can change the way the genome is packed around them, which then affect the genes expression without changing the DNA.

To analyze DNA, such as repetitive sequences, it must be sequenced. This means that the DNA is split into pieces where each piece is partly read and finally assembled back together as a puzzle or by using a reference genome as a template. To understand the interaction between DNA and histone modifications, the DNA, which is wrapped around histones, is cut into fragments and the DNA fragments bonded with the histones of interest are selected and sequenced.

The process of analyzing large amount of sequence data into meaningful outputs requires many steps which take long time and requires skills in programming and bioinformatics. Therefore, the goal of this project is to develop a pipeline that automatically process the sequencing data with focus on repetitive elements, and generates enrichment tables and visualizations presenting the correlation between repetitive families and different histone modifications. The pipeline can also be used to process the entire genome including ‘regular’ genes and non-repetitive sequences. The pipeline can analyze two types of datasets: the data that the user generated himself; or published data that the pipeline automatically generates from the NIH Epigenomic Roadmap project based on the user’s choice.

The pipeline can be used in many different types of researches. For example, apart from using the pipeline to study the interaction between TEs and histone modifications, it can also be used in the study of diseases. Researches has shown that there is a link between histone modifications and cancer, therefore it can be used to study the difference in histone modifications between cancer cells and normal cells. This information can be used to develop cancer treatment as well as to follow up a patience’s treatment period.
# Contents

1 Introduction ................................................................................................................................................... 1  
2 Materials and Methods ................................................................................................................................. 3  
   2.1 Data ......................................................................................................................................................... 3  
      2.1.1 Experiments .................................................................................................................................. 3  
   2.2 Tools ....................................................................................................................................................... 4  
3 Results .......................................................................................................................................................... 6  
   3.1 Script1 – Creates the sample sheet ......................................................................................................... 8  
   3.2 Script2 – Download data ....................................................................................................................... 10  
   3.3 Script3 – Map & Process data ............................................................................................................... 12  
   3.4 Script4 – Calculate the histone/input ratio percentage of reads per repetitive family ..................... 17  
   3.5 Script5 – Calculate histone/input mean of counts per repetitive sequence ........................................ 19  
   3.6 FastQC .................................................................................................................................................... 21  
   3.7 Ngs.plot ................................................................................................................................................. 23  
   3.8 Heatmap .................................................................................................................................................. 25  
4 Discussion ...................................................................................................................................................... 27  
   4.1 Heatmap .................................................................................................................................................. 27  
   4.2 Future development ............................................................................................................................... 28  
5 Acknowledgments .......................................................................................................................................... 29  
6 References ..................................................................................................................................................... 30  
7 Appendix ....................................................................................................................................................... 33  
   Appendix 1 ................................................................................................................................................... 33
1 Introduction

More than half of the human genome contains repetitive DNA sequences. For a long time, these sequences have not caught much attention in research because they were believed to be nonfunctional ‘junk’. However, recent studies have shown that some repetitive sequences seem to affect the epigenetic and genetic landscape, modulating expression of endogenous genes. Therefore, these misunderstood sequences may have important regulatory and evolutionary functions (1, 2).

A large part of repetitive sequences is contributed by so-called transposable elements (TE). TEs cover around 45% of the human genome (1, 3). They are mobile genetic elements that can ‘jump’ to different locations in the genome. Based on their mobilization method, they are called differently. DNA Transposons move location by a "cut-and-paste" mechanism, meaning they are cutting themselves out of one location in the genome and paste themselves into another. Retrotransposons use the "copy-and-paste" method, meaning they are creating a copy of themselves and paste the copy in another location (4). Thus, retrotransposons possess the unique ability to multiply in copy number in a genome. Mobilization events can create deletions, insertion and duplication in the genome, resulting in structural and functional changes in the DNA sequence and its epigenetic layer of regulation. Thus, TEs are playing an important role in the genomic evolution and regulation. In a new location, they for example may affect the expression of nearby genes (5, 6). The genome can also become instable due to the activation of TEs, leading to genetic abnormalities and diseases such as cancer (7). According to the genome defense hypothesis, the host genome has developed an epigenetic self-defense mechanism that is silencing the activation of TEs by histone modifications, to avoid this harmful instability (5). Epigenetics represent the modification regulating the gene expression without creating any change in the DNA (8). Histone proteins package the eukaryotic genome into so-called chromatin, providing multiple layers of compaction and regulation. Posttranslational modifications (PTMs, or ‘mark’) such as methylation and acetylation on histone are thought to be key components of epigenetic regulation (9).

To analyze how the epigenetic layer (e.g. in form of histone PTMs) are regulating gene expression, a method called chromatin immunoprecipitation followed by sequencing (ChIP–seq) is used. DNA is wrapped around a histone protein core in the fundamental repeating unit of chromatin, the nucleosomes. In a ChIP experiment, this chromatin is either sheared using sonication or digested enzymatically into mononucleosomal fragments (10). The resulting pool represents the entire genome. This pool was then sequenced and used as a control sample, called input. Antibodies are then used on the same input pool to pull down specific histone modifications and select histones of interest. These selected DNA fragments are then sequenced and used as histone modification samples. We call it an ‘enrichment’ when the antibody pulldown results in the overrepresentation of specific parts of the genome (thus the corresponding fragments are sequenced much more often), versus a ‘depletion’ of other parts.

ChIP-Seq data is analyzed through several steps. For example, the next generation (short read) sequencing data has to be pre-processed to exclude low quality reads (which are likely errors), the sequences have to be mapped to a reference genome, the data has to be normalized in order to make different samples comparable, algorithms have to be used to find enrichment of different genes etc. Even more steps are needed to analyze repetitive sequences. To manually analyze ChIP-
seq is time consuming, needs patience, a lot of knowledge about the parameters to use, as well as requires bioinformatic skills and some programming skills. The additional calculations used to find enrichments of repetitive sequences may be nearly impossible to do manually because of the large amount of data.

Standard programs have been developed for analyzing the epigenome over genes and non-repetitive regions of the genome, while repetitive regions are typically excluded from those analyses. Therefore, the goal with this project is to create a pipeline (11) that will, by a few command lines, automatically process ChIP-Seq data from the primary sequencing output to various useful plots and enrichment tables for repetitive elements and the entire genome. This pipeline both performs an analysis on the whole human genome and on the repetitive sequence regions. Such generated results could then be used to answer several research questions. The pipeline will analyze each ChIP-Seq dataset with respect to a matched input dataset; a unique feature when compared to published pipelines. This feature is crucial for an artifact-free, robust and biologically meaningful analysis of repetitive sequences.

Eventually, the pipeline could, for example, be used to understand how different histone modifications affect expression of different genes and repetitive sequences. It could be used to see if there is a difference in the DNA-histone modification interaction in different tissues, and for measuring the density of PTMs on histones. The pipeline could also be used to find where specific proteins such as the transcription factors (TFs) binds/interacts with the repetitive elements, or to find genes and repetitive elements that may be activated together. Also, it would be useful in the study of different genomic diseases (12). For example, some researches have shown that epigenetic mechanisms are linked with different types of cancers (8). Therefore, the pipeline can be used to study the difference in histone modifications between cancer cells and normal cells. This can then be used to detect histone modifications that regulates the activation or repression of cancer. Having this information, epigenetic therapies can be developed. The pipeline can finally be used to test or follow up these developed epigenetic therapies when treating a patience by comparing the changes in histone modifications during the treatment time (13).

To demonstrate the pipelines functionality, it will be tested on published ChIP-seq data from NIH Epigenomic Roadmap project to analyze the correlation between repetitive sequences and different histone modifications. The pipeline is built to automatically gather these data, which is found on the NCBI’s website. The collected data will contain experiments of four different histone modifications and experiments of their matched inputs. The histone modifications are selected based on how they affect gene expressions. Two of these histone modifications are related to gene activation; trimethylation of lysine 4 of histone H3 (H3K4me3) and acetylation of lysine 27 of histone H3 (H3K27ac). The other two selected histone modifications are related to the repression of gene expression; trimethylation of lysine 9 of histone H3 (H3K9me3) and trimethylation of lysine 27 of histone H3 (H3K27me3).
2 Materials and Methods

The goal with this project is to develop a pipeline that will analyze ChIP-seq in an automated manner.

When the pipeline is built, it needs a lot of computational power and some very specific tools to work, therefore, it is built to be run on a server called UPPMAX. This server provides a large memory space, a parallel processing cluster, and all the third-party tools that will be used by the pipeline are already available. The pipeline will work on any operative system, but a windows user need an additional program to remotely connect to UPPMAX. In this case, MobaXterm (14) was used. A windows user also has to do a simple conversion of encoding from dos (windows) to Unix when running the pipeline. The pipeline is built in the scripting language Bash because of the interest in learning a new language and to make it more universal so that no extra installation is necessary to be installed for the user to run the pipeline.

2.1 Data

The purpose of the pipeline is to analyze short read sequencing data. This data has to be retrieved from somewhere and, in this specific case, from the NIH epigenomic roadmap project which is located on NCBI’s website. Based on the users’ specifications, the pipeline will automatically retrieve this data. In this project, the specified data was ChIP experiments that contain the histone modifications H3K9me3, H3K27ac, H3K4me3 and H3K27me3, as well as their matched inputs which are used as control data. These experiments are sampled from the human genome, in different tissues and organs where diseases occur more commonly (15).

The user needs to prepare three files manually. Because these files have been prepared for this project, they can be reused for any analysis of the same repeat sequences. The first of those files is in a BED (Browser Extensible Data) format. It contains a list of genomic coordinates of all the repeat families to be analyzed (16). The file was created from the RepMasker genome annotation available through the USCS Genome Browser database (17). Repetitive sequences from RepMasker were filtered using the web-based platform Galaxy (18). The two other files are the reference genomes: the standard human reference genome, including the repetitive parts, and the repetitive human reference genome. For this project, the human reference genome (hg19) was retrieved from the Bowtie2’s website (19). The repetitive human reference is needed for extending the analysis to repeat families that are not part of the reference genome (e.g. telomeric and centromeric sequences). This reference is a repetitive human metagenome that was assembled first in form of a FASTA file from the database of repetitive sequences RepBase (20), and subsequently a bowtie2 index was built from the FASTA file.

2.1.1 Experiments

ChIP-Seq and matched input control datasets from the Epigenome roadmap project is listen in a sample sheet shown in Appendix 1.
2.2 Tools
In order to perform the different steps required for the pipeline, some tools need to be used to gather different analyzes. Those tools are already provided by UPPMAX, so the pipeline will simply use them. The short-read sequencing data that is used by the pipeline is of SRA (Sequence Read Archives) format and contains the raw data. In order for this data to be further analyzed, it needs per-base quality scores (21). Therefore, the SRA toolkit is used to convert an SRA file to a FASTQ file. Then, the FASTQ file is analyzed by the FastQC tool to perform a quality control and some basic statistics. This is a good way to ensure that there are no problems with the generated data.

Then, the short-read alignment tool, Bowtie2, is used on the FASTQ files to align them over the reference genomes. The pipeline will align the FASTQ file over both the human reference genome and the repetitive human metagenome. When using the Bowtie2 tool, the option “--fast-local” is used to allow trimming parts of the ends of the sequences. This is to create a higher number of alignments. The ChIP-seq may contain adapters, which are chemically synthesized sequences that do not exist in the actual genome sequence. Therefore, they are causing an increased number of unaligned reads (22, 23).

The aligned FASTQ file will be in the SAM (Sequence Alignment/Map) format, a file that is TAB-delimited. This file contains the reads as well as the alignment information of each read to its reference sequence (21, 24, 25). However, the SAM file is a very large text file, which means that retrieving information from it is very slow. Therefore, the SAM file is converted to a BAM (Binary Alignment/Map) file by SAMtools view, which means that the new file will be much smaller and faster to access. Also, the upcoming usage of the other tools requires a BAM file. SAMtools sort is then used again to sort the BAM file based on the genomic location. SAMtools index is used to create an index file based on the previously sorted BAM. This is done to optimize the retrieval of specific alignments by the other tools, such as SAMtools itself, used in the pipeline (25, 26).

SAMtools flagstat is then used on the sorted BAM file to calculate the total number of reads, the number of reads that have been mapped, the number of reads that are duplicates, etc. This is used to verify that the alignment of the data went well. SAMtools idxstat is also used on the sorted BAM file to count the number of reads per chromosome/repetitive family. This will be used to create a heatmap showing the enrichment of different repetitive families in different ChIP-data.

IGVtools is used by the pipeline on the sorted BAM file to create a WIG file. The WIG file has to be normalized to enable comparison between different ChIP-data, since they may contain different number of reads. The toolkit used for this step is named java-genomic-toolkit. This toolkit is not available in UPPMAX and, therefore, is downloaded automatically by the pipeline before usage. The Java-genomic-toolkit is found in Github (27). Also, Java 7 is required when using java-genomic-toolkit, but it is already provided by UPPMAX. From the java-genomic-toolkit, the scaling function from the toolRunner script (“toolRunner.sh wigmath.Scale”) is used to normalize the WIG file. The resulting WIG file contains normalized read density per 25bp (base pairs) intervals across the entire genome. This information can be used by subsequent programs to extract the read density over specific genes or elements. Following this, the IntervalStats function (“toolRunner.sh ngs.IntervalStats”) is used, together with the BED file containing the repetitive
sequences, to extract normalized read density for each intervals representing a single occurrence of a repetitive sequence (27).

The visualization program *ngs.plot* is then used on the sorted BAM file to create a plot of the average gene profiles and a heatmap. This will show a general enrichment in the gene region from transcriptional start sites (TSS) to the transcription end sites (TES) for that specific ChIP-data. *Ngs.plot* will also be used to show the general enrichment of different repetitive families. To do so, the ngs will be used in parallel on the sorted BAM file together with the BED files for each selected repetitive family (these BED files were missing for the project) to create a multiplot of the average read density profiles and heatmaps over genes (28, 29). Average plots generated by *ngs.plot* over genes and repeat families can be quantitatively compared since they rely on the same normalization.
3 Results

The main result of the project is a pipeline that analyses ChIP-seq data in an automated way to study the correlation between histone modifications and repetitive sequences in the human genome, but also for any other region of the human genome. The pipeline generates quality control information, as well as different types of results in form of graphs and tables. For the project, the pipeline was also tested using the ChIP-seq data containing the histone modifications H3K4me3, H3K27me3, H3K9me3 and H3K27ac.

The pipeline consists of five scripts (Figure 1):

- **Script1** – This script creates a sample sheet containing an informative list of all the selected ChIP-seq data. The ChIP-seq data is selected based on the histone modifications that the user wants to analyze. Each ChIP-seq data has a matched input-seq data that also has to be selected, because it will be used as control data.

- **Script2** – This script then uses the sample sheet created in **Script1** and downloads all the data files from the NCBI's website. The script will also make a control check on each downloaded file to make sure they were successfully downloaded. The ChIP-seq data and input-seq data will be in the SRA format. However, if the ChIP-seq data and the matched input-seq data was provided by the user, this script will not be used in the pipeline.

- **Script3** – This script will pre-process the downloaded files and then map them to the reference genomes. It will then process it with further steps to generate graphs and tables showing the enrichment of repetitive sequences and other genes.

- **Script4** – This script will perform algorithms on the tables generated by **Script3**. This is to calculate how many percent of reads that belong to each repetitive family.

- **Script5** – This script will perform algorithms on different tables generated by **Script3**. This is to calculate the mean value for the number of counts that belong to each repetitive family.
Figure 1: Flow-chart describing the steps of the pipeline.
The script contains arrows showing the order. The flow-chart is read from top-bottom. Squares implement normal action. Circle means that the script is looping. Box implements that the script is run many times in parallel but with different data.
3.1 Script1 – Creates the sample sheet

Script1 will be described in steps. Use Figure 2 for an easier understanding when reading these steps, as the steps here correspond to the same steps in the figure.

1. The script asks the user to type in the histone modifications that the pipeline will analyze.
2. The script then goes to the NCBI’s website and collects, based on the histones provided by the user, all the experiments’ information. The script also collects the information for all the experiments of ChIP-seq input.
3. All the information is then printed into a sample sheet, where each row contains one of the experiment’s information. Each row contains eight TAB-delimited columns. These columns are:
   - GEO accession number (e.g. GSM906418)
   - Data name (e.g. heart_left_ventricle)
   - Data name extra (e.g. Left_Ventricle)
   - Experiment type (e.g. Chip_Seq_input)
   - ResearchPlace (e.g UCSD)
   - ResearchNr (e.g. STL003)
   - ExtraInfo4 (e.g. Reference Epigenome: ChIP-Seq Input from Human Left Ventricle Tissue)
   - InputGEO (e.g. GSM906419)

However, before the script prints in the information to the sample sheet, it first checks for each experiment if they have their SRA file(s) available in the NCBI’s website. If any experiment is missing its data in the website, its information will not be printed into the sample sheet.
4. The script then sorts the sample sheet based on the following columns: Data name, Data name extra, ResearchPlace, ResearchNr, Experiment type.
5. The script then removes from the sample sheet the ChIP-seq input experiments that are not linked to any histone provided by the user; and vice versa.

The finished look for this project’s sample sheet is shown in Appendix 1.
Figure 2: A flow-chart which describes the steps in Script1.
The flow-chart contains arrows showing how to read it. The steps are numbered based on their order. Each number corresponds to the same number in the text where the step is explained, thus should be studied together with the text.
3.2 Script2 – Download data

To optimize the layout of this script and handle series of commands that will be repeatedly used throughout the script, it contains functions. Two functions are used in this script, and these functions are named "DownloadFiles" and "CheckFileExistenceAndSize". We will get more into what they do below. Use Figure 3 for an easier understanding.

1. The script starts by reading the sample sheet row by row. For each row:
2. The script calls a function named “DownloadFiles”. This function starts by creating a unique URL based on the GEO accession number. This URL points to the file belonging to this experiment.
3. Then, the script uses the URL to download the ChIP-seq data files and stores them on the user’s UPPMAX account for further usage.
4. The script then calls a second function named “CheckFileExistenceAndSize”. This function checks if the ChIP-seq data file has been properly downloaded. To do so, it first checks if the file exists in the user’s account. If it is missing, the function will re-execute the “DownloadFiles” function for that specific missing file. However, if the file exists, the function “CheckFileExistenceAndSize” will verify the file’s integrity by comparing the checksum (md5sum) of the local file and the one referred to by the URL. A checksum is a unique ID based on the file size, its content and its timestamp. By comparing the checksum of the local file versus the online file, we can verify that those two files are identical. If the checksum is different, the online file has to be downloaded again. The function then call Script3 and pass the GEO accession number from the current row to it.
Figure 3: Flow-chart describing the steps for Script2.
The flow-chart contains arrows showing how to read it. The steps are numbered based on their order. Each number corresponds to the same number in the text where the step is explained, thus should be studied together with the text. The shape of the boxes has different meanings; squares represent normal functions, circles represent that it is a loop, diamond represent that it is a Boolean and cloud represent that it is executed in parallel.
3.3 Script3 – Map & Process data

Script2 passes the GEO accession number of the experimental data that will be pre-processed and mapped over the reference genomes. The experimental data file is of SRA format and contains the “raw data”. For this file, the many steps will be performed. Following each step, the newly created file will be passed to a function named “FileCheck”. This function checks if the file’s size is greater than 1. If this is false, the script repeats the previous step. Use Figure 4-6 for an easier understanding.

1. The script starts by converting the SRA file to a FASTQ file. This adds the per-base quality scores into the data, which is required when mapping the data using, for example, Bowtie2.
2. The script then does a quality score on the FASTQ file using the FastQC tool. This will generate an HTML file which can be used to verify the quality of the data.
3. The script uses the FASTQ file again to map the ChIP-seq data over the human reference genome (hg19). This will generate a SAM file.
4. The script also uses the FASTQ file to map the ChIP-seq data over the repetitive version of the human reference genome. This will generate a repetitive version of the SAM file.
5. Using the SAMtools view function, the SAM files are then converted to BAM files, which is the binary version of the SAM files. A BAM file has a smaller file size, hence easier to process by other tools. The BAM file format is also required by other tools that will be used in the pipeline.
6. The SAMtools sort function is then used to sort the BAM files based on the genomic location.
7. The sorted BAM files are then indexed using the SAMtools index function. This creates a BAM.bai file as well as a repetitive version of the BAM.bai file. It’s important to sort and index the BAM files in order for tools, such as SAMtools, to efficiently use the BAM files when processing the data (30)
8. For the sorted BAM file and indexed BAM file that come from the data, that was mapped over the human reference genome, the script uses the IGVtools count function to create a WIG file.
9. The script then downloads java-genomic-toolkit since it is not available in UPPMAX. The toolkit is downloaded from github (27). This toolkit is then used to normalize the WIG file. This enables comparison between different ChIP-seq data that else would contain different number of reads.
10. The same toolkit is used again together with the BED file that has been created in Galaxy by the user. This BED file contains a list of repetitive sequences. Java-genomic-toolkit will be used to count the reads in intervals based on the repetitive sequences in the BED file. This will generate an IntervalStats table. This table will be used in Script5.
11. The script will also use the ngs.plot to generate plots of the average gene profile and a heatmap for the sorted BAM file and the indexed BAM file.
12. Ngs.plot will also use BED files that have been created in Galaxy. These BED files are different from the BED file used in step 10, where one BED file contains repetitive sequences from many repetitive families. Each BED file, used in this step, contains repetitive sequences of only one repetitive family. Therefore, there are many BED files of different repetitive families used in this step, by ngs.plot. This creates many heatmaps (one
per repetitive family), and one multiplot of the average gene profile (one plot containing many curves; one curve per repetitive family). However, this step was not performed for the ChIP-seq data that was tested for this project, because of limited time.

13. The SAMtools \textit{flagstat} function is then used on the sorted BAM file and indexed BAM file. This results in a text file containing the following statistics: total number of reads, number of reads that have been mapped, number of reads that are duplicates, etc.

14. The SAMtools \textit{idxstat} function is used on the sorted BAM file and indexed BAM file. This generates a statistical table containing the number of reads per chromosome in the human reference genome.

15. Step 13 is then repeated on the sorted BAM file and the indexed BAM file generated from the \textit{repetitive} human metagenome.

16. Step 14 is also repeated on the sorted BAM file and the indexed BAM file generated from the \textit{repetitive} human metagenome. The generated table is named REP.idxstat and will be used in \texttt{Script4}.
Figure 4: Flow-chart describing the steps for Script3.
The flow-chart contains arrows showing how to read it. The steps are numbered based on their order. Each number corresponds to the same number in the text where the step is explained, thus should be studied together with the text. The shape of the boxes has different meanings; big arrow represent that Script2 passed the experiment ID, squares represent normal function, soft-edged squares represent that a built-in function is called, clouds represent the output file.
Figure 5: This flow-chart is a continuation of the flow-chart in Figure 4 - describes the steps for Script3. The flow-chart contains arrows showing how to read it. The steps are numbered based on their order. Each number corresponds to the same number in the text where the step is explained, thus should be studied together with the text. The shapes of the boxes have different meanings; squares represent normal function, soft-edged squares represent that a built-in function is called, clouds represent the output file.
Figure 6: This flow-chart is a continuation of the flow-chart in Figure 4 - describes the steps for Script3. The flow-chart contains arrows showing how to read it. The steps are numbered based on their order. Each number corresponds to the same number in the text where the step is explained, thus should be studied together with the text. The shapes of the boxes have different meanings; squares represent normal function, soft-edged squares represent that a built-in function is called, clouds represent the output file.
3.4 Script4 – Calculate the histone/input ratio percentage of reads per repetitive family

Script4 will be described in steps. Use Figure 7 for an easier understanding when reading these steps, as the steps here correspond to the same steps in the figure.

1. The script starts by reading the sample sheet row by row. For each row, it checks if the experiment type is **histone modification**. If the experiment type is **input**, it skips that row. Else, it will use the GEO accession number of this experiment and its matched input to open the corresponding **REP.idxstat** files.

2. The script then reads these two files in parallel. Every single row in a file represents a different repetitive family. Each row matches its corresponding row from the other file.

3. The script then reads all the rows of the **histone-REP.idxstat** file and sum together all the values in the column containing the number of reads. The sum is saved in a variable called **histone_sum**. At the same time, the script will execute the same step for the **input-REP.idxstat** file. This sum is saved in a variable called **input_sum**.

4. The script then reads the same files over again, row by row (remember, each row represents a repetitive family). For each row of the **input-REP.idxstat** file, it checks if the column containing the number of reads is 10 or lower. If this is true, a variable called **histone_input_ratio** will be equal to NaN (Not a Number), because this value is too low to be representative. However, if the value of this column is greater than 10, it will divide it with **input_sum**. The result of this operation will be saved in a variable called **input_ratio**. The number of reads for the corresponding row in the **histone-REP.idxstat** file will at the same time be divided by **histone_sum**. The result of this operation is saved as a variable called **histone_ratio**.

5. The final calculation is then taking place. This will divide the **histone_ratio** by the **input_ratio**. The result will be saved in a variable called **histone_input_ratio**. The log2 value of this variable will then be saved in a table called **REP_table**, at the position corresponding to its repetitive family and to its GEO accession number. Finally, when the script is done reading through the sample sheet, the **REP_table** is complete and is printed into a new text file called **result_REP.idxstat** (Table 1).

<table>
<thead>
<tr>
<th>Rep-seq</th>
<th>GEO accession number1</th>
<th>GEO accession number2</th>
<th>GEO accession number3</th>
</tr>
</thead>
<tbody>
<tr>
<td>repetitive sequence1</td>
<td>1</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>repetitive sequence2</td>
<td>NaN</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>repetitive sequence3</td>
<td>3</td>
<td>2</td>
<td>NaN</td>
</tr>
</tbody>
</table>

Table 1: Example table of **result_REP.idxstat** file.

Each row represents the repetitive family and each column represent the histone/input ID. The values represent the percentage of reads that fall into each repetitive sequence. NaN is short for Not a Number.
Figure 7: Flow-chart describing the steps for Script4.

The flow-chart contains arrows showing how to read it. The steps are numbered based on their order. Each number corresponds to the same number in the text where the step is explained, thus should be studied together with the text. The shapes of the boxes have different meanings; squares represent normal function, circles represent that it’s a loop, diamond represent that it is a Boolean, cloud represent the output file.
3.5 Script5 – Calculate histone/input mean of counts per repetitive sequence

Script5 will be described in steps. Use Figure 8 for an easier understanding when reading these steps, as the steps here correspond to the same steps in the figure.

1. The script starts by reading the sample sheet row by row. For each row, it checks if the experiment type is histone modification. If the experiment type is input, it skips that row. Else, it will use the GEO accession number of this experiment and its matched input to open the corresponding IntervalStats files.

2. The script then reads these two files in parallel. Every row in a file represents a repetitive sequence. Many rows can represent the same repetitive sequence. Each row matches its corresponding row from the other file. The script then reads all the rows that belong to the same repetitive sequence.

3. For both files, the script counts the number of rows for that repetitive sequence and saves the result in a variable called histone_nr_rows and input_nr_rows.

4. The script then sums together all the values in the column containing the number of counts. The sum is saved in a variable called histone_sum and input_sum.

5. The script then divides the value of histone_sum by the value of histone_nr_rows and saves the result in a variable called histone_mean. At the same time, the script will execute the same step for the input-IntervalStats file. This sum is saved in a variable called input_mean. The log2 value of histone_mean will then be saved in a table called histone_table, at the position corresponding to its repetitive sequence and to its GEO accession number. Also, the log2 value of input_mean will be saved in a table called input_table.

6. The script then divides histone_mean by input_mean and saves the result in a variable named histone_input_mean. The log2 value of this variable will then be saved in a table named intervalStats_table, at the position corresponding to its repetitive sequence and to its GEO accession number. Finally, when the script is done reading through the sample sheet, the interval_table is complete and is printed into a new text file called result_IntervalStats (Table 2). Also, the histone_table and the input_table will be printed into new text files called histone_result_IntervalStats and input_result_IntervalStats, respectively.

Table 2: Example table of result_IntervalStats file.
Each row represents the repetitive family and each column represent the histone/input ID. The values represent the mean value number of counts that fall into each repetitive family.

<table>
<thead>
<tr>
<th>ID</th>
<th>GEO accession number1</th>
<th>GEO accession number2</th>
<th>GEO accession number3</th>
</tr>
</thead>
<tbody>
<tr>
<td>repetetive family1</td>
<td>-1</td>
<td>-3</td>
<td>3</td>
</tr>
<tr>
<td>repetetive family2</td>
<td>4</td>
<td>-4</td>
<td>2</td>
</tr>
<tr>
<td>repetetive family3</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure 8: Flow-chart describing the steps for Script5.
The flow-chart contains arrows showing how to read it. The steps are numbered based on their order. Each number corresponds to the same number in the text where the step is explained, thus should be studied together with the text. The shapes of the boxes have different meanings; squares represent normal function, circles represent that it is a loop, diamond represent that it is a Boolean, cloud represent the output file.
3.6 FastQC
The FastQC results for the selected experiments were individually checked and show that they have an overall high quality score, as seen in the *Per Base Sequence Quality* bar graph (Figure 9). Most of the experiments’ quality scores were above 28 and stayed in the green zone. However, sample GSM915331 had a bad quality score, where 14/36 bars were below 20 (i.e. in the red zone). This will be relevant when analyzing the results. Also, the sequence’s length for all the experiments is around 36 to 50 base pairs (bp). This is expected since Chip-seq are generating short-length reads (31). Figure 9 shows the FastQC results for the histone modification experiment and its matched input experiment, which originate from the CD4+_CD25-_CD45RA+_naive_primary_cells sample.
Figure 9: FASTQC result.
Two bar-graphs (left) and basic statistic tables (right) are presented. One for the H3K27ac experiment called GSM773004 (bottom) and one for its matched input experiment called GSM772920 (top). In the bar-graph, a good quality score is expected to be above 20. The graph is divided into three different zones, where green indicates a very good quality score, orange indicates a reasonable quality score and red means that the quality score is poor.
3.7 Ngs.plot

The *ngs.plot* graph generated for the ChIP-data, which was mapped over the human reference genome, shows the enrichment of the genomic region between TSS (transcriptional start sites) and TES (transcription end sites), for different histone modification experiments. (28, 29).

Figure 10 presents the result for the four different histone modification experiments. These experiments originate from one common sample that was taken from the Esophagus tissue (Appendix 1). The graph results for the active histone modification experiments, H3K4me3 and H3K27ac, have their maximum value over 0.6 and 1.2, and the graph results for the inactive histone modification experiments have their maximum value around 0.07. Those results mean that the active histone modification experiments have significantly higher peaks than the inactive histone modification experiments (Figure 10). A graph that is showing a high peak around the TSS means that genes are more expressed in that experiment.
Figure 10: Plot generated from ngs.plot.
These graphs present the ngs.plot result for the four histone modification experiments that have been sampled from the Esophagus tissue. The graph’s y-axis shows the number of mapped reads (1 per million scale) for that specific experiment. The x-axis presents the genomic region in the 5’ -> 3’ direction.
3.8 Heatmap

The heatmaps, representing the result_REP.idxstat file and the result_IntervalStats file, show the correlation between the different experiments and the repetitive families. Since the main goal of this project is the pipeline’s development, the results obtained from it will not be discussed. However, for a demonstration purpose, one of the heatmaps will be analyzed.

Figure 11 shows a part of the heatmap generated from the result_REP.idxstat file. On the x-axis, we have all the experiments sorted by their activation/repression histone features and their histone modification types. On this axis, we have the repressive histone modification experiments (H3K9me3 and H3K27me3) and the active histone modification experiments (H3K4me3 and H3K27ac). On the y-axis, all the repetitive sequences are represented alphabetically. The correlation between each experiment and repetitive sequence is represented as a colored tile based on its expression value. Blue colored tiles indicate a low expression value and the red color tiles indicate a high (enriched) value. The value is based on the percentage of reads corresponding to a repetitive sequence. Stronger colors mean stronger depletion/enrichment.

The heatmap shows the correlation between all the experiments and the repetitive sequences belonging to the repetitive family Alu. In this figure, it is possible to see that the colors are distributed in a pattern. The correlation between the repressive H3K9me3 experiments (see Figure 11: on the left side, x-axis highlighted in blue) and all the Alu sequences are mostly represented as blue colored tiles. This means that Alu seem to be low in H3K9me3 experiments. The right side of the heatmap shows the correlation between the active H3K27ac experiments (see Figure 11: on the right side, x-axis highlighted in blue) and all the Alu sequences. These tiles are red colored, meaning that Alu seem to be enriched in H3K27ac experiments. For experiments of the two other histone modifications, H3K27me3 and H3K4me3, the colored tiles are not showing a clear pattern. However, each experiment seems to affect all the Alu sequences the same way. For example, looking at one of the experiment, its whole result column will tend to either be in shades of red or in shades of blue.
Figure 11: Heatmap generated from GENE-E.
A part of a heatmap showing the enrichment of different repetitive sequences from the repetitive family Alu in the different experiments. The experiments are listed on the x-axis (top) and is ordered based on if it contains an active/repressive histone modification and what histone modification it contains. From left side to right, it is the repressive histone modifications H3K9me3 and H3K27me3 and the active histone modifications H3K4me3 and H3K27ac. The correlation between the experiments and the repetitive sequences are presented as a colored box, where red indicates a rich value of the repetitive sequence and blue indicates a poor value. On the y-axis (right side), all the repetitive sequences are represented in an alphabetic order.
4 Discussion

The goal of this project was to develop a ChIP-seq analysis pipeline that automatically analyses Chip-seq data. The data it analyzes can be data from the NIH Epigenomic Roadmap project, which it collects based on the user’s interest, or it can be the user’s own data. The pipeline analyzes both the repetitive regions of the data and the annotated genome to yield a more comprehensive picture of the epigenetic landscape than state-of-the art analysis tools. The pipeline will now be used to study how transposable elements are regulated by epigenetic factors and histone modification, or find the protein binding sites in repetitive DNA sequences for proteins such as TFs.

To demonstrate the pipeline, we used it on published data to see when and how repetitive elements are silenced by the histone modifications. The pipeline successfully analyzed ChIP-seq data from the whole human genome and on the repetitive sequence regions. The pipeline gave results in the form of graphs and tables that are easy to understand and that can be used for additional plots. However, not all expected results were possible to be generated due to lack of time. But with the results obtained, it is possible to validate the functionality of the pipeline, and to see that the data, used by the pipeline, is of good quality. We could also see, from the graph generated from the ngs.plots, that the active histone modifications are more enriched at ‘regular’ genes than at repetitive elements. This is expected because active histone modifications correlate with transcriptional activity, and many genes are transcribed. Repressive histone modifications were very weakly enriched at genes but were enriched over some repeat families.

4.1 Heatmap

The results in the result_REP.idxstat file and the result_IntervalStats file are both showing the enrichment of different repetitive families in examples containing different histone modifications. The advantage with the table of the result_IntervalStats file is that it is normalized against the total genome. This means that “artificial enrichments” arising from analysis of only a subset of genomic sequences can be avoided. To do so, the data that was used to build this table was mapped over the human reference genome. Then, the pipeline created a table for: the mean values of the histone/input ratio (result_IntervalStats file), the mean values for all the histone values (histone_result_IntervalStats file), and the mean values of all the input values (input_result_IntervalStats file). Finally, the heatmap for the input_result_IntervalStats file can be used to check if there has been any "artificial enrichments" (peaks found in the input sample) that are not biologically relevant for any repetitive sequences. These "artificial enrichments" can be seen in the heatmap, since the heatmap’s tiles are colored. A heatmap with no "artificial enrichments" would only have white tiles. In the final table that contains ChIP/input ratios, those "artificial enrichments" are ‘neutralized’ by dividing the histone mean value by its corresponding input mean value.

If result_IntervalStats file is better to use, then why is the pipeline creating the result_REP.idxstat file? As mentioned earlier, the result_IntervalStats file was created from the data that was mapped over the human reference genome, thus contains only repetitive sequences present in the annotated part of the human reference genome (e.g. hg19). However, the annotated human genome misses some repetitive sequences since parts of the chromosomes, such as centromeric and telomeric regions, have not been fully assembled into the reference genome, yet. Therefore, the pipeline also
creates the `result_REP.idxstat` file, which was created from the data that was mapped over the human repetitive sequence metagenome, and yields the enrichment and depletions amongst all repetitive sequences. Therefore, two independent statistical estimations can be used as an extra comparison to the enrichment results.

### 4.2 Future development

To optimize the pipeline, and make it more universal, some additional features could be added to it.

**Script4** and **Script5** could be called by the pipeline at the same time as they are not dependent on each other and, therefore, could run in parallel. **Sbatch** should be used for these two scripts since both require a lot of computational power and calculation time, and could be run in parallel in UPPMAX.

The current pipeline does not keep a log file of the steps that have been done in **Script3**. This should be added for the user to know where the script failed (if that happens) and restart it from there. This would prevent wasting time and computational power on data already processed.

The pipeline does not create any visualizations of the tables from the `result_REP.idxstat` file and the `result_IntervalStats` file. To implement this into the pipeline, a 6th script must be developed in a scripting language that can create plots. The plan would be to create heatmaps of the `result_IntervalStats` table, and bar-graphs showing the “standard error of the mean” (32). This error is calculated by doing a derivation of the standard deviation for each histone/input ratio. The standard deviation calculation is already developed in **Script5**; however, the plots are not.

The pipeline is not fully developed to be used on the user’s own data. The pipeline needs further development to create a sample sheet based on local data. Then, the pipeline should be tested on local data to see if there’s something else that should be implemented in the pipeline to fully analyze this data, as it does on the published data from the NCBI’s website.

The pipeline currently only works on human ChIP-seq data. The future goal is for it to also work on RNA-seq data, and for other organisms, such as the mouse.
5 Acknowledgments

I would like to thank my supervisor Simon Elsässer for allowing me to work in his team and perform this project. I want to thank all my colleagues for their support and ideas. I also want to thank my subject reader Åsa Johansson for her great advice. Finally, I want to thank my family and friends for supporting me during this project.
6 References


7 Appendix

Appendix 1 - The sample sheet used for this project

<table>
<thead>
<tr>
<th>GEO accesion</th>
<th>Data_name</th>
<th>Data_name_extra</th>
<th>Experience_type</th>
<th>ResearchP</th>
<th>ResearchR</th>
<th>Extranet</th>
<th>InputGEO</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSE772000</td>
<td>CD4+ CD8- naive_paren_cells</td>
<td>CD4+CD8- naive_paren_cells</td>
<td>Bly</td>
<td>Donor_62</td>
<td>CHP</td>
<td>Seq</td>
<td>analysis</td>
</tr>
<tr>
<td>GSE772011</td>
<td>CD4+ CD8- naive_paren_cells</td>
<td>CD4+CD8- naive_paren_cells</td>
<td>Bly</td>
<td>Donor_62</td>
<td>CHP</td>
<td>Seq</td>
<td>analysis</td>
</tr>
<tr>
<td>GSE772021</td>
<td>CD4+ CD8- naive_paren_cells</td>
<td>CD4+CD8- naive_paren_cells</td>
<td>Bly</td>
<td>Donor_62</td>
<td>CHP</td>
<td>Seq</td>
<td>analysis</td>
</tr>
<tr>
<td>GSE772031</td>
<td>CD4+ CD8- naive_paren_cells</td>
<td>CD4+CD8- naive_paren_cells</td>
<td>Bly</td>
<td>Donor_62</td>
<td>CHP</td>
<td>Seq</td>
<td>analysis</td>
</tr>
<tr>
<td>GSE772041</td>
<td>CD4+ CD8- naive_paren_cells</td>
<td>CD4+CD8- naive_paren_cells</td>
<td>Bly</td>
<td>Donor_62</td>
<td>CHP</td>
<td>Seq</td>
<td>analysis</td>
</tr>
<tr>
<td>GSE772051</td>
<td>CD4+ CD8- naive_paren_cells</td>
<td>CD4+CD8- naive_paren_cells</td>
<td>Bly</td>
<td>Donor_62</td>
<td>CHP</td>
<td>Seq</td>
<td>analysis</td>
</tr>
<tr>
<td>GSE772061</td>
<td>CD4+ CD8- naive_paren_cells</td>
<td>CD4+CD8- naive_paren_cells</td>
<td>Bly</td>
<td>Donor_62</td>
<td>CHP</td>
<td>Seq</td>
<td>analysis</td>
</tr>
<tr>
<td>GSE772071</td>
<td>CD4+ CD8- naive_paren_cells</td>
<td>CD4+CD8- naive_paren_cells</td>
<td>Bly</td>
<td>Donor_62</td>
<td>CHP</td>
<td>Seq</td>
<td>analysis</td>
</tr>
<tr>
<td>GSE772081</td>
<td>CD4+ CD8- naive_paren_cells</td>
<td>CD4+CD8- naive_paren_cells</td>
<td>Bly</td>
<td>Donor_62</td>
<td>CHP</td>
<td>Seq</td>
<td>analysis</td>
</tr>
<tr>
<td>GSE772091</td>
<td>CD4+ CD8- naive_paren_cells</td>
<td>CD4+CD8- naive_paren_cells</td>
<td>Bly</td>
<td>Donor_62</td>
<td>CHP</td>
<td>Seq</td>
<td>analysis</td>
</tr>
<tr>
<td>GSE772101</td>
<td>CD4+ CD8- naive_paren_cells</td>
<td>CD4+CD8- naive_paren_cells</td>
<td>Bly</td>
<td>Donor_62</td>
<td>CHP</td>
<td>Seq</td>
<td>analysis</td>
</tr>
<tr>
<td>GSE772111</td>
<td>CD4+ CD8- naive_paren_cells</td>
<td>CD4+CD8- naive_paren_cells</td>
<td>Bly</td>
<td>Donor_62</td>
<td>CHP</td>
<td>Seq</td>
<td>analysis</td>
</tr>
<tr>
<td>GSE772121</td>
<td>CD4+ CD8- naive_paren_cells</td>
<td>CD4+CD8- naive_paren_cells</td>
<td>Bly</td>
<td>Donor_62</td>
<td>CHP</td>
<td>Seq</td>
<td>analysis</td>
</tr>
</tbody>
</table>

... (continued with more rows)