# HeatMapper Expansion

Thesis BachelorProject IN 3405

Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Bioinformatics

Sietse Au 1382756
Freek Kuethe 1522434
Firdina Lesmana 1037668

# Foreword

This bachelor thesis is written in completion of our Bachelor degree in Computer Science at the Delft University of Technology. This is a collaborative project of the Department of Hematology of Erasmus MC Rotterdam and the department of Bioinformatics of the Faculty of Electrical Engineering, Mathematics and Computer Science at the Delft University of Technology.

The aim of this bachelor project is to aid and provide researchers at Erasmus MC as well as other researchers in similar fields in getting more accurate and rapid interpretation of large scale genomic data.

We would like thank all of the involved supervisors during this project, both from Delft University of Technology as well as from Erasmus MC, especially Marcel Reinders of the Delft University of Technology and Erdogan Taskesen of the Erasmus MC. Thank you for the useful input, suggestions, guidance and willingness to supervise our project.

Delft, February 2012

Sietse Au, Freek Kuethe and Firdina Lesmana

# Table of Contents

# Glossary

| | |
|---|---|
| **neutriphil** | Most common type of white blood cells |
| **dendritic cell** | An immune cell |
| **gene** | a molecular unit of heredity of a living organism, is made up of DNA |
| **DNA** | Deoxyribonucleic Acid, is the inheritable characteristics of an organism |
| **nucleotide base** | basic building block of DNA |
| **DNA sequencing** | determining the order of the nucleotide bases in a DNA molecule |
| **DNA microarray** | a collection of DNA spots attached to a chip |
| **DNA spot** | a spot on a microarray in which a certain DNA sequence is defined |
| **probe set** | a set of probes of a DNA microarray |
| **DNA probe** | a labeled segment of DNA used to find a specific sequence of nucleotides in a DNA molecule |
| **pathway** | a set of genes which have a relation with each other |
| **methylation** | the addition of a methyl group (CH3) to an existing molecule. Methylation in promoter regions of a gene can lead to silencing of that specific gene. |
| **DMP** | DNA Methylation Profile |
| **gene expression** | the process which uses information in a gene to synthesize a gene product |
| **GEP** | Gene Expression Profile |
| **CNV** | Copy Number Variations, are alterations in DNA sequences where gains and/or losses of DNA sequences occur |
| **SNPs / SNIPs** | Single Nucleotide Polymorphisms, are DNA sequence variations that occur when a single nucleotide (A,T,C or G) in the genome sequence is altered |
| **HeatMapper** | an application developed by the Erasmus MC for displaying heat maps |
| **correlation matrix** | matrix filled with correlation values |
| **correlation value** | Pearson's correlation coefficient |
| **Pearson's correlation coefficient** | a value between -1 and 1, representing negative or positive correlation |
| **genomic range** | a span of genomic positions |
| **genomic position** | genomic position on the human genome |
| **human genome** | entirety of a human's hereditary information |
| **affymetrix** | a company that manufactures DNA microarrays |
| **chromosomal location** | location on a chromosome |
| **thread-safe** | code is thread-safe when its data remains consistent even if multiple threads manipulate the data |
| **DNA sequence alignment** | an arranged representation of multiple DNA or RNA sequences, where areas of common properties are aligned. In the case of this project, probe data is aligned based on the location of the probe in the human genome |
| **t-test** | a statistical hypothesis test that is used to evaluate whether two sets of measures are essentially different |
| **hypergeometric distribution** | a discrete probabilty distribution that arises when a random draw is made among two distinct types without replacement |
| **GO** | Gene Ontology |

|  |  |
|---|---|
| **KEGG** | Kyoto Encyclopedia of Genes and Genomes |
| **genome browser** | an application to visualize data concerning genomes and their expression. It is a graphical interface to display information read from a genomic data |

# Part 1. Introduction

Acute Myeloid Leukemia (AML), is a form of cancer, where immature blood cells do not properly develop into different types of blood cells such as neutrophils, dendritic cells and macrophages. It is characterized by the rapid growth of abnormal white blood cells in the bone marrow. It is caused by mutations and chromosomal aberrations that obstruct the development of stem cells to properly functioning blood cells. After years of research, several subtypes of AML have been revealed.

Technological advances in DNA sequencing has allowed computerized processing of DNA data. Raw data is hard to interpret for human researchers, therefor there is a lot of interest in having computers display bio-molecular data in visually attractive and informative way for humans.

This thesis will discuss the expansion of an existing tool for visualizing biological data: HeatMapper. Three extra approaches to visualize data will be built into application. Each approach is elaborated in respectively the **Correlation, Pathways**, and **Alignment** chapter. The three approaches are:

- **Correlation**
  There exist applications to create correlation matrices, but it's mainly manual work done in general mathematical analysis packages. Visualization of correlation matrices is mostly done by heat mapping, but the disadvantage of heat mapping is that the context is lost, it's not clear for the user what each matrix element represents in human genome. The Correlation approach attempts to solve the contextual problem by allowing the user choose multiple groups of samples and choose genes of choice within one data set to generate specific visualizations of which the context can be altered by the user.

- **Pathways**
  A genetic pathway is a set of interactions between multiple groups of genes, who depend on each other to provide a function to the cell. The Pathways approach aims to displays which pathways are significantly related to specific types of AML. Users are allowed to choose two groups within AML, which they can compare and visualize. Just like the correlation approach, the data is presented in a circular diagram.

- **Alignment**
  Data is visualized by aligning multiple data sources. Unlike the other two approaches, the data is viewed not in a circle, but in horizontal tracks, each track contains data points of a certain data source, such as a specific patient. Each data point corresponds to a specific location on the genome and displays its value on a corresponding location on the track. By aligning different tracks below each other, one could easily compare different data sources, while looking at specific places of the human genome.

The elaboration of each approach will contain an analysis of the visualization, design of the functionality, design of the user-interface and implementation details and decisions. Finally the thesis will end with an evaluation of each way of visualization followed by the conclusion and recommendations chapter.

# Part 2. Correlation:

Different types of information can be measured from genes. For example the intensity of methylation near a certain gene and the intensity of the gene-expression of a gene. Research shows that methylation may play a role in the regulation of gene-expression. [Grewal SI, Rice JC] Calculating the correlation between methylation and gene-expression data may help to gain a clearer insight into the relationship of methylation in the regulation of gene-expression. First the idea of correlation will be defined and in what way it is applied to the data sets. Then the problem analysis, requirements, design and implementation follow.

## 2.1 Pearson's correlation coefficient

The Pearson correlation coefficient is an indicator for a positive and negative linear correlation. The correlation coefficient r is calculated by taking two variables, say X,Y.

$$r=\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{(n-1)s_x s_y} \text{ where } X=x_i \ldots x_n,\ Y=y_i \ldots y_n,$$

where $\bar{x}, \bar{y}$ are the averages of $X$ and $Y$ respectively.

and where $s_x, s_y$ are the standard deviations of $X$ and $Y$ respectively

Observe the following:

- if $x_i$ and $y_i$ are predominantly above average OR predominantly below average then $(x_i-\bar{x})(y_i-\bar{y})$ would be positive
- if $x_i$ is predominantly below average and $y_i$ is predominantly above average vice versa then $(x_i-\bar{x})(y_i-\bar{y})$ would be negative

What can be seen here is that a positive correlation indicates a general tendency that large values of X are associated with large values of Y and that small values of X are associated with small values of Y. A negative correlation would indicate that large values of X are associated with small values of Y and large values of Y are associated with small values of X.

## 2.2 Computation of correlation coefficients

The approach provides three ways to pick the genes to be correlated:

1. Select 1...24 chromosomes and correlate the GEP data with the DMP data of the genes in the chromosomes.
2. Build two lists of gene-symbols, one list of gene-symbols in the GEP data and one list of gene-symbols in the DMP data. These genes are then correlated with one another. It is possible that gene-symbols in the GEP data do not exist in the DMP data vice versa. Therefor the computation of the output is split up in calculating the correlation of three sets A, B, C visualized by the colors red, green and blue respectively

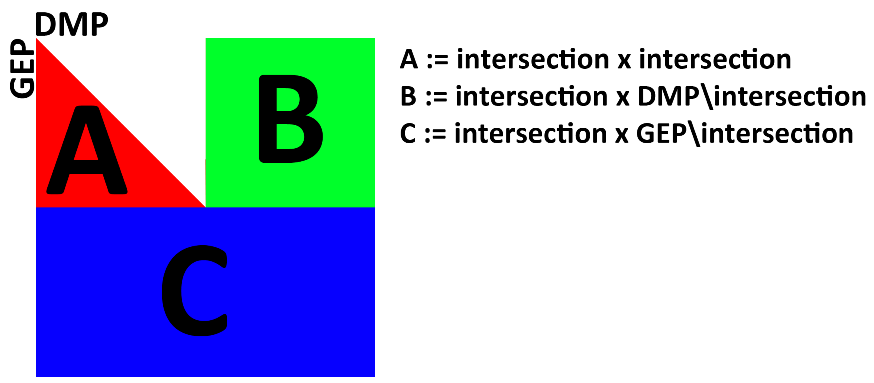

DMP
GEP
A
B
C
A := intersection x intersection
B := intersection x DMP\intersection
C := intersection x GEP\intersection

*Figure 1: Schematic overview of a matrix in which data has to be calculated.*

DMP represents the columns of gene unique identifiers
GEP represents the rows of gene unique identifiers

DMP and GEP are split up in three sets:

A. **Intersection of GEP and GEP**; whereas only half of the matrix needs to be calculated as the values are mirrored values of the values in the red area in A.

B. **Intersection x DMP\intersection**; this is a full heat map as each value in the matrix is unique.

C. **Intersection x GEP\intersection**; this is also a full heat map as each value in the matrix is unique.

3. Create a heat map of all like the red area marked with the letter A in Figure 2. In a heat map both axes, in this case, the GEP axis and the DMP axis have corresponding elements (genes). Only half of the matrix has to be computed as the 2nd half (white) is an exact mirror image of the 1st half (red).

## *2.3 Problem analysis*

The correlation approach allows the user to select a correlation-threshold for a large data set and visualize this in a circular diagram. It allows users to generate a circular diagrams given a threshold for the underlying correlation values.

Correlation between two variables, in this case a DMP variable and a GEP variable, can be computed by using the Pearson's correlation coefficient. Correlation is usually visualized in correlation matrices. however by displaying the correlation in a matrix, the context of each correlation value is fades away, the user will not be able to grasp the entire picture by just looking at the calculated correlation values.

A way to solve this is by visualizing each matrix entry as a relation between a DMP variable and a GEP variable as a link in a circular diagram see Figure 2. By visualizing the matrix entry in a circular diagram, it instantly becomes clear to the user how strongly genomic ranges relate to one another given a threshold.

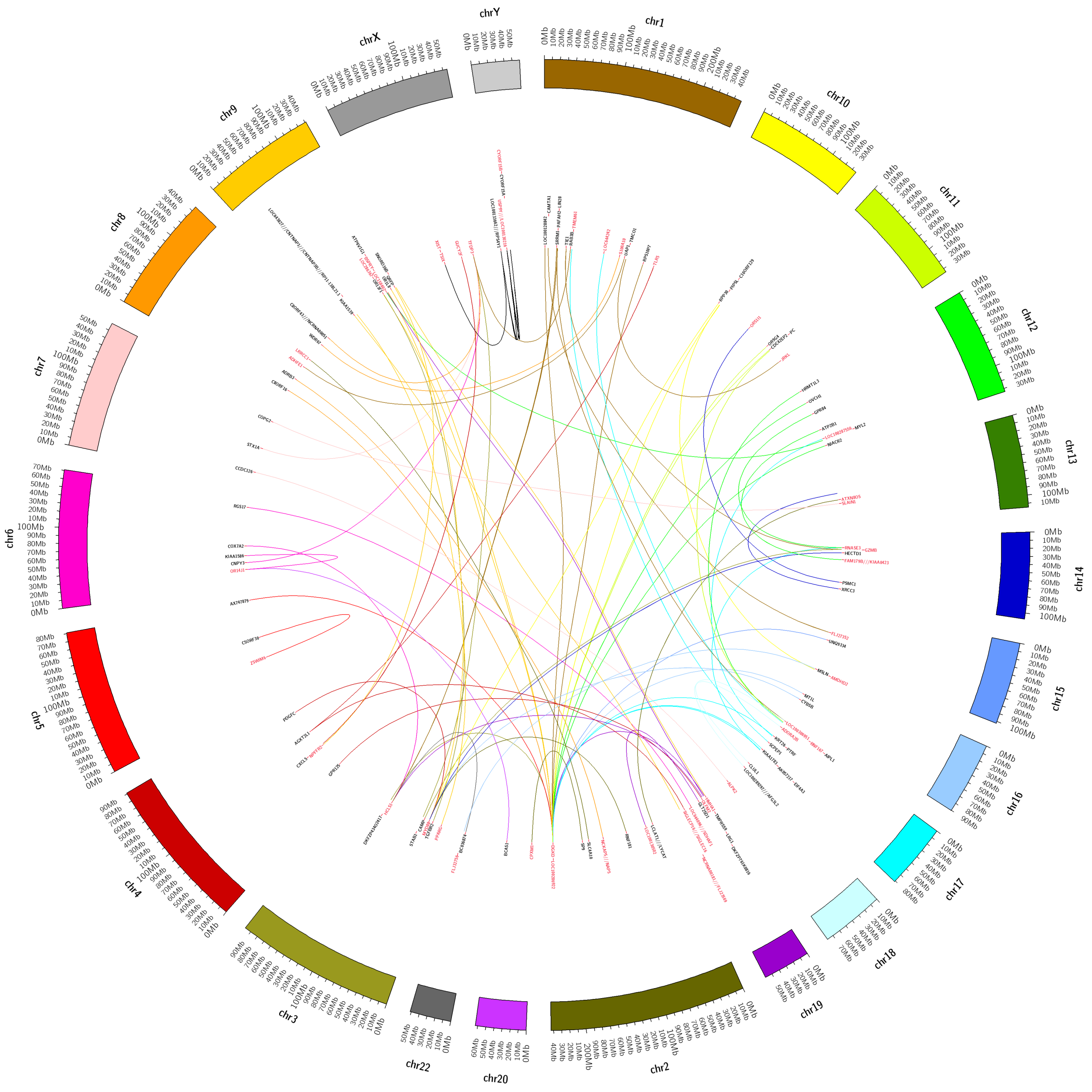


*Figure 2: high-threshold (0.8) of the entire genome filtered on a clinical data set. The red gene-symbols are from the GEP set and the black gene-symbols are from the DMP set.*

## *2.4 Requirements*

### 2.4.1 Reading Affymetrix annotation files

Affymetrix annotation files are comma separated values (CSV) files generated by a company named Affymetrix. The annotation files contain information about each probe from a DNA micro array. Reading annotation files is essential in order to information about probe sets such as gene-symbol name, chromosomal location can be extracted, which can be used for filtering, selecting and grouping genes.

### 2.4.2 Reading human sample files containing GEP or DMP data values

Annotation files give information about each probe, but in order to calculate correlation coefficients sample values are needed. Each probe has n GEP samples and m DMP samples, each sample has a unique identifier. The identifier is essential, because the intersection of the sample identifiers of both the GEP and the DMP probe set needs to be used; as each sample represents a person, it's not useful to see if person X's GEP value has any linear relation with another person Y's DMP value.

#### 2.4.3 Letting the user set a correlation threshold

The user needs to set an absolute correlation threshold (both negatively and positively correlated variables will then be displayed) that will correlate the genes between GEP and DMP. The more stringent value will result in less correlation between GEP and DMP.

#### 2.4.4 Letting the user select genes of interest

With the same analogy as setting a correlation threshold, it is up to the use to decide which genes are interesting.

#### 2.4.5 Letting the user select samples of interest

Samples represent patients and each patient has different properties. The user would like to select samples with certain properties.

#### 2.4.6 Displaying data in a circular diagram

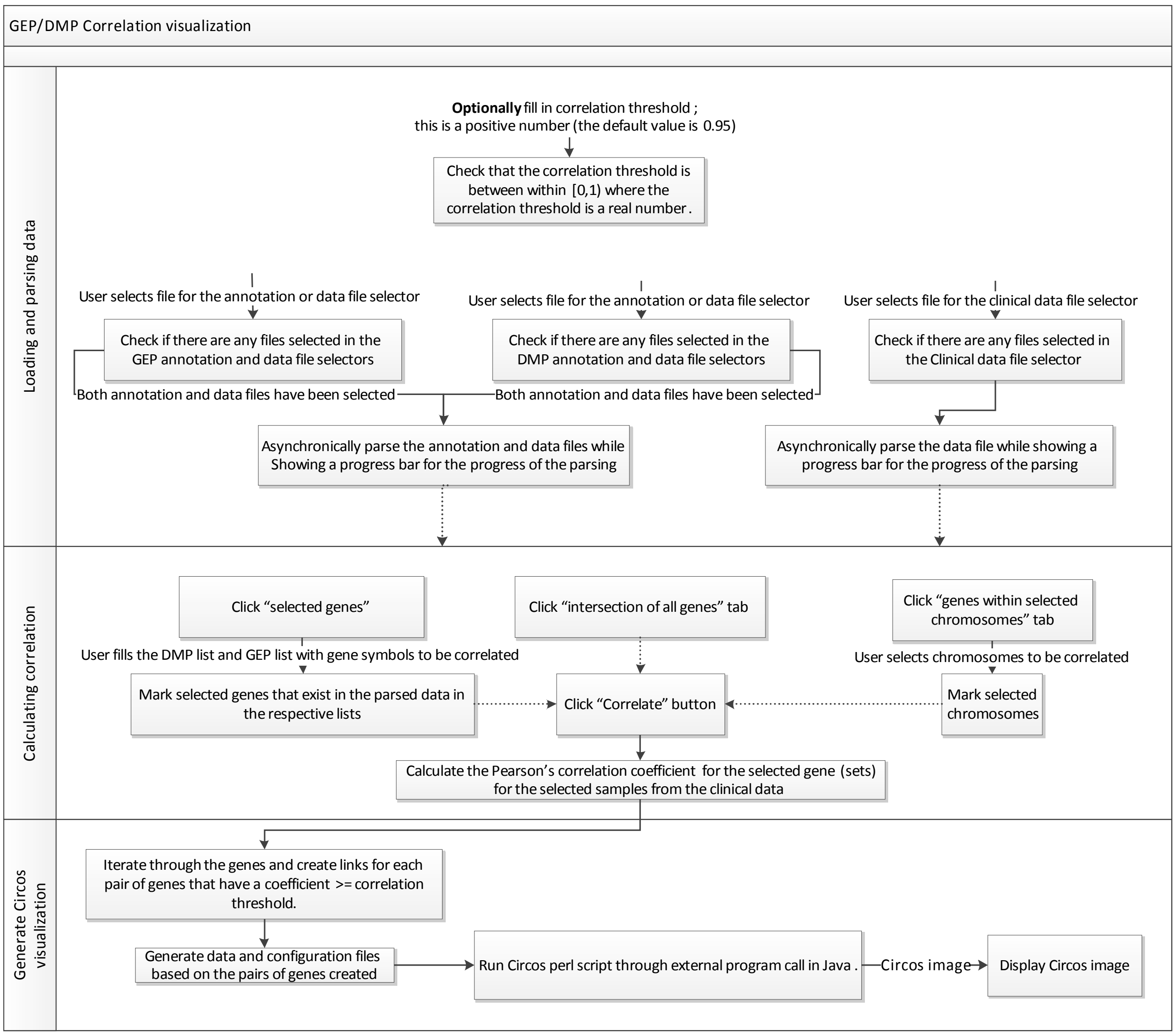
A circular representation of data gives more information about relations between genes than tabular data. An external perl script called Circos [Circos] will be used as it is an established tool for generating genomic visualizations.
In the next section the general design will be elaborated with the help of a flow chart.

### *2.5 Design*

The process of getting a visualization is split up in three distinct parts (see Figure 3):

- Loading and parsing of input data
- Selection and calculating correlations
- Displaying the resulting visualization

This is also the order of the flow of the application.


GEP/DMP Correlation visualization
Loading and parsing data
Optionally fill in correlation threshold ; this is a positive number (the default value is 0.95)
Check that the correlation threshold is between within [0,1) where the correlation threshold is a real number.
User selects file for the annotation or data file selector
Check if there are any files selected in the GEP annotation and data file selectors
User selects file for the annotation or data file selector
Check if there are any files selected in the DMP annotation and data file selectors
User selects file for the clinical data file selector
Check if there are any files selected in the Clinical data file selector
Both annotation and data files have been selected
Both annotation and data files have been selected
Asynchronically parse the annotation and data files while Showing a progress bar for the progress of the parsing
Asynchronically parse the data file while showing a progress bar for the progress of the parsing
Calculating correlation
Click "selected genes"
Click "intersection of all genes" tab
Click "genes within selected chromosomes" tab
User fills the DMP list and GEP list with gene symbols to be correlated
User selects chromosomes to be correlated
Mark selected genes that exist in the parsed data in the respective lists
Click "Correlate" button
Mark selected chromosomes
Calculate the Pearson's correlation coefficient for the selected gene (sets) for the selected samples from the clinical data
Generate Circos visualization
Iterate through the genes and create links for each pair of genes that have a coefficient >= correlation threshold.
Generate data and configuration files based on the pairs of genes created
Run Circos perl script through external program call in Java .
Circos image
Display Circos image
*Figure 3: Flow chart, "Calculating correlation" part represents the three ways to pick genes in 2.2*

## 2.5.1 Loading and parsing of data

To be able to do calculations, there has to be a way to take files as an input. A file location needs to be identified. Once the file location has been identified, the file has to be parsed. Once the file has been parsed, the following part will allow the user to select genes of interest.

## 2.5.2 Selection of genes and calculating correlations

Selection of genes is important, for convenience two extra ways of selecting genes have been added: "intersection of all genes" and "genes within selected chromosomes". The first will only calculate correlations coefficients of genes which are contained in both the DMP and GEP files. The latter will allow users to correlate genes of entire chromosomes with one another.

## 2.5.3 Displaying the resulting visualization

Circos requires a configuration file and data files in a certain format. These files will need to be generated accordingly to the selection of genes. Once this is done, the perl script can be activated given the configuration and data files. The perl script will then generate a circular diagram.

## *2.6 Implementation*

This part will elaborate the implementation decisions made.

### 2.6.1 Model-view-controller

The software pattern used will be the model-view-controller. This is a way of separating, data object logic, program logic and user-interface logic. A package will be created for model, view and controller. In the case of the functionality to be implemented, the model will contain data objects which hold data parsed from input files, the view will contain the user-interface implementation, and finally the controller will contain the code for manipulating the data objects.

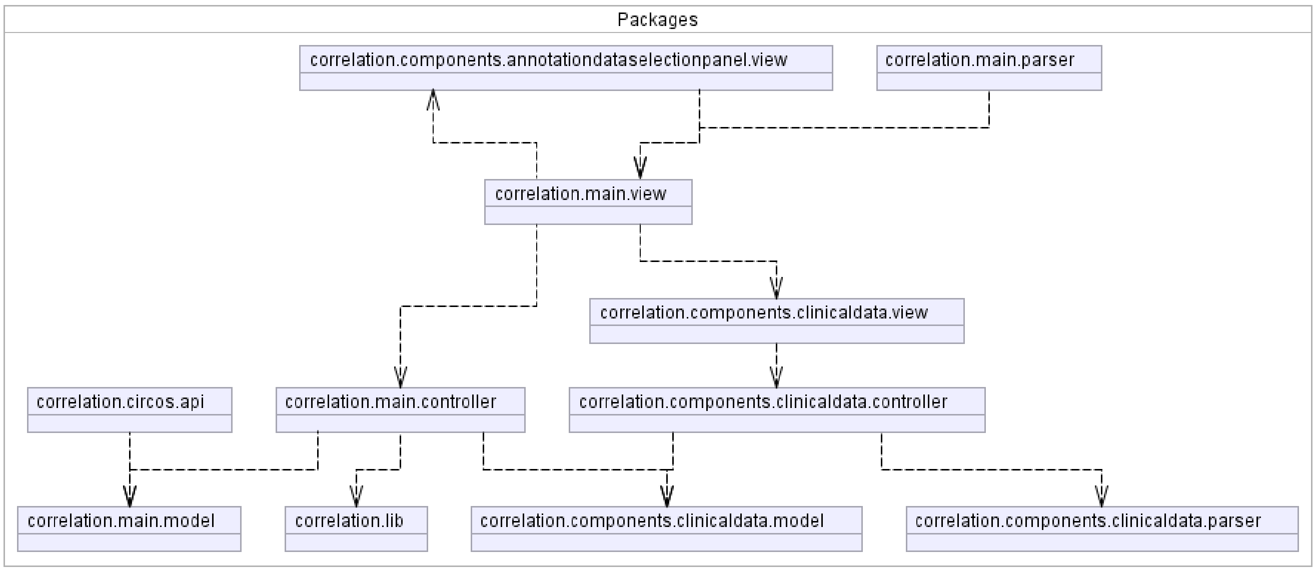
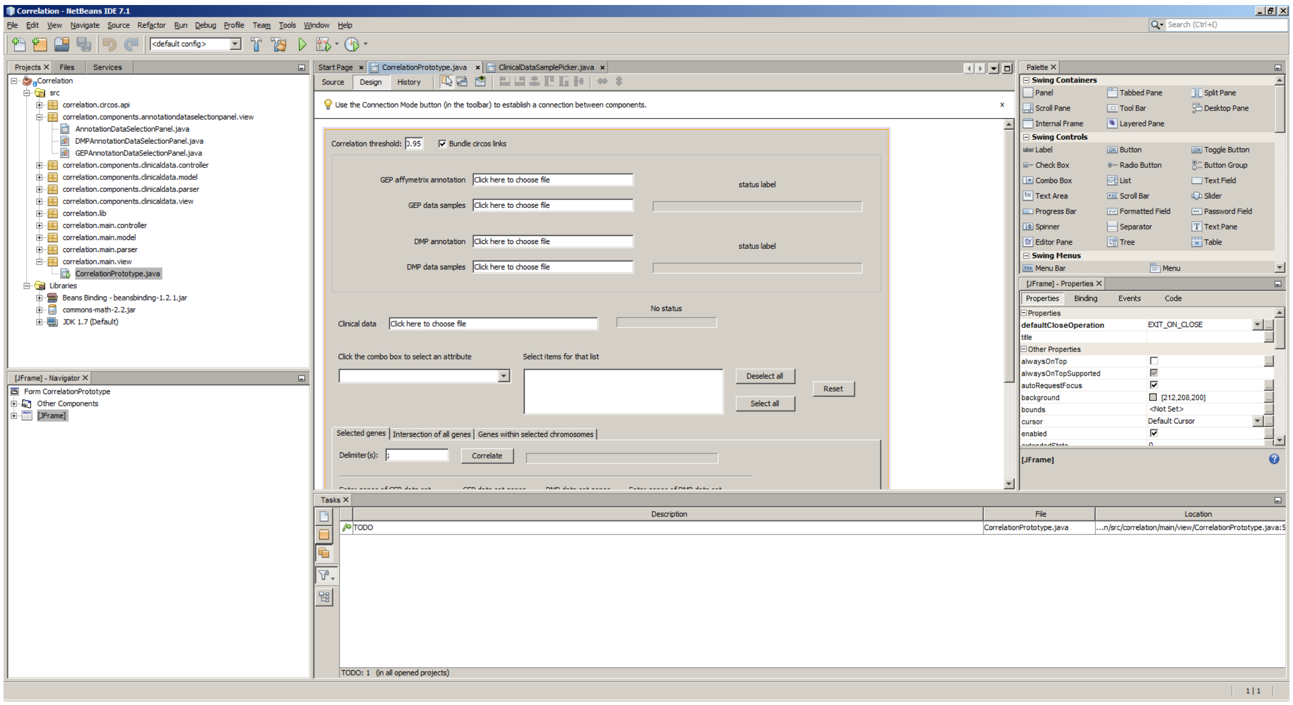

Packages
correlation.components.annotationdataselectionpanel.view
correlation.main.parser
correlation.main.view
correlation.components.clinicaldata.view
correlation.circos.api
correlation.main.controller
correlation.components.clinicaldata.controller
correlation.main.model
correlation.lib
correlation.components.clinicaldata.model
correlation.components.clinicaldata.parser

*Figure 4: Overview of the code structure*

In Figure 4 the model-view-controller pattern is visible;the views do not have direct control over the data, they delegate high level instructions to the controllers which then manipulate the data in the models.

### 2.6.2 Front end

The front end is built with the built-in drag and drop user-interface editor of the NetBeans IDE. By using the NetBeans IDE more time can be spent on the implementation of back end of the functionality.



Use the Connection Mode button (in the toolbar) to establish a connection between components.

<default config>

Correlation threshold: 0.95 ☑ Bundle circos links

GEP affymetrix annotation: Click here to choose file
GEP data samples: Click here to choose file
DMP annotation: Click here to choose file
DMP data samples: Click here to choose file
Clinical data: Click here to choose file
Delimiter(s): ;

| | |
|---|---|
| Properties | |
| defaultCloseOperation | EXIT_ON_CLOSE |
| title | |
| Other Properties | |
| alwaysOnTop | ☐ |
| alwaysOnTopSupported | ✓ |
| autoRequestFocus | ✓ |
| background | [212,208,200] |
| bounds | <Not Set> |
| cursor | Default Cursor |
| enabled | ✓ |

| Description | File | Location |
|---|---|---|
| TODO | CorrelationPrototype.java | ...n/src/correlation/main/view/CorrelationPrototype.java:5 |

TODO: 1 (in all opened projects)

*Figure 5: NetBeans IDE for creating user interfaces*

In Figure 5 the NetBeans IDE is shown. On the right hand side are components which can be dragged into the interface in the middle. It is also possible to create custom components, these should be dragged into the interface in the middle from the source tree on the left.

## 2.6.3 Back end

The implementation of the back end consists of three main parts:

1. Parsing input data
2. Manipulating the input data
3. Creating a visualization of the input data based on the input.

### 2.6.3.1 Parsing the input data

In total there are five different input files to be parsed all in CSV format:

- GEP Affymetrix annotation file
  - information of each probe set
- GEP data file
  - measurements for multiple samples for each probe set
- DMP annotation file
  - information of multiple probes in each probe set.
- DMP data file
  - measurements for multiple samples for each probe set
- Clinical data file
  - information about each sample

The attributes that needs to be extracted from the first four files are:

1. unique identifier of each probe set
2. chromosome in which the probe set is located
3. the starting position in the chromosome
4. the stopping position in the chromosome
5. the gene uid which is measured
6. mapping of sample name to value.

So the data structure containing these 6 attributes will be called a row. Here follows a definition of a row:
$row := \langle uid, chr, start, stop, g, s \rangle$ ,where uid, chr, start, stop, g and s correspond to 1, 2, 3, 4, 5, 6 respectively.

The extraction of the data for the DMP files are quite straightforward; the values in the columns of the DMP annotation file correspond to attributes 1 – 5 and the values in the columns of the DMP data file correspond to attribute 6.

The GEP Affymetrix annotation file is less straightforward; attributes 2 – 4 need to be extracted from the column called “Alignments”. The value in this column is formatted as follows:

[chromosome]:[start position]-[stop position] ([strand]) // [alignment matching percentage] // [chromosomal location]

Example: chr6:30856165-30867931 (+) // 95.63 // p21.33

This would be an probe set in chromosome 6 with start position 30856165, stop position 30867931, a positive strand, a matching percentage of 95.63 and at chromosomal location p21.33. It is also possible that there are multiple alignments; these are separated by a triple forward slash “///”. In that case the data of the alignment with the highest matching percentage will be used.

Example: chr6:30856165-30867931 (+) // 95.63 // p21.33 /// chr4:30856169-30867967 (+) // 80.32 // p21.33

In this case the first data of the first alignment will be used: chromosome 6 with start position 30856165 and stop position 30867931.

As a sanity check the column “chromosomal location” will be checked against the data with the highest matching percentage, if the chromosomes do not match the entry will not be used. Attribute 1 has its own respective column in the GEP Affymetrix annotation file. As for attribute 5, the value is a string containing several gene-symbols separated by a triple forward slash “///”.

In order to aggregate aliased gene-symbols, gene symbols are given an internal unique identifier, whereas multiple aliases of the same gene-symbol can be identified with one unique gene identifier. The GEP data file is similar to the DMP data file where the columns represent the value of a specific sample in a specific probe set.

A clinical data file is a file where the rows are samples and values and the columns the attribute name of said values. The extraction is therefore trivial: each row is represented by a data structure which has a attribute to value mapping.

### 2.6.3.2 Manipulating the input data

The data has been extracted to compute correlations. The input for the correlations is a set of GEP gene symbols, a set of DMP gene symbols and a set of sample identifiers. Using the GEP and DMP gene-symbols sets, the rows containing the gene-symbols in those sets are gathered. Internally there is a conversion from gene-symbol to its respective gene unique identifier. Correlation values are then calculated given the input and saved in an Object.

Example:

Input for calculation

$genes_{GEP} = \{gene1\}$

$genes_{DMP} = \{gene3\}$

$samples = \{1234, 2345, 4566\}$

Conversion gene-symbol to gene unique identifier:

$geneUids_{GEP} = \{1 \Rightarrow [gene1]\}$

$geneUids_{DMP} = \{2 \Rightarrow [gene2]\}$

Rows containing gene1 for GEP and gene3 for DMP: (non-relevant attributes are excluded for convenience)

$row_{GEP} = \{\text{gene-uid} \Rightarrow 1, [1234 \Rightarrow 1, 2345 \Rightarrow 2, 4566 \Rightarrow 3]\}$

$row_{DMP} = \{\text{gene-uid} \Rightarrow 2, [1234 \Rightarrow 4, 2345 \Rightarrow 7, 4566 \Rightarrow 6]\}$

Using the formula in 2.1 Pearson's correlation coefficient, if we take X as $[1234 \Rightarrow 1, 2345 \Rightarrow 2, 4566 \Rightarrow 3]$ and Y as $[1234 \Rightarrow 4, 2345 \Rightarrow 7, 4566 \Rightarrow 6]$

then the correlation value = 0.6546; a positive correlation.

The correlation matrix would look like this:

| | 1 |
|---|---|
| 2 | 0.65 |

## 2.6.3.3 Creating a visualization of the input data based on the input

See Part 5 Circos for the implementation of Circos.

# Part 3. Pathways

The expansion of the HeatMapper with visualization functionality pathways to visualize genes and possible pathways linked to those genes will be further elaborated in this chapter. First, problem analysis and its solutions will be explained. Thereafter, a UML diagram, a function diagram and several screenshots of the user interface will be displayed. Finally, details about the methodology and the implementation will be furthermore elaborated.

## *3.1 Problem Analysis*

It is essential for researchers to have a better visualization in pathway analysis, because of the increased complexity of data-sources. Therefore is this expansion to the HeatMapper MADEx. This expansion will be able to display pathways and their significantly related genes in a circular diagram. This circular diagram will also be visualized, in the sameway as Correlation part from chapter 1 using Circos.

Cell behavior is regulated by a complex network of intracellular and extracellular signaling pathways. Researches and experiments have discovered the identification and characterization of the components that make up such signaling pathways. Researchers have discovered many important biological pathways through laboratory studies and experiments of cultured cells, bacteria, mice and other organisms. Many of these identified pathways have similar working in humans. How these pathways are controlled, how they communicate with each other and most importantly, what happens when they do not function properly, this is what pathway analysis implies. [MolDevices]

The main target of this project part is to get other form (circular presentation) of visualization. This gives researchers another view of visualization which genes belong to a specific pathway and which pathways are significantly related to specific AML type(s). It might be able to show a possible connection between a pathway and a specific AML type that is related to a mutated specific gene.

From several studies, It is known that there have been several mutated genes discovered that might be the cause of AML, such as CEBPα, K-RAS, etc. Users, in this case researchers will be able to compare genes from patients or samples that belong to one of these groups with another group or with samples from a control group. Comparing one AML type with a control group is mostly desired.

## *3.2 Interactions and Design*

There will be some interactions between the users and this pathway part, namely users can access this new pathway functionality by pressing the pathway button on the HeatMapper, he will have to choose a .csv file containing clinical data to use/open that will be imported and read. The file containing the annotations will automatically imported once the next-button is pressed. This .csv file has to have comma as separator and the numbers have to use point to denote decimal. Users will have to choose totally two files as input for this program part.

Another form of interaction is that it is also possible for the users to choose which genes of AML types they want to compare and visualize. They will have to pick two genes from a list of several genes read from the annotation file and they will have to choose two of them or one from this list and control group to compare. For example, they can choose to compare CEBPα with the control group or CEBPα with K-RAS. After this interaction process they can also fill in an alpha to threshold p-values found from the t-tests before. Users also have the possibility to choose if he wants to apply multiple-test or not. Under this alpha field, there is a checkbox which he can check or uncheck.

Thereafter, user has to choose one of five files: c1,...,c5; that contains gene set collections. These files are already downloaded from the Gene Set Enrichment Analysis site(GSEA). [GSEA]. Chosen file will then be imported and added to the database automatically.

These five GSEA files are:

- c1: positional gene sets.
- c2: curated gene sets.
- c3: motif gene sets.
- c4: computational gene sets.
- c5: gene ontology (GO) gene sets.

Figure 11 shows a diagram of some actions the users need to perform in order to get the desired end visualization.

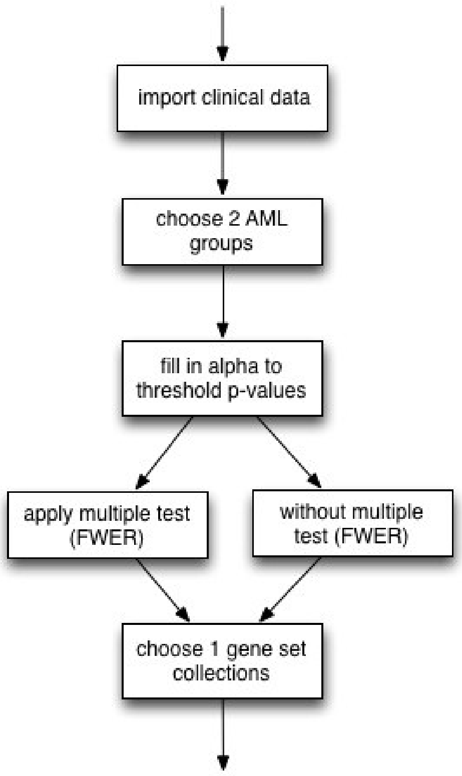

import clinical data
choose 2 AML groups
fill in alpha to threshold p-values
apply multiple test (FWER)
without multiple test (FWER)
choose 1 gene set collections

*Figure 6: User interaction diagram.*

Finally in Figure 6 below, an image that illustrates the expected end image that approximately will be displayed using circos.

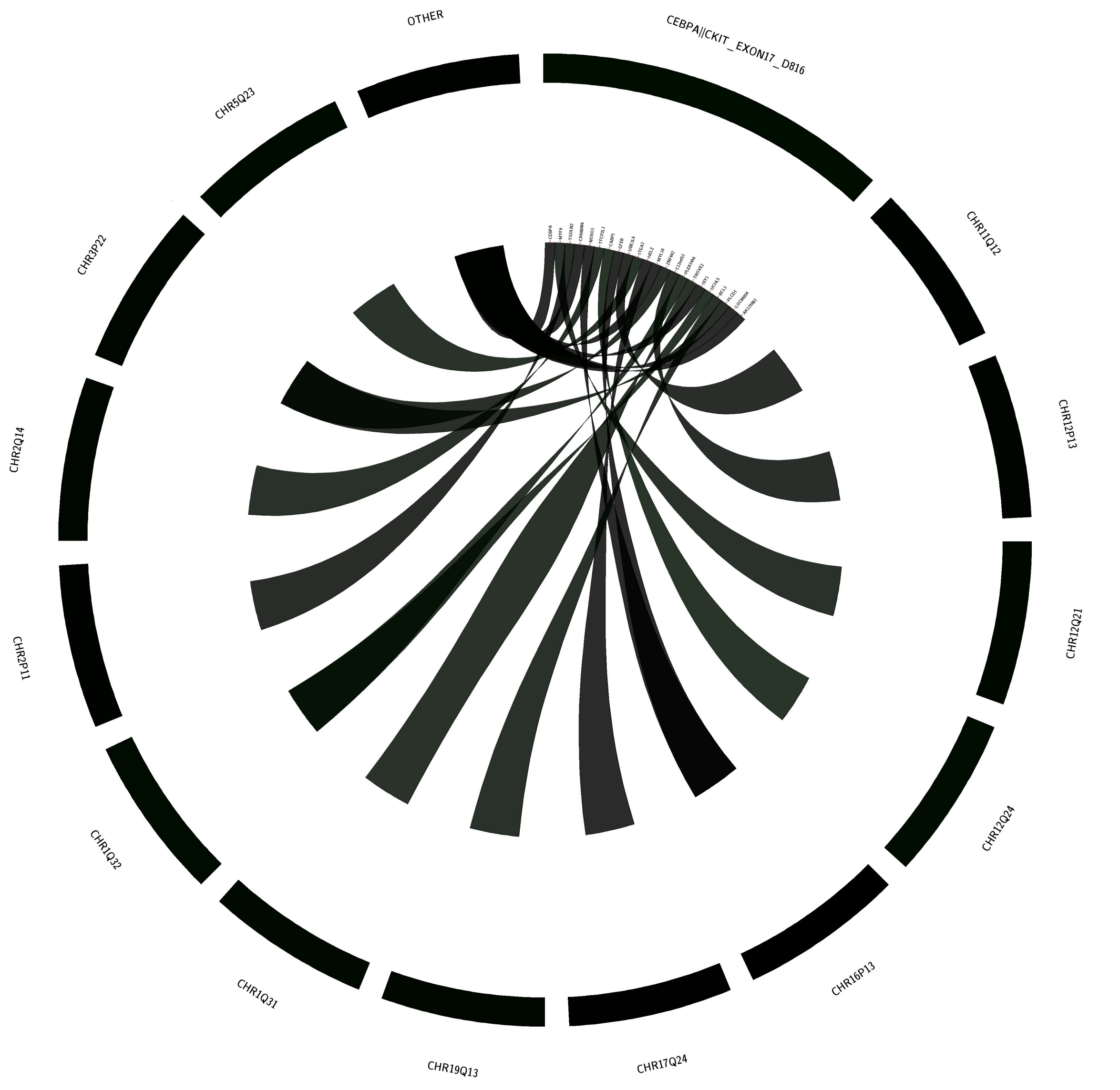
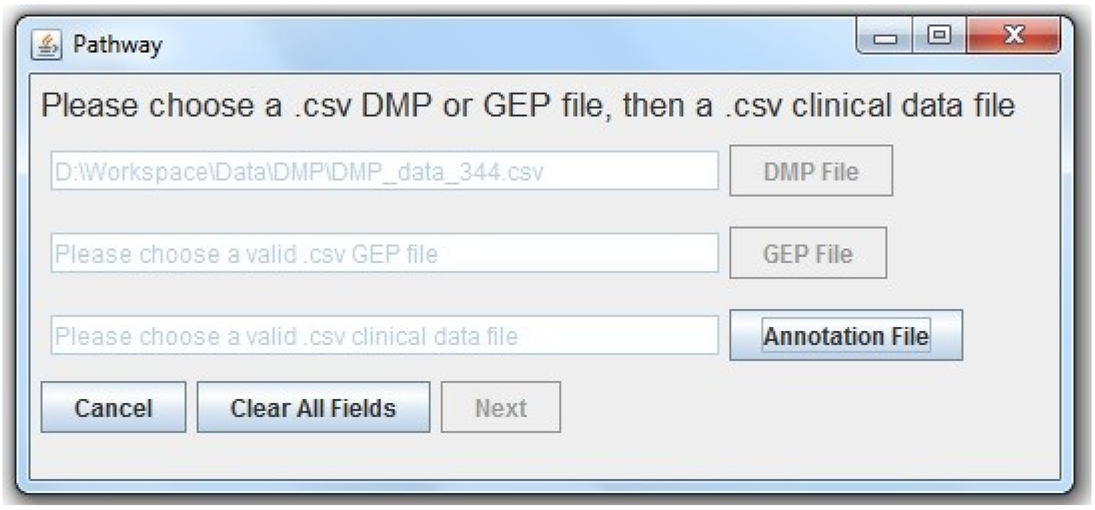


*Figure 7: Expected end result displayed in Circos. Outside the ring are categories of pathways that are significantly related to gene symbols in the two groups chosen before.Gene symbols that not belong to a group in a GSEA file will be connected to 'OTHER'.*

Some screenshots of user interactions in pathway part are shown below.


Pathway

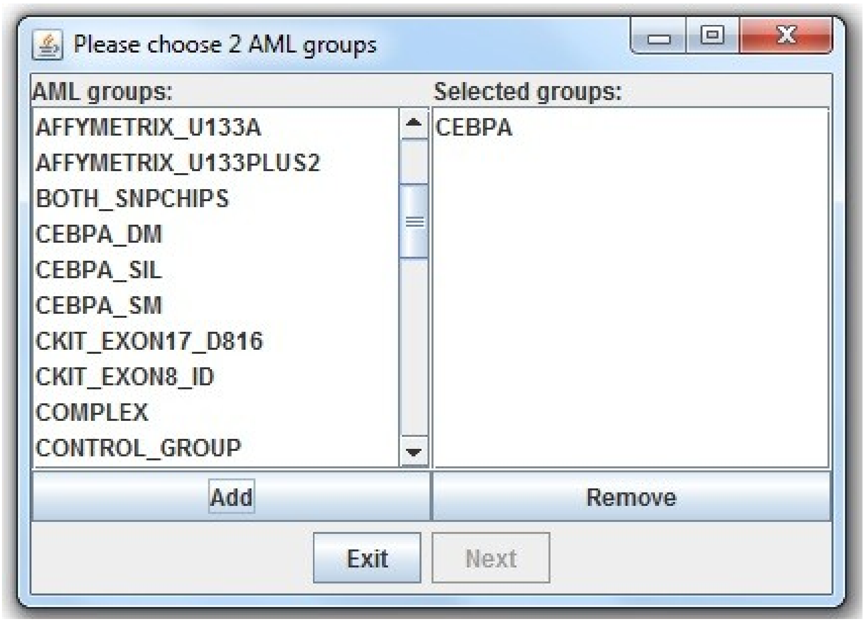
Please choose a .csv DMP or GEP file, then a .csv clinical data file

D:\Workspace\Data\DMP\DMP_data_344.csv

Please choose a valid .csv GEP file

Please choose a valid .csv clinical data file

*Figure 8: First screen after pathway has started. User has to choose one .csv file, DMP or GEP data and one clinical data.*


Please choose 2 AML groups

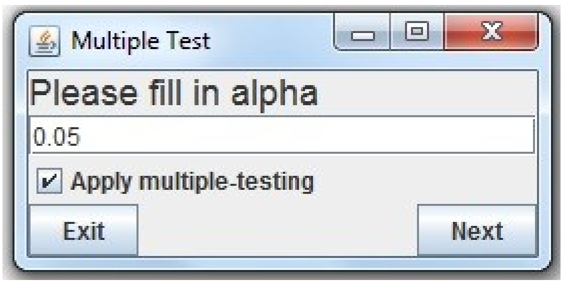
AML groups:

AFFYMETRIX_U133A
AFFYMETRIX_U133PLUS2
BOTH_SNPCHIPS
CEBPA_DM
CEBPA_SIL
CEBPA_SM
CKIT_EXON17_D816
CKIT_EXON8_ID
COMPLEX
CONTROL_GROUP

Selected groups:

CEBPA

*Figure 9: Screenshot when user has to choose two groups within AML. User can add one group at a time from left list to the list of selected groups. The Next button will be enabled when there are two groups are on the right list.*


Multiple Test

Please fill in alpha

0.05

[✓] Apply multiple-testing

*Figure 10: Screenshot when user has to fill an alpha to threshold found p-values. User can check or uncheck the checkbox for multiple-test.*

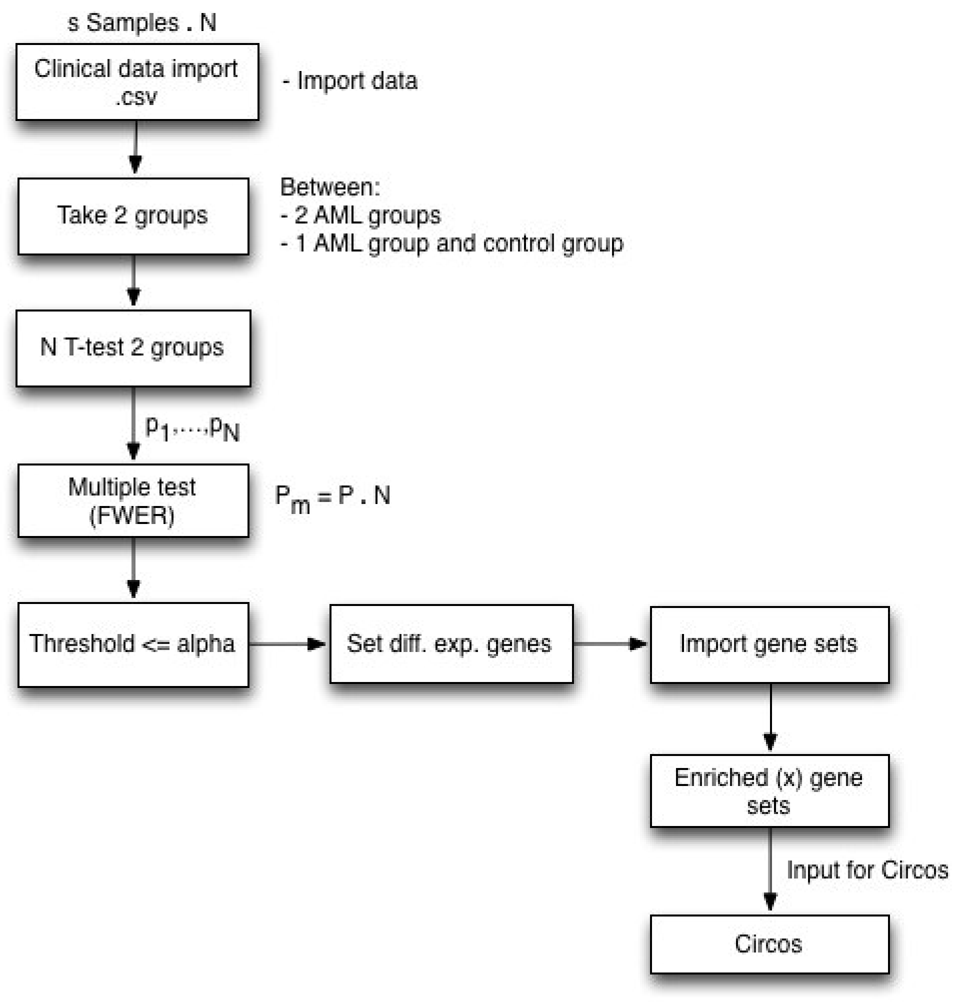


*Figure 11: Function diagram of pathway functionality*

This figure displays the whole process of pathway expansion step by step. It starts with choosing a .csv file (data file) as input and displaying a circular diagram in circos as output. Steps in between input and output will be elaborated in 3.3 below.

## *3.3 Methodology*

This methodology sub-chapter gives explanations about calculations and methods that need to be applied to the data in order to get end result of this pathway functionality: generating an input file for circos.

### 3.3.1 T-Test

After inserting and importing input file to a database file, next step would be to test two groups using t-test. T-test will give the observed significance level or p-value that associated with these two groups, this two-tailed t-test compares the means of these groups, even if they have different numbers of samples. Several p-values ($p_1$,...,$p_n$) are obtained from these p-tests which after that are thresholded by an alpha which filled in by the user. [TTest]

If the user choses to apply multiple-test, obtained p-values from t-tests will be corrected using recently developed multiple-test methods. Suppose that there are 20000 p-values, which means that there are 20000 separate hypothesis tests. If for example, a standard p-value cut-off (alpha) of 0.05 is being used, then 20000*0.05 = 1000 genes would be expected to be “significant” by chance.

### 3.3.2 Multiple Testing

| | $H_0$ is true | $H_0$ is false |
|---|---|---|
| **Do not reject $H_0$** | Correct decision<br>1-α | Incorrect decision<br>1-β |
| **Reject $H_0$** | Incorrect decision<br>α | Correct decision<br>β |
| | α = P(Type I Error) | β = P(Type II error) |

FWER or Family Wise Error Rate is then the probability of at least one type I under the global null hypothesis, FWER = P(V>=1). Back to the example above, suppose that p-values are approximately around $10^{-7}$, then the end result of p-values after being corrected by $10^4$ will be $10^{-3}$. ($P_m$ = P **.** N). [FWER]

Last step of this process is to generate config files that will be used as input for circos given genesets from tests mentioned above.

## *3.4 Implementation*

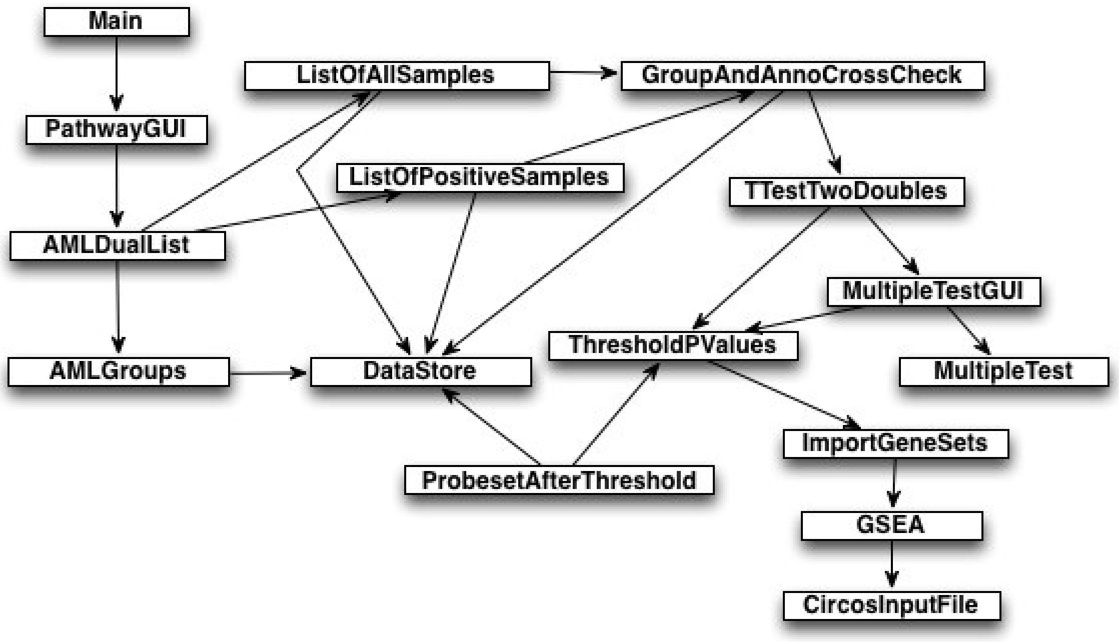


*Figure 12: Simplifed UML diagram of pathway part*

Figure 12 displays global UML diagram of pathway expansion with its main classes (important classes). For more details of this UML, please see appendix at the end of this report.

### 3.4.1 Reading and Importing Files

Data in CSV file format will be read and imported using H2 Database engine. H2 is a relational database management system written in Java and is available as open source software. This database can be embedded in Java applications or in client-server mode. It is possible to create both in - memory tables, as well as disc-based tables and these tables can be temporary tables or even permanent.[H2] It is decided to use the disc-based tables in stead of memory tables, because it turns out after several tests that this program will stop working, it deals with java memory heap space problem. On the other hand, disc-based tables will work slower, but it can handle bigger data.

The main programming APIs are SQL and JDBC, stands for The Java Database Connectivity, the industry standard for database - independent connectivity between the Java

programming language and a wide range of databases - SQL databases and other data sources, such as spreadsheets or flat files. The JDBC API also provides a call-level API for SQL-based database access. It is possible to use Java programming language to utilize “Write Once, Run Anywhere” capabilities for applications that require access to enterprise data.[JDBC]
The JDBC API mainly has three features:

- Establish connection with a database or access any tabular data source
- Send SQL statements
- Process the resultsets

### 3.4.2 Processing data

After importing files to a database, next step would be comparing which sample numbers belong to two groups that have been chosen by the user. This step can be found in class GroupAndAnnoCrossCheck.java. This class has doCrossCheck() method that returns sample numbers that can be found in the annotation file as well as in that groups.

Applying t-test to these sample numbers from two groups is the next step. How many times this t-test are done, would be the same as the total number probesets in the annotation file. Suppose that there are 20000 number of probesets, for every probeset, that is probeset$_1$, probeset$_2$, ... ,probeset$_n$, a t-test is being applied. 20000 p-values from these t-tests will be found. This t-test process could be found in TtestTwoGroups.java.

For Example: in Table 1, four samples are found that are positive to AMLgroup1 and in Table 2 six samples are found from AMLgroup2.

| Probeset | Sample$_1$ | Sample$_3$ | Sample$_7$ | Sample$_{29}$ |
|---|---|---|---|---|
| Probeset$_1$ | x.xxxxxx | x.xxxxxx | x.xxxxxx | x.xxxxxx |
| Probeset$_2$ | x.xxxxxx | x.xxxxxx | x.xxxxxx | x.xxxxxx |
| ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... |
| Probeset$_{20000}$ | x.xxxxxx | x.xxxxxx | x.xxxxxx | x.xxxxxx |

*Table 1: Samples from AML group 1*

| Probeset | Sample$_2$ | Sample$_9$ | Sample$_{30}$ | Sample$_{34}$ | Sample$_{72}$ | Sample$_{219}$ |
|---|---|---|---|---|---|---|
| Probeset$_1$ | x.xxxxxx | x.xxxxxx | x.xxxxxx | x.xxxxxx | x.xxxxxx | x.xxxxxx |
| Probeset$_2$ | x.xxxxxx | x.xxxxxx | x.xxxxxx | x.xxxxxx | x.xxxxxx | x.xxxxxx |
| ... | ... | ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... | ... | ... |
| Probeset$_{20000}$ | x.xxxxxx | x.xxxxxx | x.xxxxxx | x.xxxxxx | x.xxxxxx | x.xxxxxx |

*Table 2: Samples from AML group 2*

Next step, these 20000 p-values might have to be corrected using FWER multiple-test, depends on what the user choses. Class MultipleTest.java will handle this multiple-test. These 20000 found p-values would then have to be filtered or thresholded using alpha that is filled in by the user, a standard alpha of 0.05 would be displayed in the field, but users can fill another alpha value. Suppose that 100 probesets are found from this thresholding process, then this pathway process would go on further with this 100 probesets. These probesets use special probeset annotations that are different from the ones in the Gene Set Enrichment Analysis (GSEA) files. These annotations have to be matched using either DMP-annotation or GEP-annotation file loaded earlier. Results from this matching are gene symbols. This matching process finds place in DMPAnnotation.java or GEPAnnotation.java.

Found gene symbols from steps above will further be matched with one of the five selectable GSEA files. Only one file that has been chosen by the user will be loaded. A matching process finds place after this loading process and it will return a list of pathways or categories that these gene symbols belong to.

These gene symbols and their pathways will then be used in CircosConfig.java to automatically generate input .txt files for Circos. Two HashMaps have to be created in order to create an object. The first one contains mapping between the two chosen groups and their gene-symbols, while the second one contains mapping between gene-symbols to their related pathways. Please refer to Circos implementation part 5.

# Part 4. Alignment

## 4.1 Problem Analysis

It may be useful to the researchers to  view certain parts of their data as graphs, and compare it to other types of data. This can be done using a DNA sequence alignment. A sequence alignment is when the different data points, such as probe sets for expression data, are placed in a track according to their location on a chromosome[Bioinformatics: Sequence and Genome Analysis]. This way, multiple types of data or different samples can be placed in different tracks to be compared. This allows researchers to find certain regions of interest on the genome.

The goal for the Alignment part of the project is to visualize expression, methylation. CNV and SNP data per sample, using tracks. There are already programs able to do this, called genome browsers. These programs are great for comparing different types of data, as long as it is links to specific locations on chromosomes. They often already contain basic information about the different genes on the human genome. A detailed description and comparison between different commonly used genome browsers can be found in the orientation report of this project.

While these genome browsers can certainly be used to visualize the data, they often require very specific input formats and are not very interactive. This part of the project will be to allow the user to not just visualize the data, but also interact with it. The requirements for this project will be to create a tool where the user, after uploading the data, can first select what types of data and samples he wants to show. The user should also be able to set thresholds. This will allow the user to focus on extreme values. Lastly, the user should be able to merge samples into groups. These group samples will show the average values in their tracks. This can be useful when comparing groups, such as control groups versus patients. This tool should then be able to visualize the data quickly, and allow the user to readjust the options.

## 4.2 Design

The alignment part of the project will be a simple tool where the researchers can import the data. This data is then stored in tracks, each assigned to a sample. Each sample has multiple tracks for each type of data. These different types of data are referred to as attributes. The tool will have a simple one window GUI, where the user can perform all the required steps, before pressing a button and visualizing the data. This can be seen in the function diagram shown bellow. Let us first focus on the input.

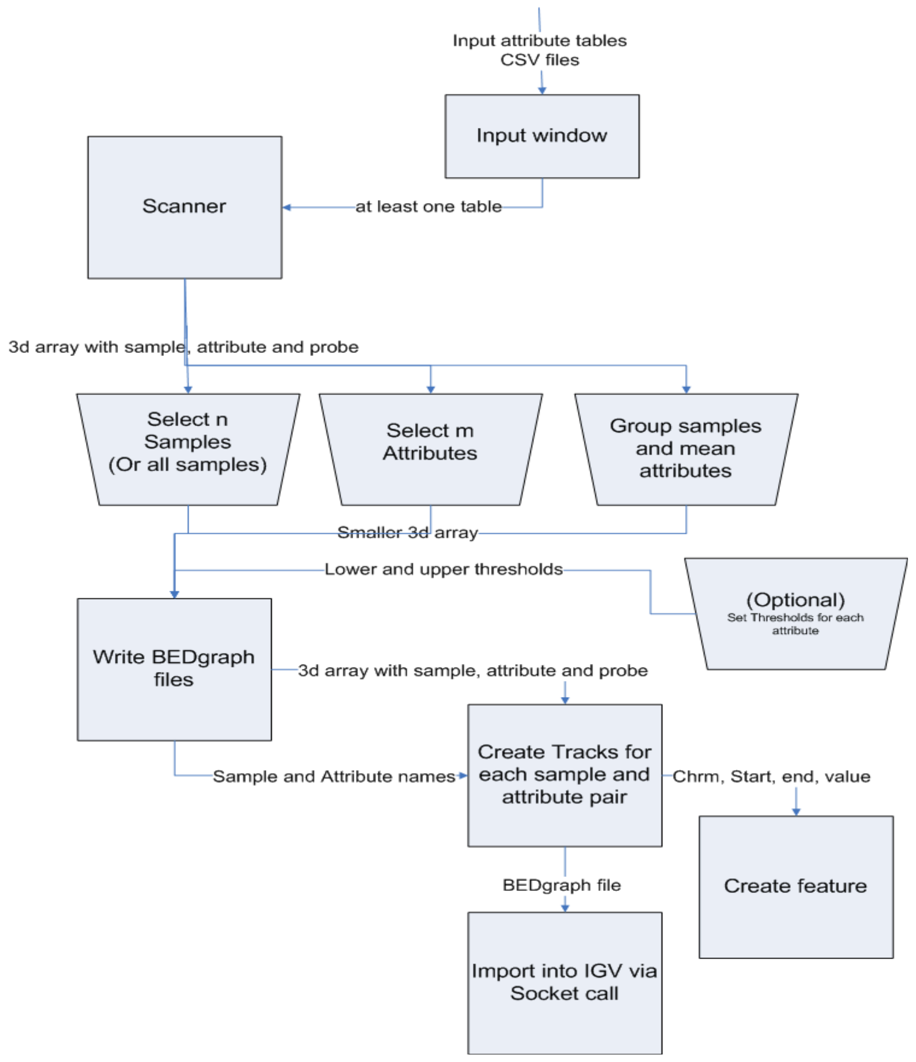

Input attribute tables
CSV files
Input window
at least one table
Scanner
3d array with sample, attribute and probe
Select n Samples (Or all samples)
Select m Attributes
Group samples and mean attributes
Smaller 3d array
Lower and upper thresholds
(Optional)
Set Thresholds for each attribute
Write BEDgraph files
3d array with sample, attribute and probe
Sample and Attribute names
Create Tracks for each sample and attribute pair
Chrm, Start, end, value
Create feature
BEDgraph file
Import into IGV via Socket call

*Figure 1: function diagram, showing the different actions a user can take*

### Input

The input looks the same as in the other parts of this project. Each attribute is imported as a CSV file with a corresponding annotation file. The data file contains a large table. Each row represents a probe set. The first  column give an ID and the following columns represent each sample. The annotation file is also a large table that provides the chromosome number, gene symbol, start and stop for each probe set. The tool will need to collect the data for each probe set out of both files, by matching the names on the

On the GUI, the user should be able to browse to these files. Once the location of both files have been chosen he can give the attribute a name and start importing it. He can repeat this for multiple attributes. He should also be able to delete attributes he no longer needs.

### Interaction

Once one or more attributes have been loaded the user should be able to do some simple tasks. The user should be able to see what attributes are loaded. He should also be able to see how many samples have been loaded and how many of those are selected.

He can then enable and disable certain samples and attributes. This will give the user control over what tracks should be visualized. He should also be able to set thresholds per attribute, so that he can look for extreme values. Finally, he should be able to make sample groups, where multiple samples are merged into one sample, by averaging out all the tracks per attribute. This could be useful when a researcher wants to compare groups of samples.

A drop-down box should allow the user to easily select an attribute and get some basic information about this attribute, such as its name, if the attribute is enabled and disables and what the current thresholds, if any, are. The user should then be able to change this information via some text-fields and buttons.

An effective way to allow the user to enable or disable certain samples is to keep a list of sample names. The user should be able to add or remove names from this list. The user should be able to add or remove multiple samples at a time, perhaps by typing multiple sample names into a text-field.

The merge function will work in a similar way. The user can type multiple sample names into a text field. Using a button, the user can then create a new sample, containing the average values of all the input samples, and automatically add it to the list to be displayed.

**Clinical data searcher**

To help the user find samples he might want to select or merge, an extra tool will be created. This tool can select a clinical data file. This is also a csv file, containing properties of each sample (test subject) such as age and gender. After browsing to such a file, the user can give a property to search on as well as the requested value.

To give an example, consider a clinical data file with three columns. The first column gives the sample-ids. The second and third column give the age and gender respectively. The first row gives the name of each column, and all following rows the values for each sample. The user can select “gender” as the search property and “male” as the requested value. This will return the sample-ids of all male samples. These ids can then be used to select, deselect or merge the samples as explained previously.

**Visualization**

Once the user is done modifying the tracks, he should be able to visualize the tracks. This should be a relatively quick process, so that the user can go back, make new modifications and visualize again. At first, an implementation using awt and swing was considered, but it is more practical to use an existing genome browser for this. This will allow the user to also compare other data to the data visualized by the tool.

In the orientation report, a comparison of different genome browsers is given. Integrative genomics viewer (IGV) was chosen, due to its simple java based api and the ability to run it locally, instead of through a website.[IGV user guide]

The tool will create output files, which can be loaded into IGV using a socket call. If the user does not have IGV open at the time that these files are generated, he can still import them manually into IGV later. In both cases, the results will look similar to the picture below.

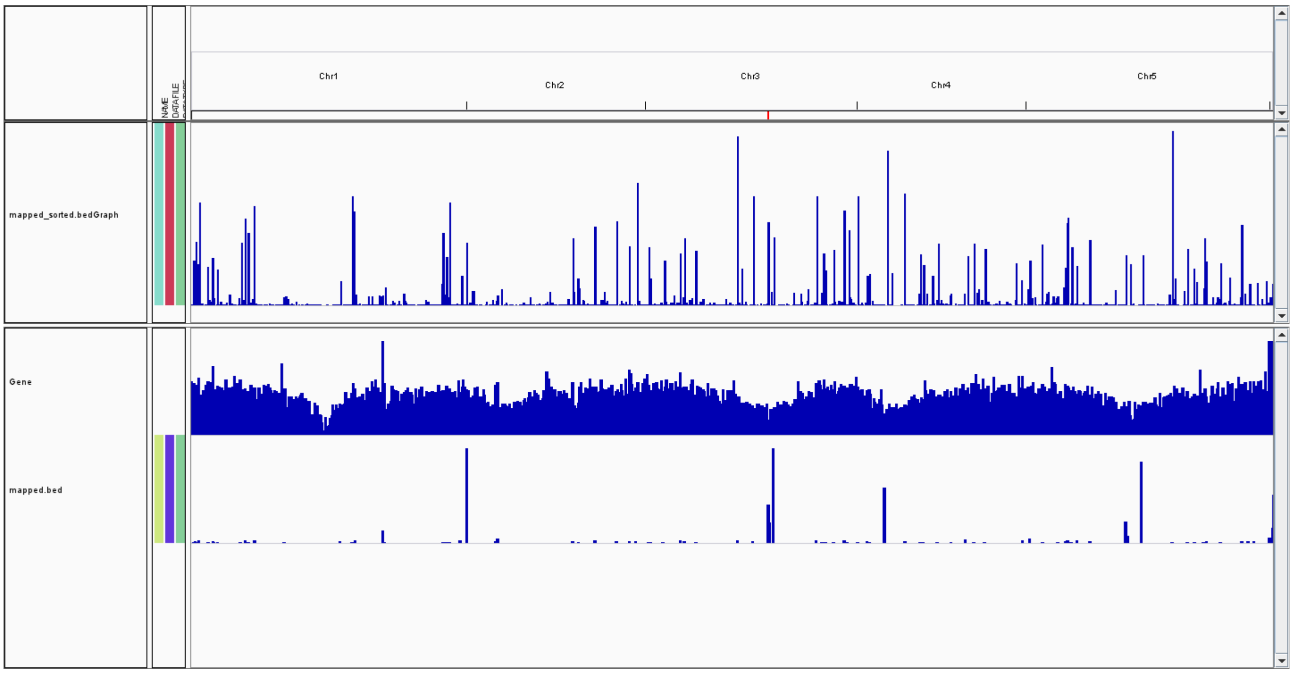


*Figure 2: A screenshot of IGV, showing different data tracks*

## 4.3 Implementation

The design of the Alignment part of the project can be divided into three distinct functions: reading the input, allowing the user to manipulate the data and visualizing the output. These functions are performed using a GUI called AlignmentWindow and a manager class called Alignment.

**Reading the input**

The reading of the input works much alike the other parts of the project. For each attribute (RNA expression, methylation, CNP, SNPs) a file must be uploaded with the values for each probe and sample pair, as well as an annotation file to map the probes to the correct location on the correct chromosome. The Alignment class contains an instance of a class called AlignmentParser, whichs reads and stores the data. Due to the similarities, an abstract class called FileParser, which is used throughout the entire project to parse CSV files, is inherited by the class AlignmentParser. The AlignmentParser in turn also uses another sub-class of FileParser called AnnotationParser.

To store the data, the program uses a Probe objects, each containing the chromosome number, start and end value for a specific probe set. These objects are basically little containers for the data. For performance, these containers need to remain as small as possible, which is why they do not contain a probeset id or sample value. Two other container classes, annotationProbe and ValueProbe, in turn each contain a Probe objects, but add a name for the Probeset (the Probeset id used in the input files) and the value for a specific sample respectively. AnnotationProbes are used during the loading process and ValueProbes are finally stored until the program terminates. The ValueProbes are stored in Track objects. A sample object contains multiple of these tracks representing each attribute. Finally, an ArrayList of Sample objects is stored in the Alignment class, which will perform most operations. This can be seen in the UML.

Once activated, the AlignmentParser will first call upon AnnotationParser to provide a full list of annotation probes, containing the name, chromosome number, start and stop for every probe set, but no values. This data is obtained from the annotation file. The AlignmentParser will then read each row of the data file, each row corresponding to a probe set. For each row, it will search for the corresponding AnnotationProbe. It will then create a ProbeRow objects, containing the Probe object contained in the AnnotationProbe as well as a list of values for each sample column, stored as a float. A list of these ProbeRows is then loaded into the Allignment class. The Alignment class also receives a list of sample names from the parser, which matches the order of the ValueProbes. Using this list and the list of ProbeRows, ValueProbes for each sample and probe set pair are created. These ValueProbes are stored in separate tracks for each samples. The Alignment class will then generate a sample object to attach the track to. If the sample already exist, because a previously loaded file also generated a sample with the same sample-id, the track is added to that sample object, instead of generating a new one. Because of the high performance requirements for all these processes, the entire loading process is done on a background thread.

**Data manipulation**

Once the data is stored, it can be manipulated. The Alignment class contains a list of selected samples as well as a list of attribute objects. These objects contain a boolean to see if it is enabled or disabled as well as the lower and upper thresholds. When the write method of Alignment is called, the attribute list will be used to determine what tracks and Probes are displayed. This will be explained further in visualizing the input.

Each attribute object in the attribute list can be modified via the GUI. A dropdown box will allow the user to select the attribute he wants to modify. A button can set the enabled boolean to true or false. Another button and two text-fields will allow the user to set the thresholds of the attribute. Lastly, the user can delete the attribute completely.

The GUI also enables the user to modify the selected samples. This is done by a large text field and four buttons. The four buttons are “select”,”deselect”, “select all” and “deselect all”. In the text-field the user can enter multiple sample names separated by semicolons. The smart thing about this is that these names can be copied easily from the input CSV files. These sample names are then used for the select and deselect buttons. The alignment class will then search for those samples and set a boolean called “selected” to true or false.

The merge function works similar. A large text-field will allow the user to enter multiple sample names. A merge button will then call the the merge method of Alignment. The merge method of Alignment will accept an array of sample names. It will then look up the corresponding samples. It will then retrieve all the tracks from the samples and call a helper method attributeMerge(Track[] tracks) for each attribute.

This helper method will return a new Track, by averaging out all the probes of the input tracks. The Track class has a sum(Track other) and divide(int n) method for this purpose. The merge method can then use the newly created tracks to create a new Sample. This Sample is then added to the ArrayList.

**Clinical Data Searcher**

The clinical data searcher is a simple gui where a file location can be given (with the help of a file chooser) as well as a search column and search value. A search button will create a clinicalparser object, which will read the selected file, searching the search column for the requested search value. This parser will then print the corresponding sample names in a results area.

If the search column is empty, the parser instead merely provides the first line of the csv file. This gives a list of column names, from which the user can select. If the search value is empty, the parser will print a list of all unique values in the search column. This to allows the user to make a selection.

Once the user has successfully searched for a list of samples, three buttons allow him to immediately select, deselect or merge those samples. The implementation for this is simple. The clinical data searches simply copies the list into the select or merge field of the main gui and calls the doClick() method of the corresponding button.

**Visualizing the output**

When the user is done, he can press a button to visualize the data. The program will produce an output file for each Track in the bedGraph format. In the bedGraph format, the first line of the file gives some basic information about the track, such as the name that has to be displayed. The following lines provide the probe information, with the chromosome number, start, end, and value separated by spaces.

This output file is created using write methods in Probe, Track, Sample and Alignment. These method must also check to ensure that only the required data is written. The write method in Alignment will check what samples are selected. It will then call the write method of the selected samples, with the boolean and threshold lists as parameters. The write method of the samples will then check the boolean list to see what Tracks must be written. The write method of the Tracks receive the thresholds that apply to them. These two integers are then passed to the Probes write method, which will check its own value to determine whether it needs to write. If so, it will write its own data as a line in an output BEDgraph file.

Using a socket, these BEDgraph files are loaded into IGV via a port call. This is done using a class called AllignmentSocket, which handles the connection to IGV. That means that if IGV is running and set to allow port commands, this program can directly load the data into IGV to be visualized. This option allows the user to make further adjustments to the visualization in IGV and compare the data to other tracks loaded into IGV.

# Part 5. Circos

Circos is a visualization tool used to generate circular visualizations in Part 2 and Part 3.

## *5.1 Why Circos?*

Circos is an established visualization tool for generating circular visualizations of data which contain a lot of relationships between objects.

## *5.2 Implementation*

To integrate Circos into Part 2 and Part 3, a Java wrapper around the original Circos perl package had to be implemented.

In Figure 13 it is visible that there are 3 classes dedicated to export files: Genes, Karyotypes and Links. Each of these writes an output data file for Circos, while CircosColors writes its output to a predefined circos configuration file (circos.conf) which is written to the user's disk by CircosConfig. In the next section the structure of Circos is explained, what it requires and what the Circos API implementation can provide. After that the specific implementations of CircosConfig: CircosConfigCorrelation and CircosConfigHypergeo will be explained.

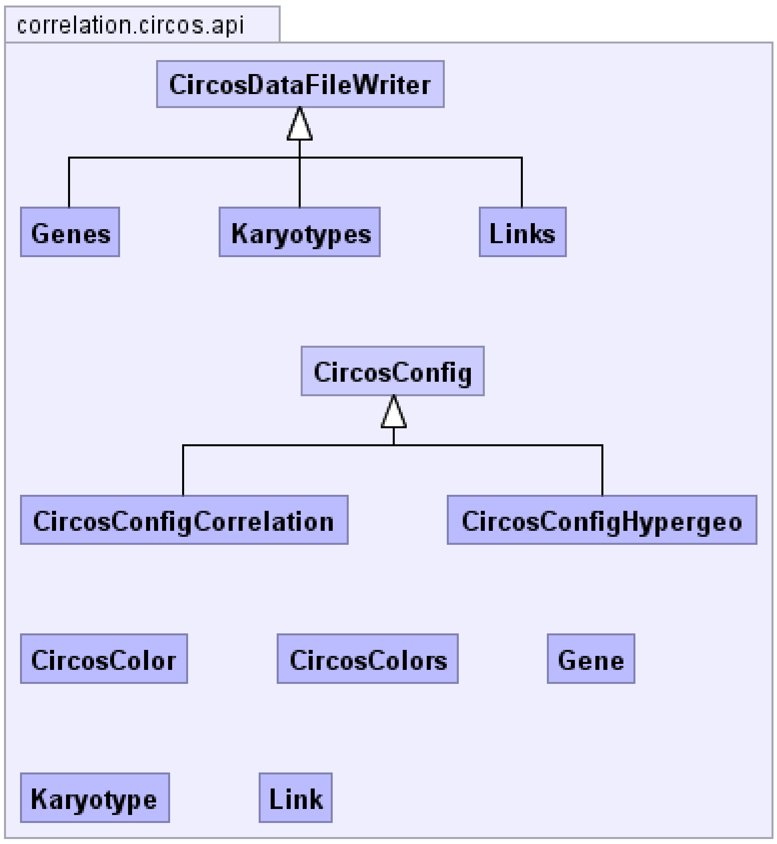


*Figure 13: Circos API code structure*

### 5.2.1 Structure of Circos

To be able to generate an image with Circos, the following files need to be generated:

- a Circos configuration file
  - this file contains settings for the size and height of the image, spacings, etc file location of the genes, karyotype and links file
- a genes file
  - this file contains the genes and its location on a certain chromosome
- a karyotype file
  - this file contains the size, name and color of each chromosome
- a links file
  - this file contains source and destination locations where links needs to be drawn

On top of that the following files need to be copied to the system

- circos-0.56
  - Circos script
- custom fonts
  - Circos uses certain custom fonts which need to be included
- bundlelinks tool
  - this is a tool to reduce the number of links in the links file given a minimum link span.

### *5.2.2 Circos API specifications*

The implementation of the Circos API does not contain the entire set of functionality of the Circos perl package, but it is a subset which suffices for this project.

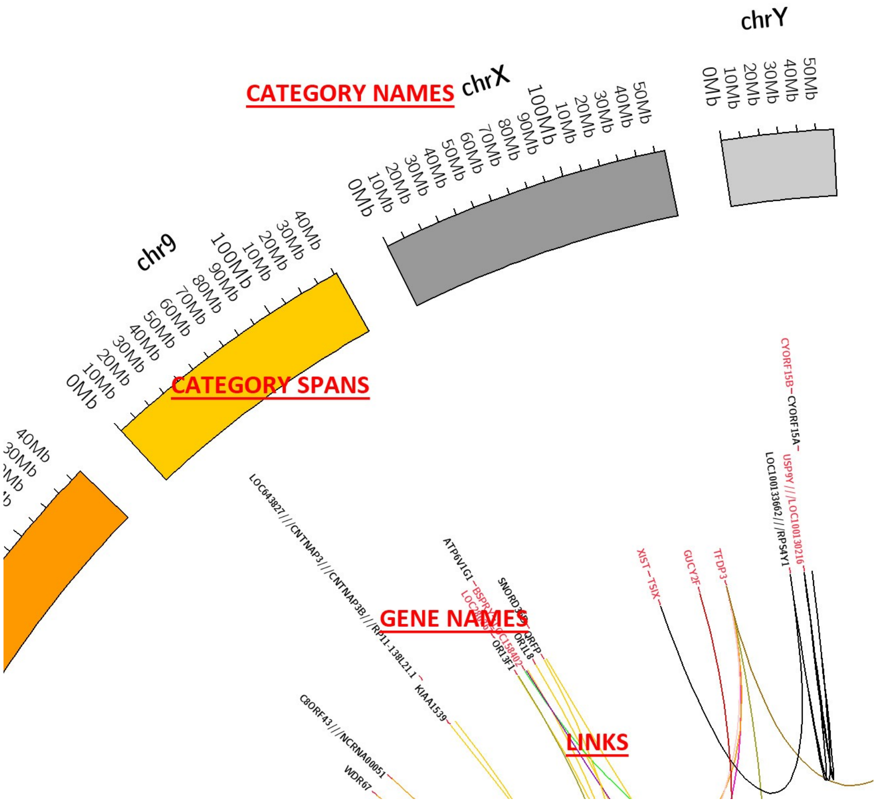


*Figure 14: Screenshot of a part of a circular diagram generated using CircosConfig*

This subset allows

- the display of gene-names at specific locations on a circular diagram
- the display of links between 2 arbitrary locations on a circular diagram
- the display of category span names on a circular diagram
- the user to select any color each category span on a circular diagram
- the user to select any color for links
- bundling of links

All of these functionalities are displayed in Figure 14 except for the bundling of links.

## CircosConfigCorrelation implementation

The specific implementation for the Correlation approach in Part 2 for Circos was quite straightforward. As correlations are calculated, each correlation value is linked to two genes. Given a threshold, a correlation value becomes a link when the $|(correlation\,value)| \geq threshold$ , the names of the genes can be extracted by looking the data up with the GeneSymbols (see Appendix), and at the same time the chromosomes can be looked up by extracting the data from gene-symbols as well. These links, genes and chromosomes are then put into the CircosConfig.

## CircosConfigHypergeo implementation

The implementation for the Pathways approach in Part 3 for Circos is even more simple than the

CircosConfigCorrelation. The Pathways approach has a set of genes which are linked to two group and multiple pathways. The set of genes can be enumerated at the group's category span and links can then be drawn between the groups and the multiple pathways modeled as category spans.

# Part 6. Evaluation

This chapter will discuss the results of the project. This includes how the quality of the produced programs have been measured as well as the clients feedback.

Part of the evaluation process is testing. Both automatic and manual testing has been used, throughout the project, to ensure that all the code does what it is supposed to do. This includes unit testing, where individual classes and methods are tested, as well as manual functional testing, where the entire program is run to see if it performs as required by the design.

The second part is ensuring that the program meets all the demands of the client. Throuhgout the project, the client has frequently tested the final build and provided featback, such as new feature requests or usability issues.

Listed below are both the testing methods used in each part of the project as well as the clients comments regarding the final build.

## *6.1 Correlation*

Complex methods have been tested using automated unit testing and testing of the user-interface has been done with a Test Plan (see Appendix: Correlation Test Plan).

## *6.2 Pathways*

## *6.3 Alignment*

Most classes and methods in the alignment tool have been tested using automated unit testing using Junit, an Eclipse plugin. However, the functionality of the gui and the output generation and visualization in IGV have been tested manually.

The Alignment tool was able to meet most requirements but not all. For example, the client requested the feature to give different data types, different colors in IGV. He also asked that the thresholds feature would highlight certain probe sets, instead of hide all other probe sets. These requests appeared difficult to implement in IGV. However, all the requirements requested originally where met.

One problem with the program is its memory usage. The loading of large input files can sometimes take up to 10 minutes and the java heap size can get quite large. This requires the user to adjust his maximum heap size if he wants to load large files. Visualization can also be slow, if the user wants to display a large number of samples in IGV.

# Part 7. Conclusion and recommendations

This chapter is a summary of the overall results of the project and gives recommendations to improve the created tools.

### *Correlation*

The aim of this part of the expansion was "*Calculating the correlation between gene expression profile data and DNA methylation profile data and consequently displaying the correlations in a circular diagram given a certain threshold.* "

The part of the expansion does exactly this and more. During the project new requirements such as filtering by samples came up which defined this part of the expansion in more detail. But while the application is functioning properly when used as explained in the manual, it misses error messages and bug reporting functionality; at some points it's not clear for the user whether an error has occurred in the background or that the program is still running.

### *Pathways*

This program part is able to produce a visualization in a circular diagram, but there are still plenty of improvements need to be considered. First of all, much faster processing is inquired, it takes at least 5 minutes, from starting the program part until an image shows up. Optimization is highly recommended regarding this pathways part. Stability is also an issue, there are not enough warning showed during process. Program part will stop working when an error occurs without showing any warning, it also has to do with a better user-friendly user-interface.

As mentioned in part 3, this program part imports data into a database. Database file can be temporary stored in memory or in the disc. Perhaps another approach of importing data might have worked better, more stable and even more robust.

### *Alignments*

While not all requested features where able to be implemented, the program can perform most of the required functions. For other features, such as changing the color of tracks or being able to highlight certain probes, other output file formats or genome browsers would need to be considered.

The greatest problem with the program is its performance. In hindsight, a better implementation would not have stored all the values in memory. Instead, the program should only read values when it needs to. For example, if the user wants to load in a data file, select a sample and visualize only that sample, a better implementation would first only read the sample names, then allow the user to select the sample, and then read the corresponding values, to generate the output file, that must be loaded into IGV.

## Bibliography

Grewal SI, Rice JC: Grewal SI, Rice JC, Regulation of heterochromatin by histone methylation and small RNAs", 2004
Circos: Martin I Krzywinski, Jacqueline E Schein, Inanc Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J Jones, and Marco A Marra, Circos: An information aesthetic for comparative genomics, 2009
MolDevices: Molecular Devices, Inc, Pathway Analysis, 2011, http://www.moleculardevices.com/Applications/Drug-Discovery/Pathway-Analysis.html
GSEA: BROAD Institute, Gene Set Enrichment Analysis, 2011, http://www.broadinstitute.org/gsea/index.jsp
TTest: Richard Lowry, Concepts and Applications of Inferential Statistics, 1999, http://faculty.vassar.edu/lowry/ch11pt1.html
FWER: Mark J. van der Laan, et al., Multiple Testing. Part III. Procedures for Control of the Generalized Family-Wise Error Rate and Proportion of False Positives, 2004
H2: H2, H2 Database Engine Version 1.3.164, 2012, http://www.h2database.com/html/main.html
JDBC: Oracle, JDBC Overview, 2011, http://www.oracle.com/technetwork/java/overview-141217.html

Bioinformatics: Sequence and Genome Analysis: Mount D.W., Bioinformatics: Sequence and Genome Analysis, 2004
IGV user guide: Judy McLaughlin, IGV user guide, 2011

# Appendix: Problem Description

Acute myeloid leukemia (AML) is a form of leukemia. Leukemia is a type of cancer of the blood, the adjective myeloid suggests that the cancer is related to the bone marrow or spinal cord. There are different subtypes of AML, each marked by certain combination of genetic mutations. The Erasmus University Medical Center Rotterdam is conducting several researches in determining many subtypes of AML.

At this moment, some researchers at the Erasmus University Medical Center Rotterdam use an existing visualization tool, called HeatMapper, a visualization tool that allows accurate and rapid interpretation of the data obtained by large scale gene expression profiling. Our assignment will be to expand the HeatMapper which is developed by several researchers of the Department of Hematology and the Department of Bioinformatics at the Erasmus University MC Rotterdam.

The expansion of the HeatMapper should aid researchers at the Erasmus University Medical Center Rotterdam as well as other researchers in similar field.

The expansion should consist of three new functions:

- Calculating the correlation between gene expression profile data and DNA methylation profile data and consequently displaying the correlations in a circular diagram given a certain threshold.
- Displaying pathways of a gene expression profile or a DNA methylation profile in a circular diagram that significantly related to specific genes.
- Create a tool to easily align expression, methylation, SNP and CNV data in a genome browser.

# Appendix: Correlation class diagrams

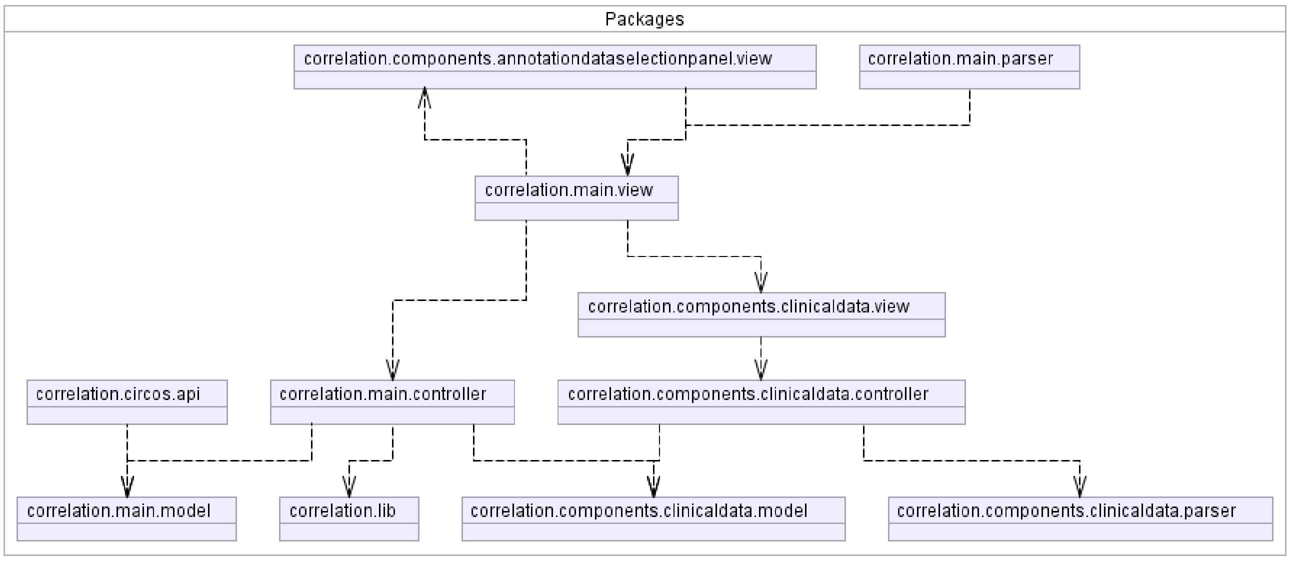
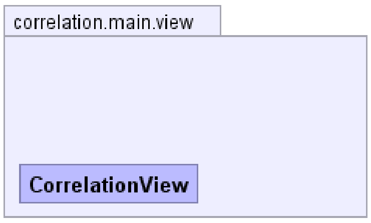

Packages
correlation.components.annotationdataselectionpanel.view
correlation.main.parser
correlation.main.view
correlation.components.clinicaldata.view
correlation.circos.api
correlation.main.controller
correlation.components.clinicaldata.controller
correlation.main.model
correlation.lib
correlation.components.clinicaldata.model
correlation.components.clinicaldata.parser

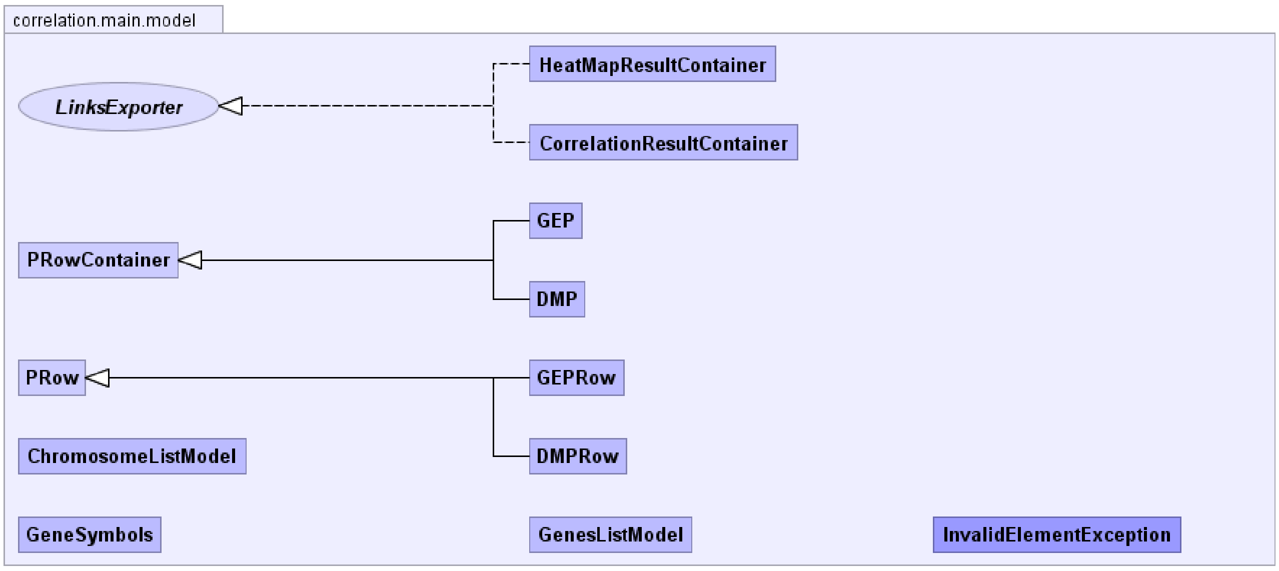
*Figure 15: The class overview*

In the pages hereon, each class will have a short description.

## *correlation.main*

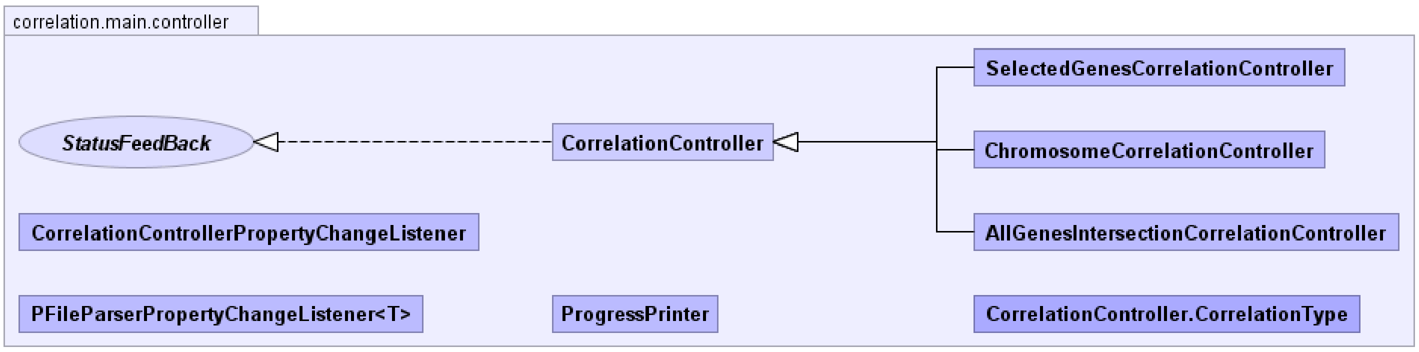


- CorrelationView
  - This is the user-interface of the application.

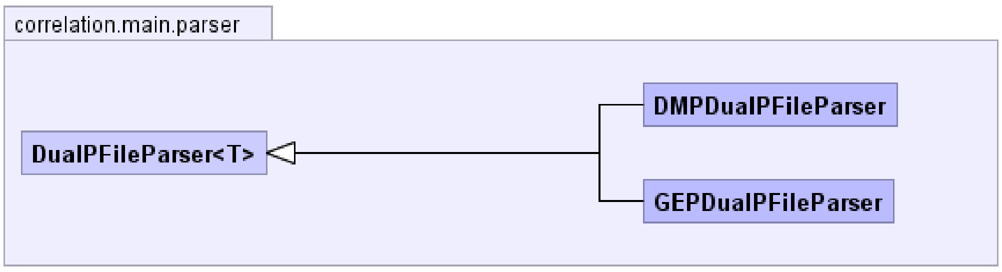


- LinksExporter

    - An interface for correlation value containers to export links
- HeatMapResultContainer
    - A data structure which saves a heat map in a one-dimensional array
- CorrelationResultContainer
    - A data structure which saves correlations in a two-dimensional array
- PRowContainer
    - A container for PRow objects
- GEP
    - A container for GEPRows, it represents the data extracted from the GEP annotation and data files
- DMP
    - A container for DMPRows, it represents the data extracted from the DMP annotation and data files
- PRow
    - A data structure which holds the variables needed to identify a row in a an annotation and data file.
- GEPRow
    - A data structure which represents a row in the GEP annotation and data file.
- DMPRow
    - A data structure which represents a row in the DMP annotation and data file.
- GeneSymbols
    - A static class which is thread-safe, it contains data about every gene parsed into the application.
- ChromosomeListModel
    - Model for an element in the user-interface
- GenesListModel
    - Model for an element in the user-interface
- InvalidElementException
    - An exception thrown when an element in a parsed row in either the annotation or the data file is deemed invalid.


correlation.main.controller
StatusFeedBack
CorrelationController
SelectedGenesCorrelationController
ChromosomeCorrelationController
AllGenesIntersectionCorrelationController
CorrelationControllerPropertyChangeListener
PFileParserPropertyChangeListener<T>
ProgressPrinter
CorrelationController.CorrelationType

- StatusFeedBack
  - An abstract class which lets the CorrelationController dispatch status messages to the user interface (incomplete)
- CorrelationController
  - An abstract class containing all the methods to correlate selected genes and samples.
- SelectedGenesCorrelationController
  - The controller for the selected genes tab.
- ChromosomeCorrelationController
  - The controller for the chromosomes tab
- AllGenesIntersectionCorrelationController
  - The controller for the all genes intersection tab
- CorrelationControllerPropertyChangeListener
  - A listener for CorrelationController which keeps track of the progress of the calculations.
- PFileParserPropertyChangeListener
  - A listener for the PFileParser classes, it also keeps track of the progress of the parsing of files.
- ProgressPrinter
  - A class dedicated to printing the progress in the console for debugging purposes.
- CorrelationController.CorrelationType
  - An enum which holds the type for each of the 3 children classes of CorrelationController.



- DualPFileParser<T>
  - A class for parsing 2 files at once namely annotation and data files.

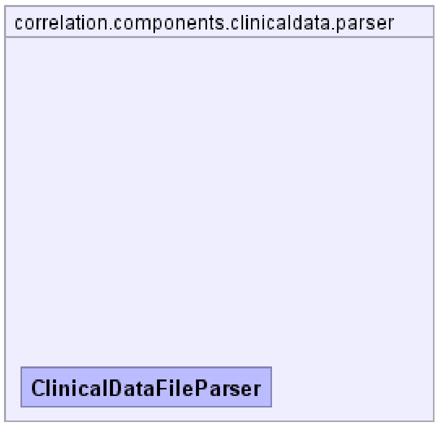
- DMPDualPFileParser
  - A class for parsing DMP annotation and data files
- GEPDualPFilePaser
  - A class for parsing GEP annotation and data files

### *correlation.lib*

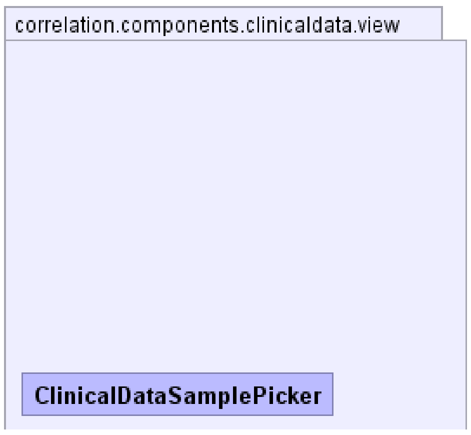
correlation.lib
CorrelationSet
PearsonsCorrelationBoxedFloat

- CorrelationSet
  - A static class which contains frequently used methods for calculating set operations on correlation results
- PearsonsCorrelationBoxedFloat
  - A class which is a wrapper around the Apache Commons Math library component, it has extended the existing class to support arguments for boxed floats.

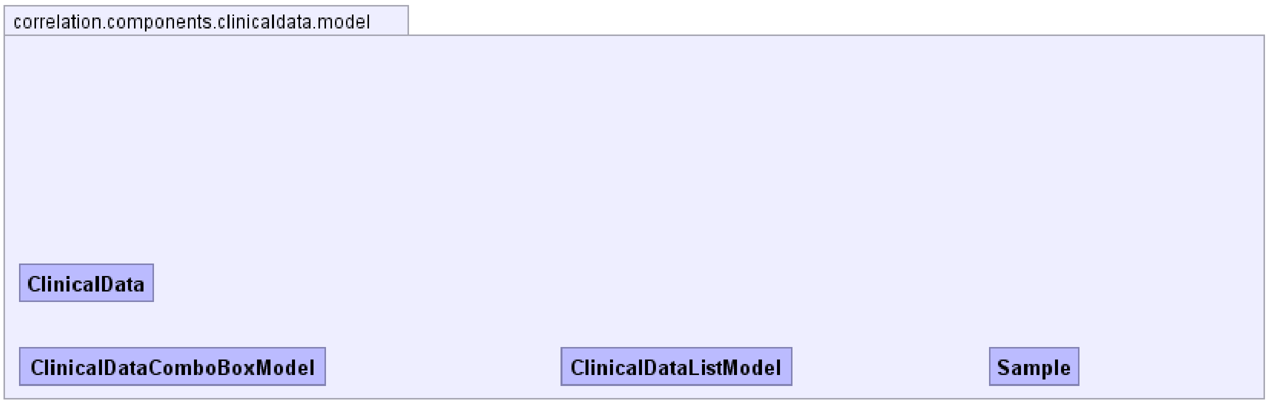
### *correlation.components.clinicaldata*



- ClinicalDataFileParser
  - A parser for clinical data files



- ClinicalDataSamplePicker
  - The user-interface component for selecting samples based on properties in clinical data.

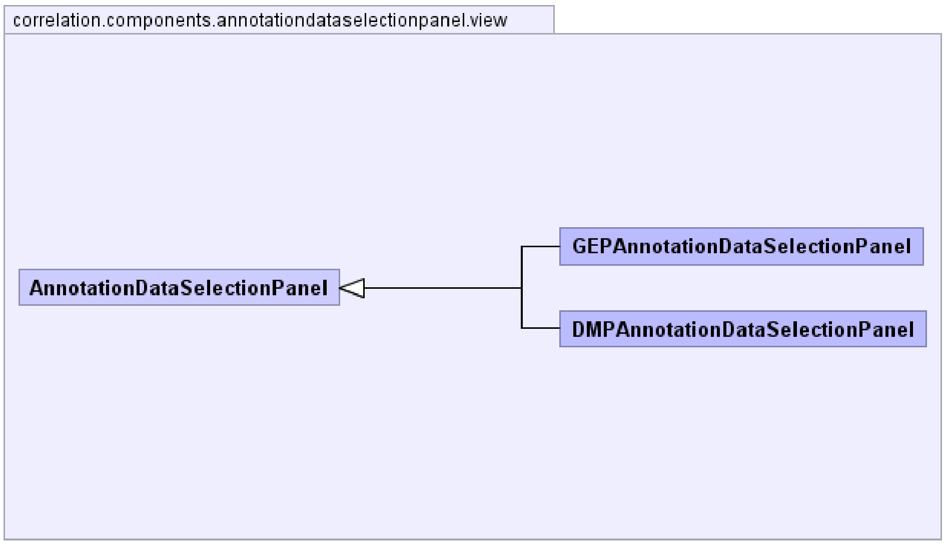

correlation.components.clinicaldata.model
ClinicalData
ClinicalDataComboBoxModel
ClinicalDataListModel
Sample

- ClinicalData
  - A data structure for holding clinical data. It also serves as a model for selecting and deselecting certain clinical data attributes
- ClinicalDataComboBoxModel
  - A model for a user-interface ComboBox component
- ClinicalDataListModel
  - A model for a user-interface List component
- Sample
  - A data structure holding the attribute to value mappings of clinical data.

correlation.components.clinicaldata.controller
ClinicalDataFileParserPropertyChangeListener

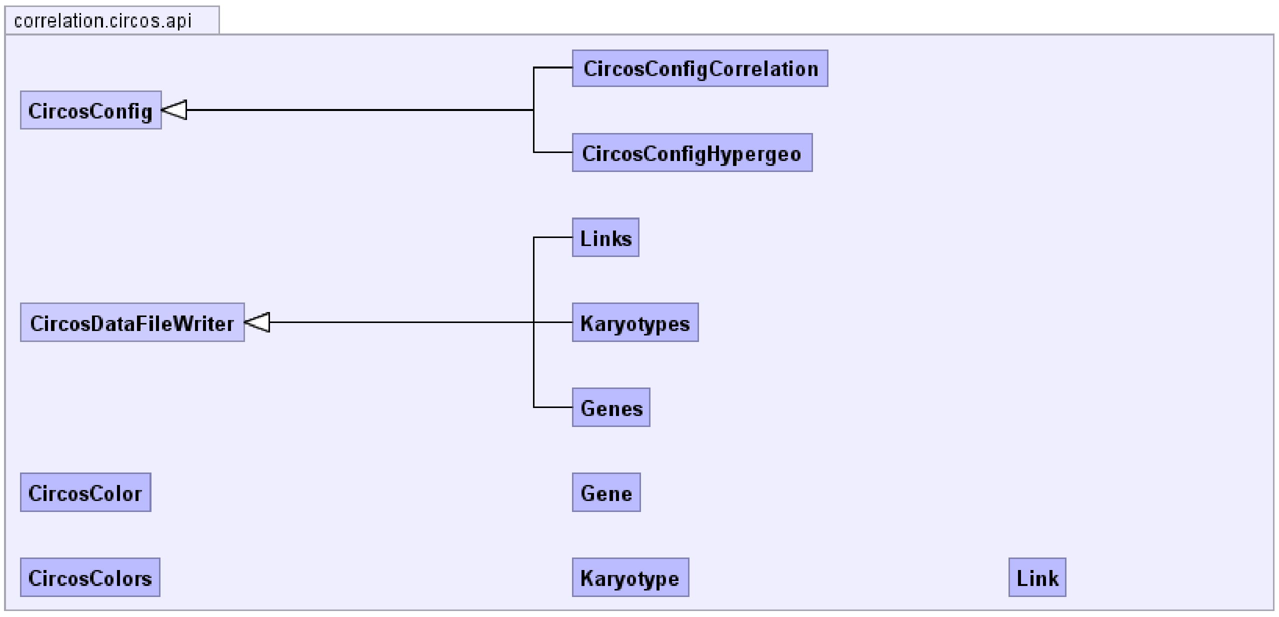
- ClinicalDataFileParserPropertyChangeListener
  - A listener which listens to which values at which attributes are selected.

### *correlation.components.annotationdataselectionpane*


correlation.components.annotationdataselectionpanel.view
AnnotationDataSelectionPanel
GEPAnnotationDataSelectionPanel
DMPAnnotationDataSelectionPanel

- AnnotationDataSelectionPanel
    - A user-interface component for selecting a annotation and data file
- GEPAnnotationDataSelectionPanel
    - A user-interface component for selecting a GEP annotation and data file
- DMPAnnotationDataSelectionPanel
    - A user-interface component for selecting a DMP annotation and data file

## *correlation.circos.api*



- CircosConfig
  - An abstract class for generating Circos configuration files and for running the perl Circos script
- CircosConfigCorrelation
  - A specific implementation of CircosConfig made for the Correlation approach
- CircosConfigHypergeo
  - A specific implementation of CircosConfig made for the Pathways approach
- CircosDataWriter
  - An abstract class which allows classes to write a data string to a given file
- Links
  - A class representing a links data file for Circos
- Karyotypes
  - A class representing a karyotypes data file for Circos
- Genes
  - A class representing a genes data file for Circos
- CircosColors
  - A data structure for storing all the colors used in CircosConfig
- CircosColor
  - A single color definition for Circos
- Gene
  - A data structure for saving the gene name, chromosomal location
- Karyotype

    - A data structure for saving the category span
- Link
    - A data structure for saving a Circos link.

## Appendix: Class Diagram Alignment

This appendix contains the UML class diagram of the Alignment part of the project. The Alignment part of the project is explained in chapter 4.

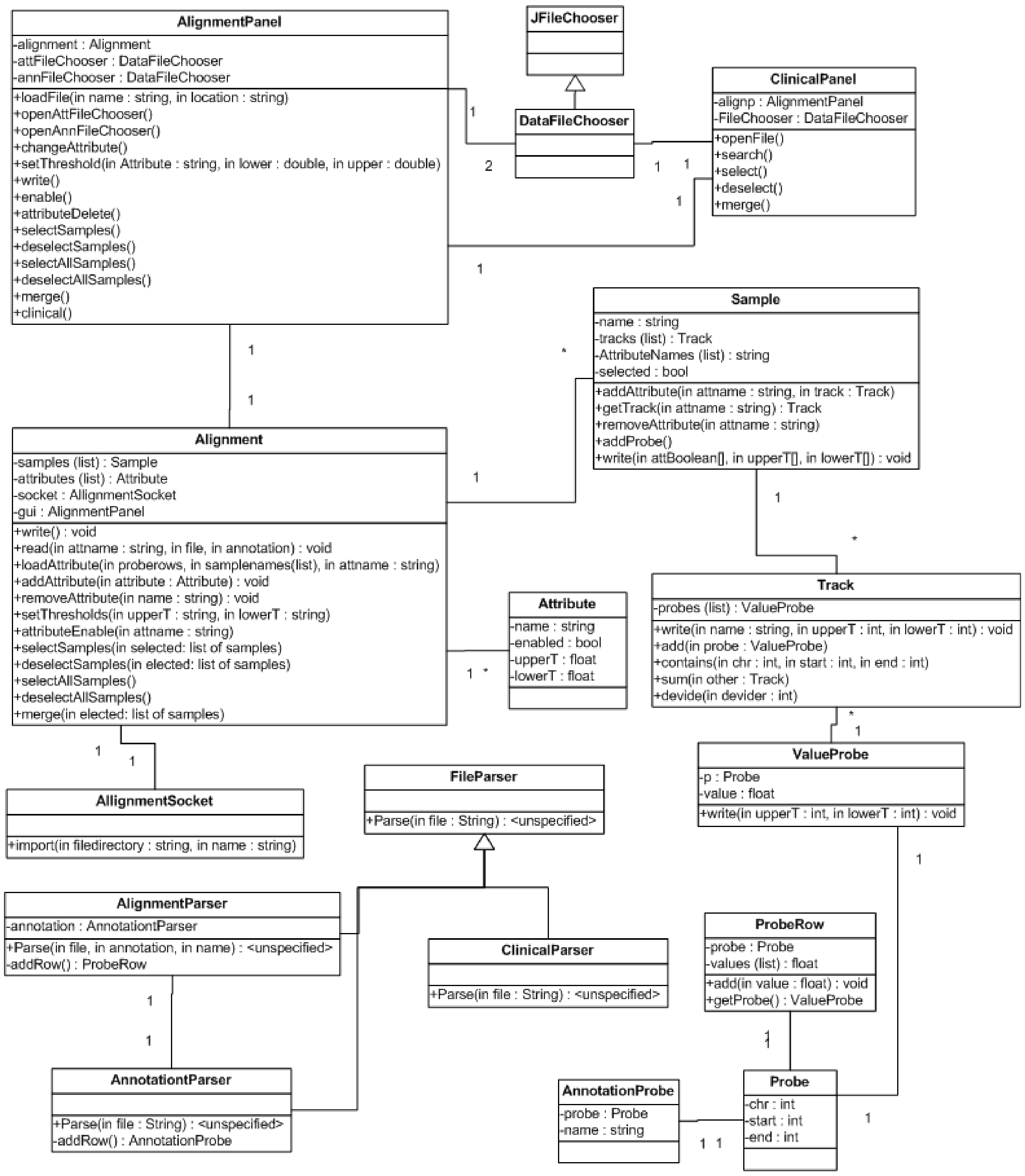

AlignmentPanel
-alignment : Alignment
-attFileChooser : DataFileChooser
-annFileChooser : DataFileChooser
+loadFile(in name : string, in location : string)
+openAttFileChooser()
+openAnnFileChooser()
+changeAttribute()
+setThreshold(in Attribute : string, in lower : double, in upper : double)
+write()
+enable()
+attributeDelete()
+selectSamples()
+deselectSamples()
+selectAllSamples()
+deselectAllSamples()
+merge()
+clinical()
JFileChooser
DataFileChooser
ClinicalPanel
-alignp : AlignmentPanel
-FileChooser : DataFileChooser
+openFile()
+search()
+select()
+deselect()
+merge()
Sample
-name : string
-tracks (list) : Track
-AttributeNames (list) : string
-selected : bool
+addAttribute(in attname : string, in track : Track)
+getTrack(in attname : string) : Track
+removeAttribute(in attname : string)
+addProbe()
+write(in attBoolean[], in upperT[], in lowerT[]) : void
Alignment
-samples (list) : Sample
-attributes (list) : Attribute
-socket : AllignmentSocket
-gui : AlignmentPanel
+write() : void
+read(in attname : string, in file, in annotation) : void
+loadAttribute(in proberows, in samplenames(list), in attname : string)
+addAttribute(in attribute : Attribute) : void
+removeAttribute(in name : string) : void
+setThresholds(in upperT : string, in lowerT : string)
+attributeEnable(in attname : string)
+selectSamples(in selected: list of samples)
+deselectSamples(in elected: list of samples)
+selectAllSamples()
+deselectAllSamples()
+merge(in elected: list of samples)
Attribute
-name : string
-enabled : bool
-upperT : float
-lowerT : float
Track
-probes (list) : ValueProbe
+write(in name : string, in upperT : int, in lowerT : int) : void
+add(in probe : ValueProbe)
+contains(in chr : int, in start : int, in end : int)
+sum(in other : Track)
+devide(in devider : int)
ValueProbe
-p : Probe
-value : float
+write(in upperT : int, in lowerT : int) : void
AllignmentSocket
+import(in filedirectory : string, in name : string)
FileParser
+Parse(in file : String) : &lt;unspecified&gt;
AlignmentParser
-annotation : AnnotationtParser
+Parse(in file, in annotation, in name) : &lt;unspecified&gt;
-addRow() : ProbeRow
ClinicalParser
+Parse(in file : String) : &lt;unspecified&gt;
ProbeRow
-probe : Probe
-values (list) : float
+add(in value : float) : void
+getProbe() : ValueProbe
AnnotationtParser
+Parse(in file : String) : &lt;unspecified&gt;
-addRow() : AnnotationProbe
AnnotationProbe
-probe : Probe
-name : string
Probe
-chr : int
-start : int
-end : int
1
2
1
1
1
1
1
1
*
1
1
*
1 *
*
1
1
1
1
1
1
1
1 1
1

*figure 16: UML of the alignment tool*

## Appendix: Pathway class diagram

In this appendix part is shown complete UML class diagram of this pathway part. Details explanation about the processes of this class diagram is elaborated in chapter 3.

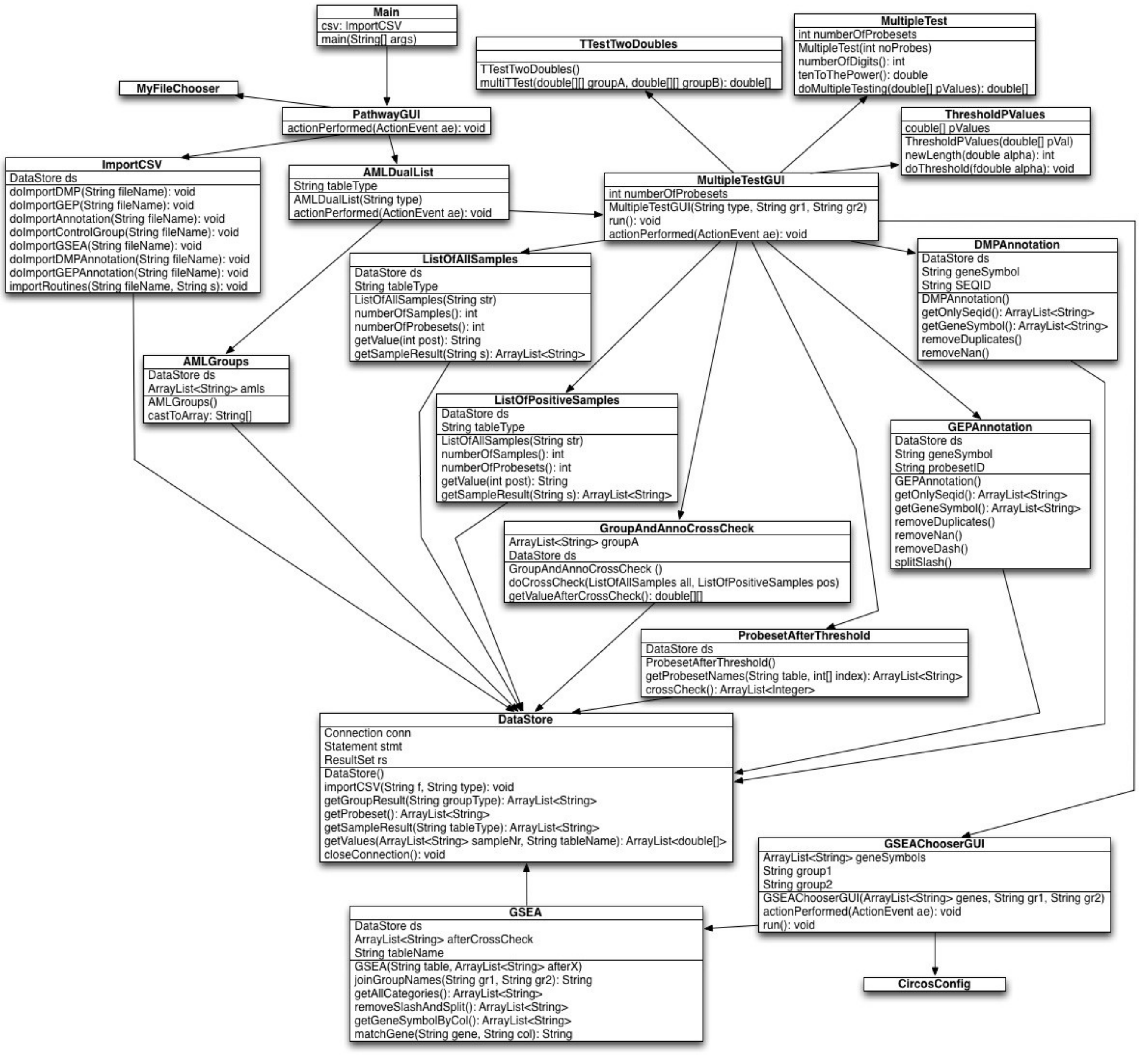

Main
csv: ImportCSV
main(String[] args)
MyFileChooser
PathwayGUI
actionPerformed(ActionEvent ae): void
TTestTwoDoubles
TTestTwoDoubles()
multiTTest(double[][] groupA, double[][] groupB): double[]
MultipleTest
int numberOfProbesets
MultipleTest(int noProbes)
numberOfDigits(): int
tenToThePower(): double
doMultipleTesting(double[] pValues): double[]
ThresholdPValues
couble[] pValues
ThresholdPValues(double[] pVal)
newLength(double alpha): int
doThreshold(fdouble alpha): void
ImportCSV
DataStore ds
doImportDMP(String fileName): void
doImportGEP(String fileName): void
doImportAnnotation(String fileName): void
doImportControlGroup(String fileName): void
doImportGSEA(String fileName): void
doImportDMPAnnotation(String fileName): void
doImportGEPAnnotation(String fileName): void
importRoutines(String fileName, String s): void
AMLDualList
String tableType
AMLDualList(String type)
actionPerformed(ActionEvent ae): void
MultipleTestGUI
int numberOfProbesets
MultipleTestGUI(String type, String gr1, String gr2)
run(): void
actionPerformed(ActionEvent ae): void
DMPAnnotation
DataStore ds
String geneSymbol
String SEQID
DMPAnnotation()
getOnlySeqid(): ArrayList&lt;String&gt;
getGeneSymbol(): ArrayList&lt;String&gt;
removeDuplicates()
removeNan()
ListOfAllSamples
DataStore ds
String tableType
ListOfAllSamples(String str)
numberOfSamples(): int
numberOfProbesets(): int
getValue(int post): String
getSampleResult(String s): ArrayList&lt;String&gt;
AMLGroups
DataStore ds
ArrayList&lt;String&gt; amls
AMLGroups()
castToArray: String[]
ListOfPositiveSamples
DataStore ds
String tableType
ListOfAllSamples(String str)
numberOfSamples(): int
numberOfProbesets(): int
getValue(int post): String
getSampleResult(String s): ArrayList&lt;String&gt;
GEPAnnotation
DataStore ds
String geneSymbol
String probesetID
GEPAnnotation()
getOnlySeqid(): ArrayList&lt;String&gt;
getGeneSymbol(): ArrayList&lt;String&gt;
removeDuplicates()
removeNan()
removeDash()
splitSlash()
GroupAndAnnoCrossCheck
ArrayList&lt;String&gt; groupA
DataStore ds
GroupAndAnnoCrossCheck ()
doCrossCheck(ListOfAllSamples all, ListOfPositiveSamples pos)
getValueAfterCrossCheck(): double[][]
ProbesetAfterThreshold
DataStore ds
ProbesetAfterThreshold()
getProbesetNames(String table, int[] index): ArrayList&lt;String&gt;
crossCheck(): ArrayList&lt;Integer&gt;
DataStore
Connection conn
Statement stmt
ResultSet rs
DataStore()
importCSV(String f, String type): void
getGroupResult(String groupType): ArrayList&lt;String&gt;
getProbeset(): ArrayList&lt;String&gt;
getSampleResult(String tableType): ArrayList&lt;String&gt;
getValues(ArrayList&lt;String&gt; sampleNr, String tableName): ArrayList&lt;double[]&gt;
closeConnection(): void
GSEAChooserGUI
ArrayList&lt;String&gt; geneSymbols
String group1
String group2
GSEAChooserGUI(ArrayList&lt;String&gt; genes, String gr1, String gr2)
actionPerformed(ActionEvent ae): void
run(): void
GSEA
DataStore ds
ArrayList&lt;String&gt; afterCrossCheck
String tableName
GSEA(String table, ArrayList&lt;String&gt; afterX)
joinGroupNames(String gr1, String gr2): String
getAllCategories(): ArrayList&lt;String&gt;
removeSlashAndSplit(): ArrayList&lt;String&gt;
getGeneSymbolByCol(): ArrayList&lt;String&gt;
matchGene(String gene, String col): String
CircosConfig

*Figure 17: UML Class Diagram pathway part*

## Appendix: Correlation Test Plan

This test plan contains several test cases which are manually performed by a tester on program execution.

| Test case | Description | Expected result |
|---|---|---|

| | | |
|---|---|---|
| Test case 1 | User clicks on choose file field | Open File dialog pops up |
| Test case 2 | User presses OK after choosing a file in Open File dialog | File name is written in the choose file field. |
| Test case 3 | User has selected both an annotation and data file for GEP or DMP | File starts loading and fills in the chromosomes in the chromosomes tab and selects the genes (if any) in the selected genes tab. |
| Test case 4 | User has selected a clinical data file | File starts loading and after finishing fills both the checkbox and list. |
| Test case 5 | User has pressed the correlate button on the chromosomes tab or intersection of all genes tab or selected genes tab | Correlation of data has started and after it has finished it will show an image if there are correlations with a certain threshold or an empty image if there aren't any. |
| Test case 6 | User has selected some properties in the clinical data section | Selections stay in place unless user changes. |

## Appendix: User manual

# User Manual

_HeatMapper Expansion_

# Table of Contents

# Introduction

Currently HeatMapper draws heatmaps and displays clinical data next to the heatmap, but  it is not clear enough for the researchers what each matrix element represents in human genome. Therefor this expansion to HeatMapper is made.

This expansion provides three manners of visualizing DNA data.

- Correlation

  This expansion is an environment in which the user can select genes and samples to correlate and consequently visualize the results in a circular diagram.

- Pathways

  This second expansion provides another view, also in a circular diagram, between two AML groups and their significantly related pathways, according to the the Molecular Signatures Database (MsigDB).

- Alignment

  The final expansion is a tool, that can load the data files and visualize them in Integrative Genomics Viewer(IGV). This is a genome browser, where the data is shown on horizontal tracks, aligned based on their location on the genome.

This software users manual describes how to use these three expansions to HeatMapper.

# Part 1. Input

This first part of the manual deals with the input files used in all three tools. The different input files used can be divided in three catagories:

- data files
- annotation files
- clinical files

However, they all have the following characteristics. They are comma seperated value tables (.csv) where each element in the table is seperated by a ";". Numeric elements should use a "." as a decimal seperator.

## Data file

Data files provide the values of sample and probe set pairs. The correlation and pathways tools use GEP and DMP data files. The alignment tool uses any data file that follows the format outlined here.

In a data file, the first row provides a list of sample names. Each row after that denotes a probe set. The first column provides the probe set id for each probe set and further columns provide the value that the probe set has in that corresponding sample. An example is given below.

*Probeset;sample1;sample2*

*probe1; 2.2; 1.2*

*probe2; 2.4; 0.8*

## Annotation file

An annotation file provides info about a specific probe set. Annotation files are always linked to a specific data file and most often be provided alongside one.

Just as in the data file, the rows represent probe sets and the first column provides the probe set ID. The columns however, represent properties of the probe set, such as the start and end location on the genome or the chromosome on which it is located.

*Probeset;start;end;chromosome*

*probe1; 50000; 50200; 1*

*probe2; 60000; 60200; 1*

These columns can be in any order, but they need to have specific names to be recognized.

The Alignment tool requires the chromosome number, start and end position of the probe set. It will search for columns containing "chr", "start" or "end" for the chromosome number. Alternatively, a single column named "Alignment" can be provided. This column must contain the chromosome number, start and end for each probe set in the following format.

*Chromosome:start-end*

For example: *2:50000-50200*

## Clinical file

The clinical file is used to provide info about specific sample patients. Each row represents a sample, with the first column giving the sample ID. The folowing rows provide info about the patient. For example:

*sample-id; gender; age*

*sample1;male;48*

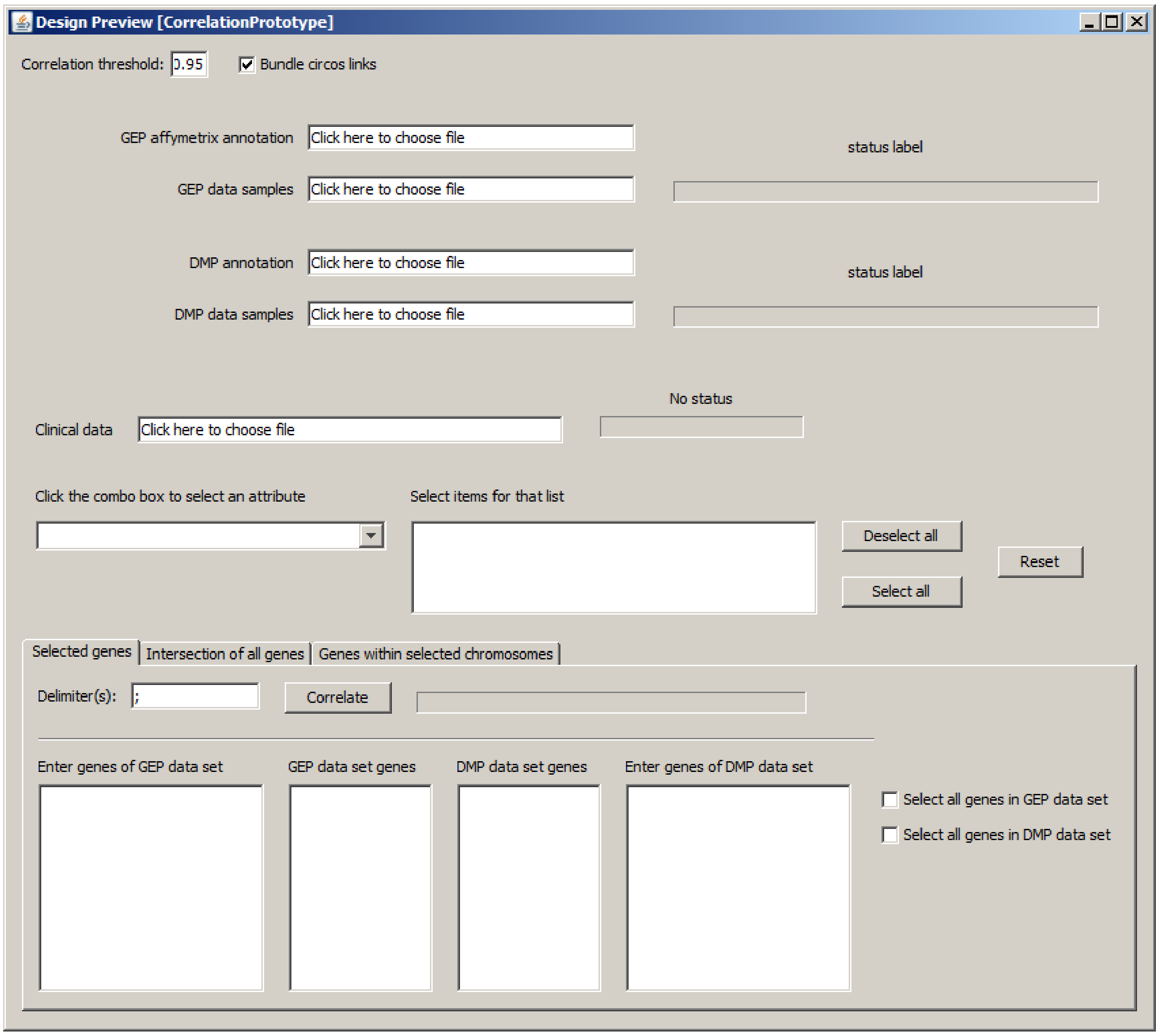
*sample2;female;36*

# Part 2. Correlation

## The user interface

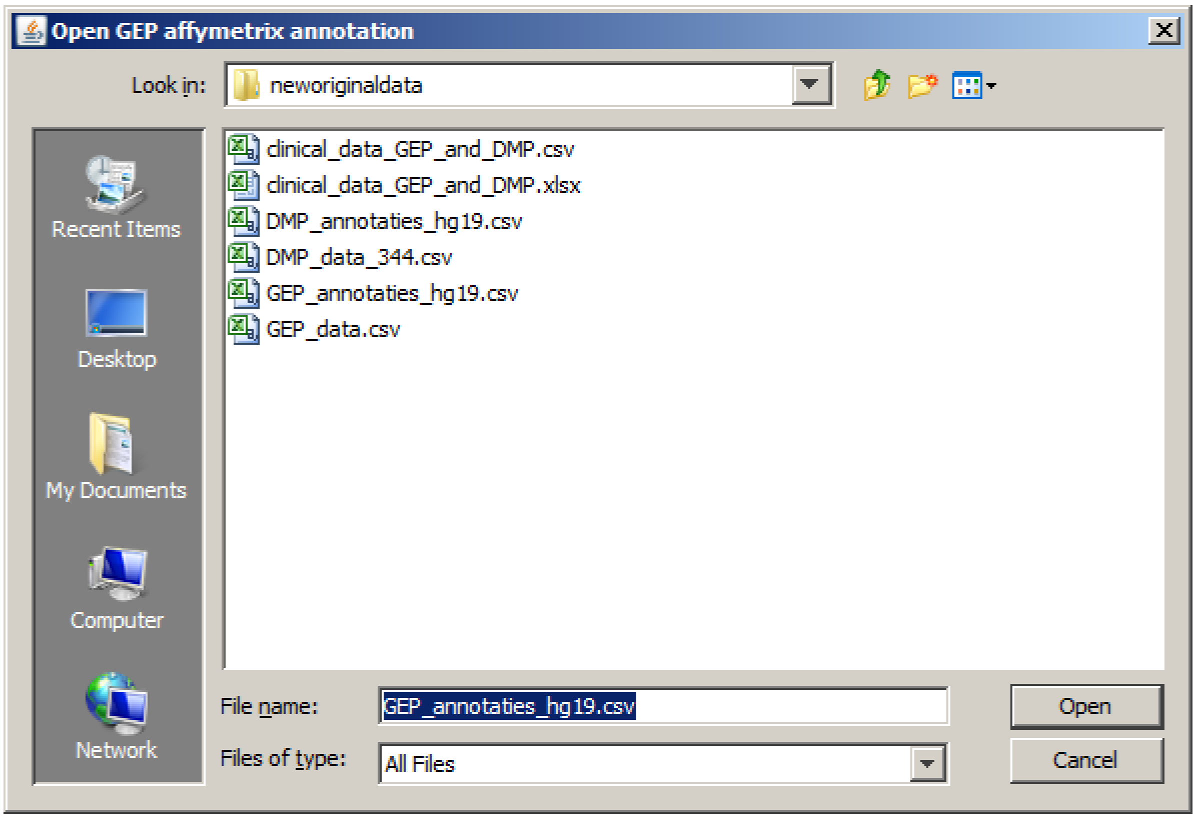
The user interface consists of 3 parts:

1. Loading the annotation and data files
2. Loading of the clinical data file
3. Selecting genes of interest and generate a correlation image.


Design Preview [CorrelationPrototype]
Correlation threshold: 0.95
Bundle circos links
GEP affymetrix annotation
Click here to choose file
GEP data samples
Click here to choose file
status label
DMP annotation
Click here to choose file
DMP data samples
Click here to choose file
status label
No status
Clinical data
Click here to choose file
Click the combo box to select an attribute
Select items for that list
Deselect all
Select all
Reset
Selected genes
Intersection of all genes
Genes within selected chromosomes
Delimiter(s): ;
Correlate
Enter genes of GEP data set
GEP data set genes
DMP data set genes
Enter genes of DMP data set
Select all genes in GEP data set
Select all genes in DMP data set

## File input

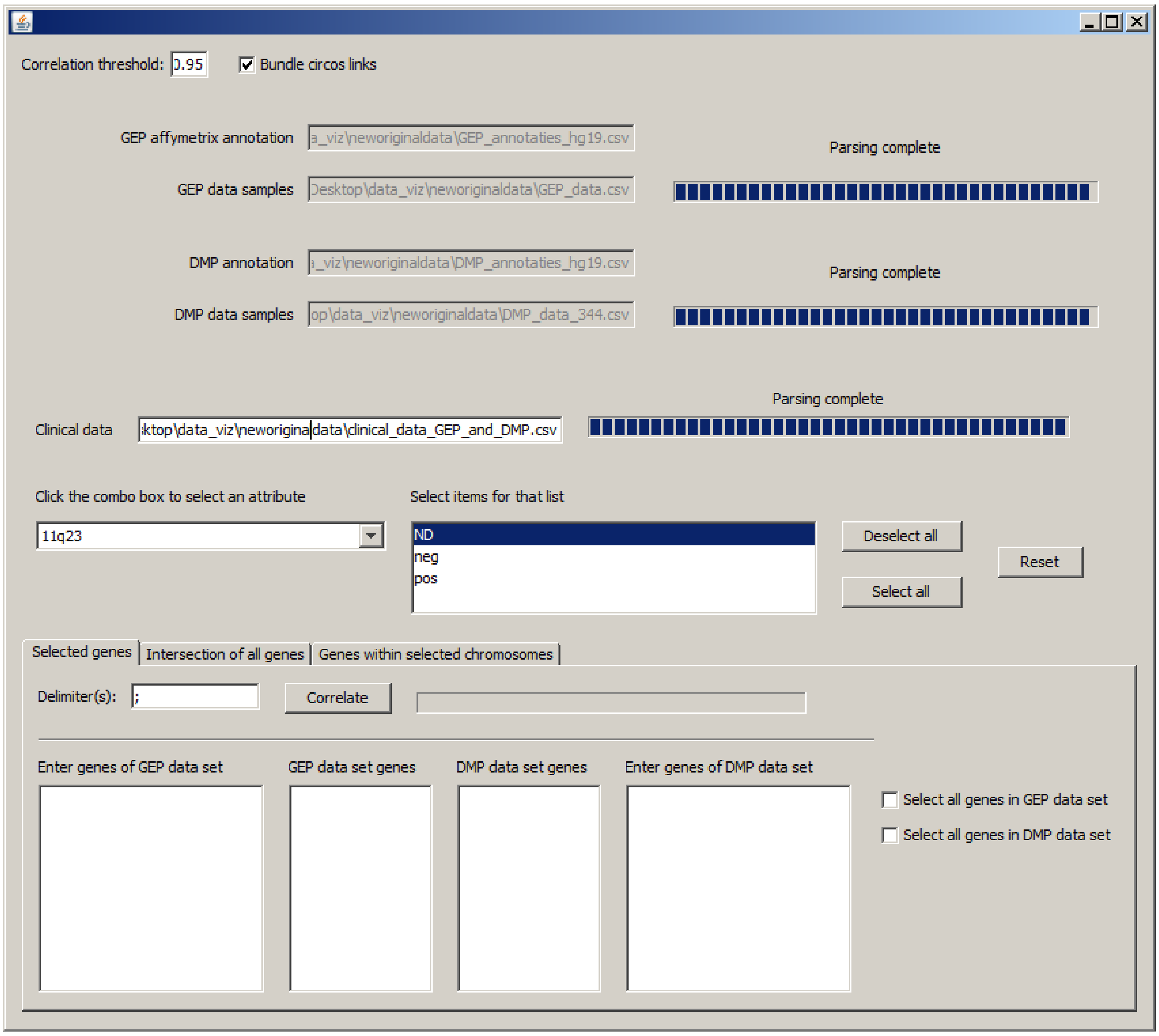
By clicking on the text fields it's possible to choose the respective input files by selecting them in a file dialog.



clinical_data_GEP_and_DMP.csv
clinical_data_GEP_and_DMP.xlsx
DMP_annotaties_hg19.csv
DMP_data_344.csv
GEP_annotaties_hg19.csv
GEP_data.csv

File name: GEP_annotaties_hg19.csv
Files of type: All Files

## Filtering on Clinical Data


Correlation threshold: 0.95 ☑ Bundle circos links

GEP affymetrix annotation: a_viz\neworiginaldata\GEP_annotaties_hg19.csv

GEP data samples: Desktop\data_viz\neworiginaldata\GEP_data.csv

Parsing complete

DMP annotation: a_viz\neworiginaldata\DMP_annotaties_hg19.csv

DMP data samples: op\data_viz\neworiginaldata\DMP_data_344.csv

Parsing complete

Clinical data: ktop\data_viz\neworiginaldata\clinical_data_GEP_and_DMP.csv

Parsing complete

Click the combo box to select an attribute: 11q23

Select items for that list: ND (selected), neg, pos

Delimiter(s): ;

Correlate
Enter genes of GEP data set
GEP data set genes
DMP data set genes
Enter genes of DMP data set
Select all genes in GEP data set
Select all genes in DMP data set

It is possible to choose an attribute and then select the items that you desire. Only the samples matching the selected criteria will be used in the calculations.

The “Deselect all” button deselects everything in the list of the currently selected attribute.

The “Select all” button selects everything in the list of the currently selected attribute.

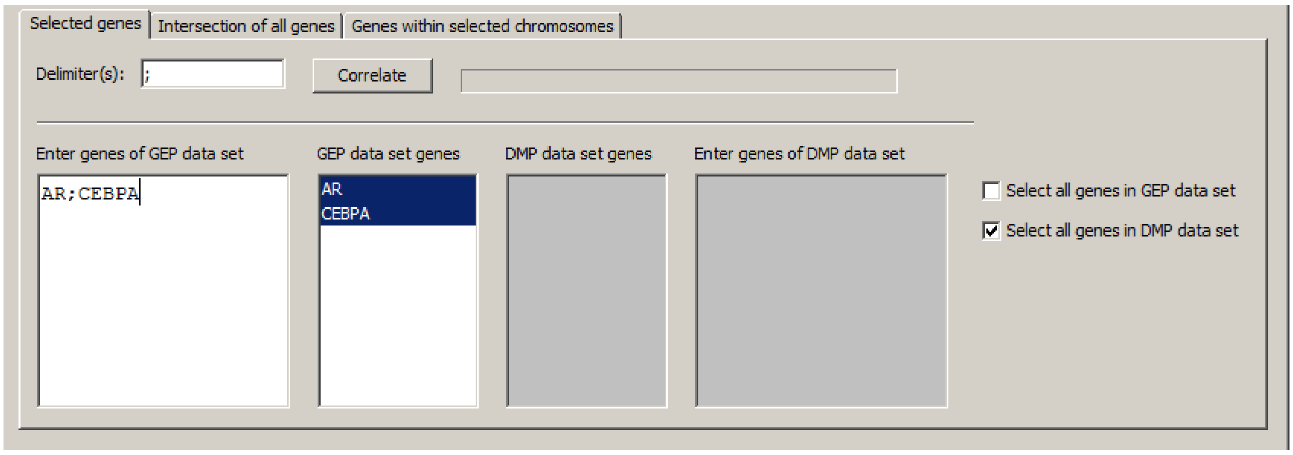
The “Reset” button deselects everything, if nothing is selected, all samples will be used.

## Correlation options

There are 3 ways to select genes of interest, each with a slightly different user-interface.

### Selected genes

There is a delimiter field, two text areas and two lists. The delimiters are applied to the input in the two text areas. The input is parsed real-time and parsed symbols will be displayed in the respective lists.



Selected genes
Intersection of all genes
Genes within selected chromosomes
Correlate
Enter genes of GEP data set
GEP data set genes
DMP data set genes
Enter genes of DMP data set

Delimiter(s): ;

AR;CEBPA

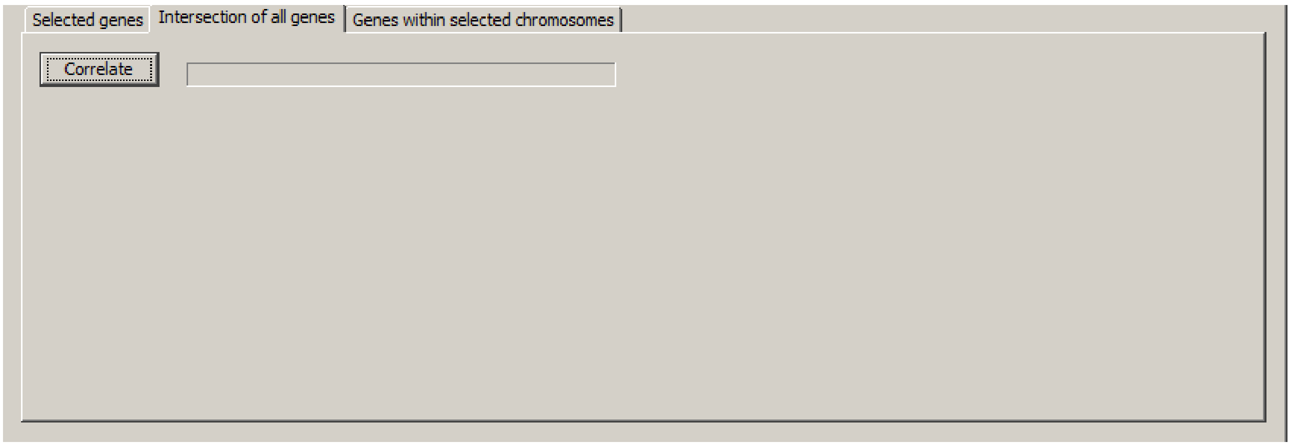
AR
CEBPA

[ ] Select all genes in GEP data set
[✓] Select all genes in DMP data set

The user can choose to put genes of interest in the text areas, and there's a quick checkbox for selecting all genes of either the GEP or DMP data set.

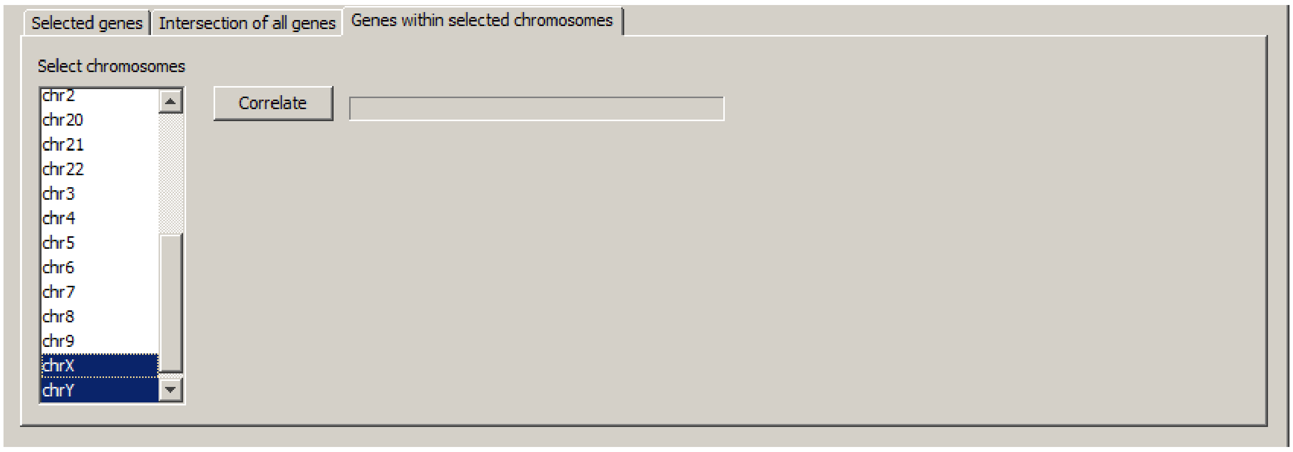
### Intersection of all genes

This mode only has a Correlate button, as it computes the intersection of gene-symbols of both the GEP data and the DMP data. Press the correlate button to start.



## Genes within selected chromosomes

After parsing has finished, chromosomes which were parsed will be displayed in the select chromosomes list. After 1 or more chromosomes have been selected, the correlate button can be pressed and progress will be displayed in the progress bar.



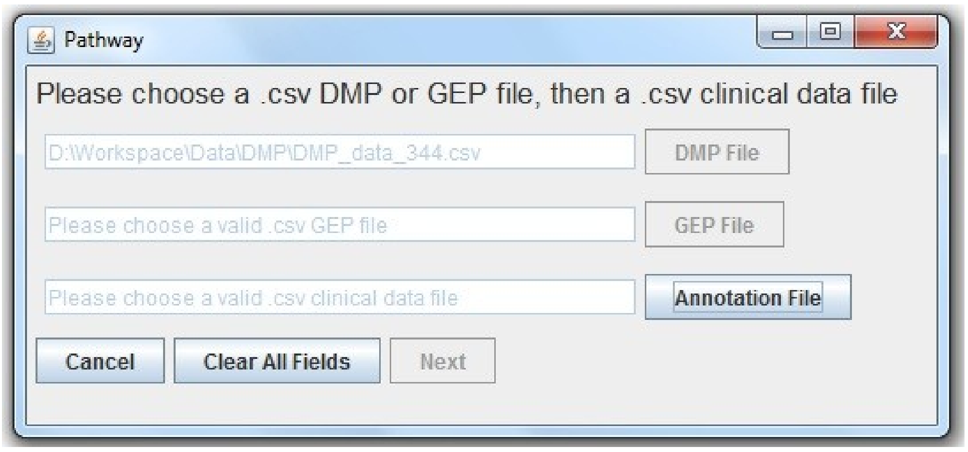
# Part 3. Pathways

This part 2 describes step by step how to operate this pathways expansion in order to obtain an end visualization in the form of a circular diagram, visualized using Circos.

## File Input

To enter this part, just click on the Pathways Button on HeatMapper initial screen.

This pathway home screen will then appear:



Please choose a .csv DMP or GEP file, then a .csv clinical data file

D:\Workspace\Data\DMP\DMP_data_344.csv

Please choose a valid .csv GEP file

Please choose a valid .csv clinical data file

*figure 1: Home screen of pathways part. In this example, a DMP .csv file is already selected.*

Functionalities of the buttons on this screen:

- First, please choose an input file of a gene expression or a methylation data. If it is a methylation data, click on “DMP file”-button, for a gene expression data, click on “GEP file”-button.
- After choosing a DMP or a GEP file, the “Annotation file”-button will become selectable.

  Please choose a clinical data file as input by clicking on the “Annotation file”-button.
- Clicking “Cancel”-button will return to HeatMapper initial screen.
- “Clear all fields”-button will reset chosen file(s).
- ”Next “-button will become selectable when an appropriate input file and a clinical data file are entered, pressing this button will go on further to the next screen.

Note: It might take a little while to load the .csv files completely.

## AML groups selection

After previous steps, a screen with two columns (left and right) will appear.

The left column contains a number of AML groups, which exactly two of these groups have to be selected. Groups that have been selected will  show up on the right column. Due to the limitation of two groups being selected, this program allows maximum two groups on the right column.

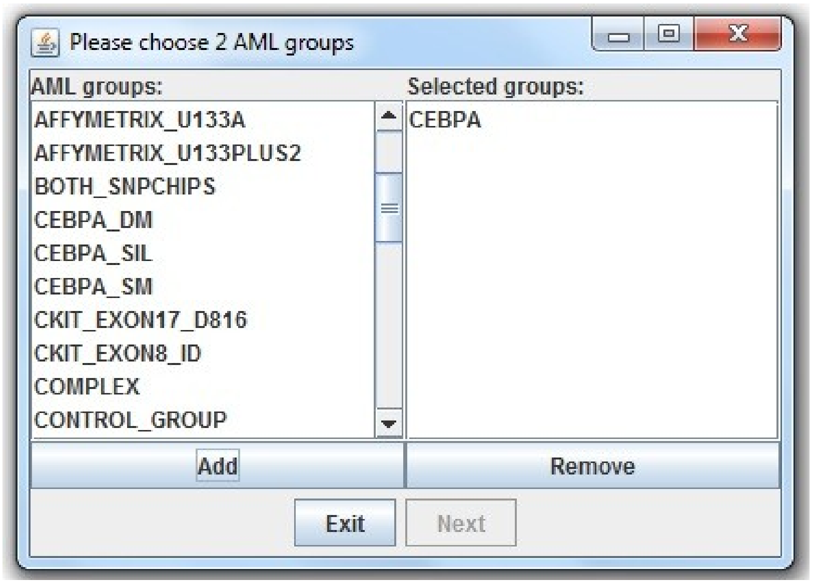


*figure 2: Choices of several AML groups. User has to choose 2 groups to compare.*

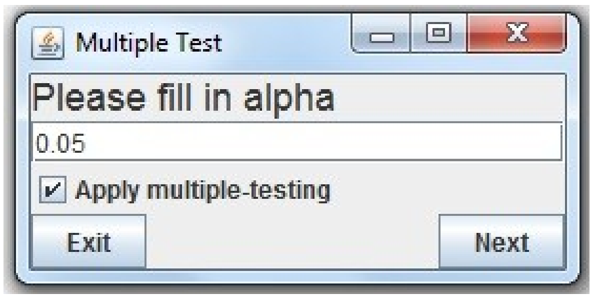
Functionalities of the buttons on this screen:

- Click on one group on the left column and click on  ”Add” –button to move it to the right column or to select it.
- Click on one group on the right column and click on “Remove”-button to return this group  to the left column.
- When two groups are added, the “Next”-button will become selectable. Pressing this button will go on further to the next screen.
- “Exit”-button will close the application.

Note: clinical data DMP and GEP will also be automatically loaded to determine which AML groups are available to compare as shown in figure 2. In this figure, one group is already added to the right column.

## Multiple Test (FWER)

On this screen, there is a field to threshold. User can fill in another value for this threshold. A standard threshold is set to 0.05. This number must use point to denote decimal, otherwise this application will not accept it. Below this field, there is a checkbox to decide whether to apply multiple test or not.



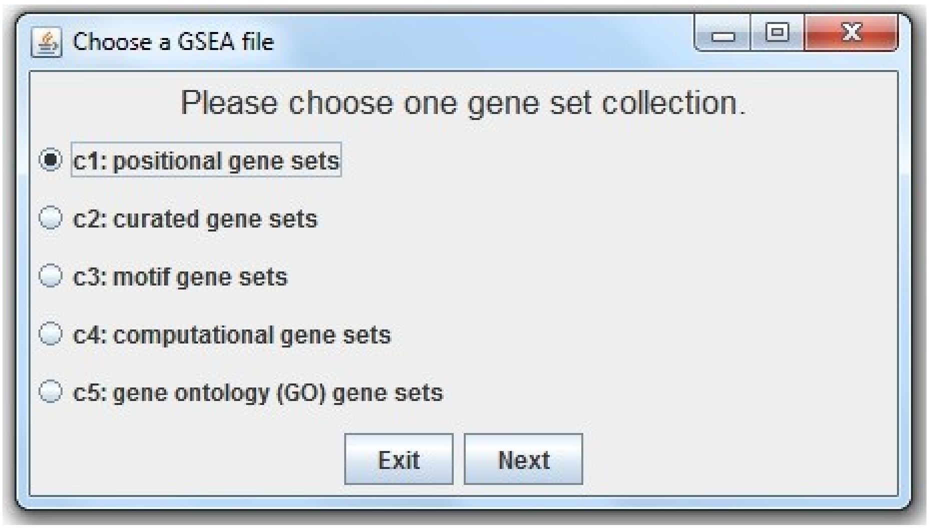
Please fill in alpha
0.05
[✓] Apply multiple-testing

*figure 3: Multiple test screen.*

Functionalities of the buttons on this screen:

- Clicking on “Exit”-button will exit the application.
- Pressing “Next”- button will go on further to the next screen.

## Gene Set Collections Chooser



Please choose one gene set collection.
● c1: positional gene sets
○ c2: curated gene sets
○ c3: motif gene sets
○ c4: computational gene sets
○ c5: gene ontology (GO) gene sets

*figure 4: Five gene set collections that one of them has to be chosen.*

There are totally five gene set collections that are one at a time selectable, namely:

- c1: positional gene sets.
- c2: curated gene sets.
- c3: motif gene sets.
- c4: computational gene sets.
- c5: gene ontology (GO) gene sets.

Please choose one of these five collections and press ”Next”-button to go on further with the process or “Exit”-button to quit the application.

Note: It might take a little while to complete this step.

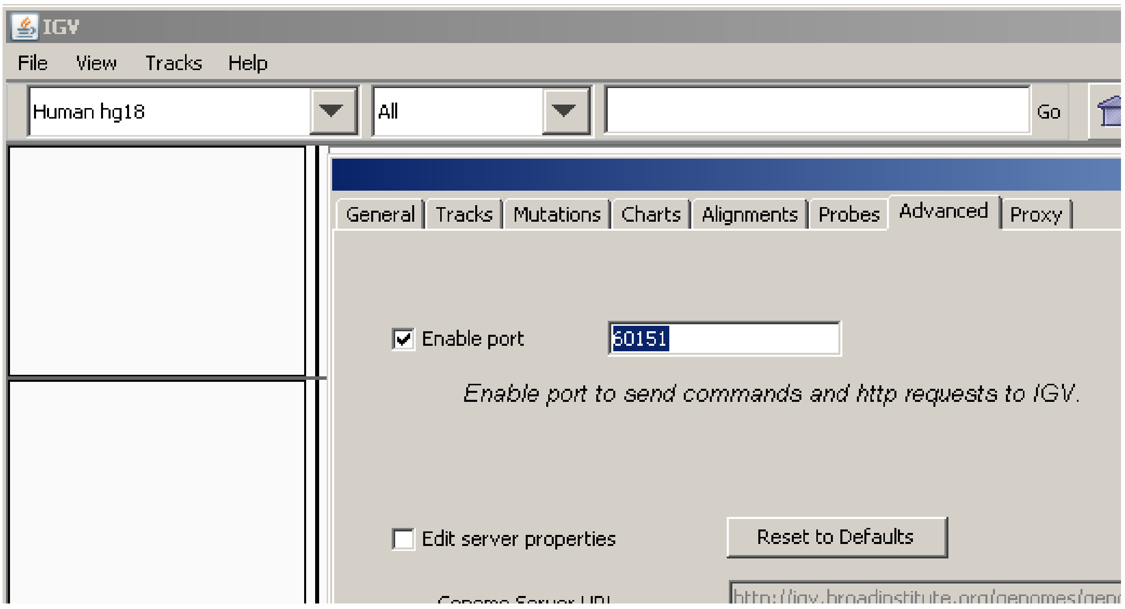
## Displaying Result Using Circos

Note: It might take a little while to complete this last step.

## Part 4. Alignment

This final part describes the functions of the alignment tool. The alignment tool is created to load genomic data into IGV from the format used in the other parts of this project. On top of that, the user can manipulate the data to influence what is visualized and how.

To ensure the correct functioning of the program, make sure IGV is started up before opening the tool. In IGV, open the preference window under view and go to the advanced tab as shown in figure 5. Ensure that port commands are enabled for port 60151. Also consider how much memory IGV will need for your intended goal. After that, run the alignment tool.



☑ Enable port 60151

Enable port to send commands and http requests to IGV.

☐ Edit server properties

*figure 5: Setting IGV to allow port commands.*

### Loading in attributes

The first two things you will see on the tools graphical user interface (seen on figure 6) are the info window, which is not displaying much info for now, and the “Add a new data type” section. Data types refer to different types of data files that can be loaded into the program, such as expression or methylation data. Any type of genomic data can be loaded into the program as long as a data file, containing the values for all the probe set and sample pairs, as well as an annotation file, containing the chromosome, start and end position for each probe set, are provided. These files must be provided in the same csv format as used in the other parts of this project. To program will store the data per sample. Each sample will have multiple tracks for each data type.

First, you should give the data type a name. This name will be used in IGV. After that, the “select data File” and “select annotation” buttons will allow you to browse to the files you want to import. You can also type in the path manually, but you will need to use double backslashes. After that, you can press load. The loading process can take a while, depending on the size of the input files. You can load multiple files at once, manipulate already loaded data types and use the clinical searches while you wait. Please note that the sample selection and merge functions may not work properly, until all data types are loaded.

Once the data types are loaded, the info window will display the total amount of samples loaded and selected as well as some basic info about each loaded data type.

An unfortunate property of the tool is that the program can be quite memory intensive. If you find that you have performance issues when loading in large files, it may be required to increase the heap size allocated to java. The most reliable way to do this is to create a new system variable called ”_JAVA_OPTIONS” with a body of “-Xmx500m” or -Xmx1g. This should set the java maximum heap space to 500 megabyte or 1 gigabyte respectively. Please ensure that you have enough RAM for this.

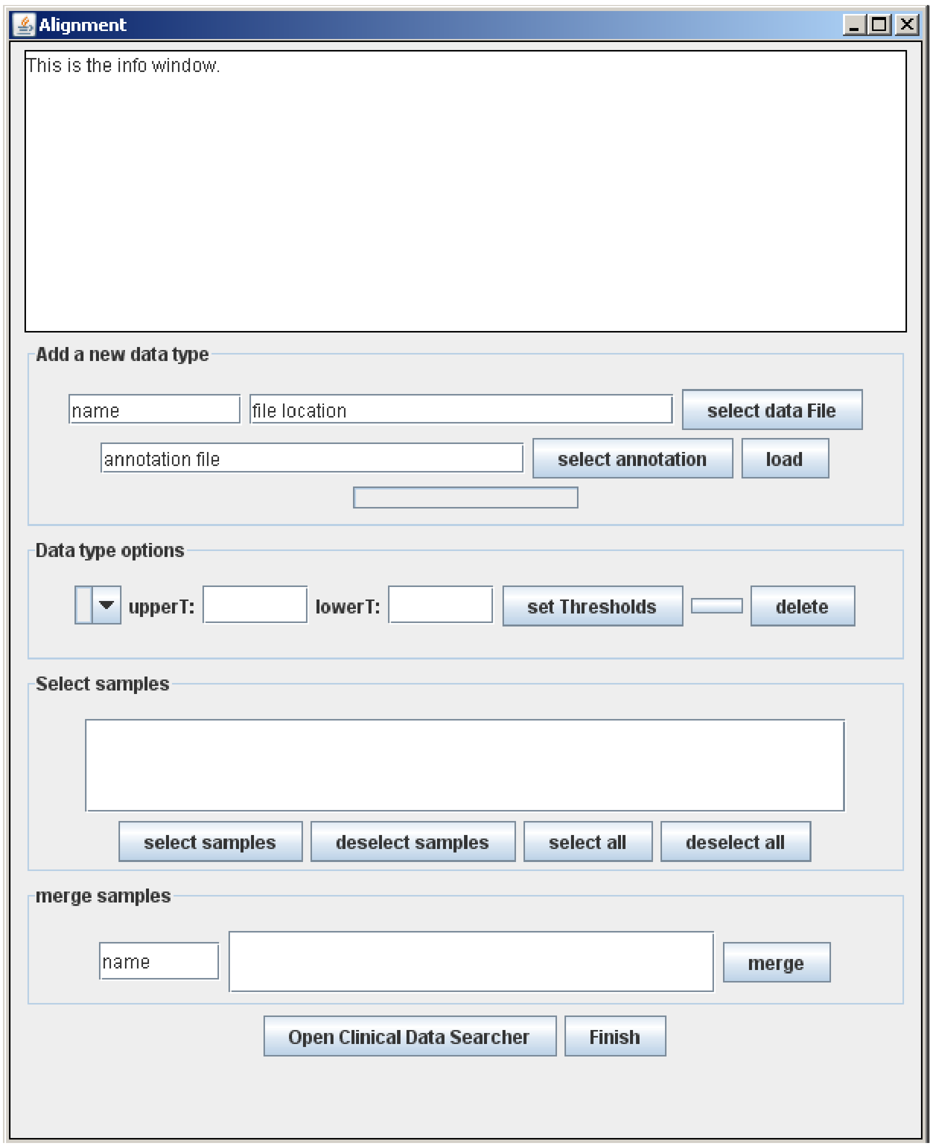


*figure 6: The main graphical user interface of the alignment tool*

## Data type options

In the “data type options” section, loaded data types can be selected via the dropdown box. You may then:

- set thresholds for the data type,
- enable or disable the data type or,
- delete the data type.

Thresholds determine what probe sets are shown or not shown when the data is visualized in IGV. Probe sets with values that fall outside the threshold range will be ignored. To set the thresholds, edit the upper and lower thresholds valued and click "set Thresholds". What if you only want to see outliers? If the upper thresholds is equal or lower then the lower Thresholds, the program will only display outliers and ignore all probe sets in between the two values. Note that the default of zero for both thresholds means that all probe sets are shown by default.

The next button in this section can be used to enable or disable the entire data type. Disabled data types are entirely ignored during visualization. If you permanently want to remove a data type, you can unload it with the delete button.

## Selecting samples

The next section on the main panel is allow you to select what samples should be shown in IGV. By default all samples are enabled when they are added. You can select or deselect all samples by using the select all and deselect all buttons. To select or deselect specific samples, you must write their names in the field seperated by semicolons.

Lets give an example, where you only want to display the samples 2281, 2282 and 2283. The easiest way to do this would be to first deselect all samples and then enter 2281;2282;2283 in the text field. Press select to select these samples. The info window will always give up to date info on how many samples are loaded and selected.

You can also select and deselect via the clinical searcher, which will be explained later in this manual.

## Merging samples

Merging samples works much the same as selecting samples. Merging takes the tracks from several selected samples and creates a new sample with the average of all those tracks. First you must give a name for the new sample. You then write the names of the samples you want to merge, seperated by semicolons, and press the merge button. If any of the selected samples have tracks of the same data type, the new merged sample will recieve a track of that data type where the values for all probe sets in that track are averaged. Merging samples can also be done via the clinical searcher.

## The clinical data searcher

The clinical data searcher, as shown in figure 7, is a tool to allow users to find samples based on a clinical data file. A clinical data file is a comma seperated value file where every row represents a sample. The first column is always the sample id. Aditional columns provide properties such as gender and age.

You can browse to such a file using the "choose clinical data file button". By giving a specific column to search in (such as "gender") as well as a desired value (such as "male") you can search for a group of samples that meet that search criteria.

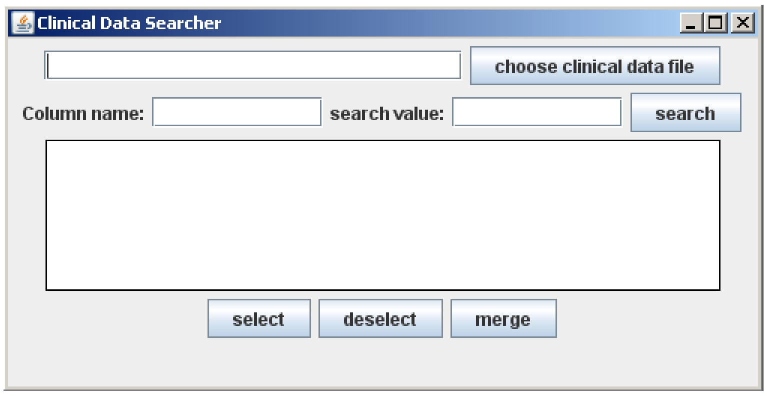


*figure 7: The clinical data searcher tool*

If you press search without giving a column name, you will recieve a list of all column names and if you press search without giving a search value, but only a column name, you will recieve a list of possible values present in that column.

After you have recieved a list of samples, you can directly select, deselect or merge these samples from the Clinical data search window.

## Loading into IGV

Once you are done selecting samples and data types, you can press finish. This will load all selected tracks into IGV. Tracks will be sorted on sample first and data type second. To load the files into IGV, the program will create output files called .bedGraph files. These files are stored into C:\AlignmentOutput. After loading is done, the result should look something like figure 8.

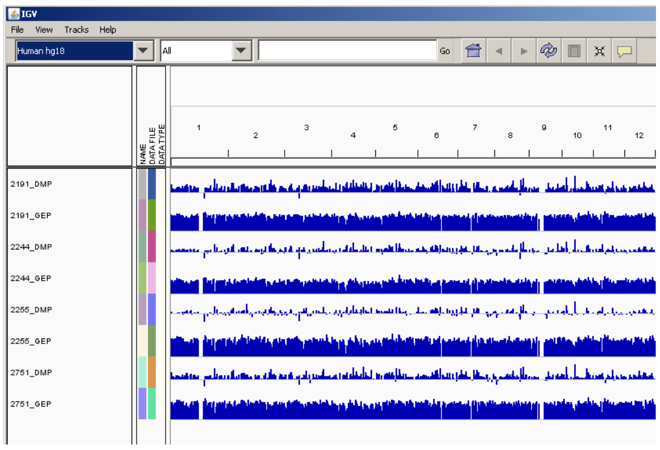


*figure 8: Example visualization in IGV*