

## Summer Studentship report

### 1. Executive summary

The aim of the project was to develop modules for a system called genepattern. Several bioinformatic tools for the analysis of microarray data needed to be converted into modules such that they could work on different platforms. To archive this it was necessary to modify and rewrite scripts in different programming languages. The languages were java, R, Perl and Matlab. Furthermore, for an unpublished framework with several different analysis tools a method was established to create standalone modules for components of the framework. Such framework components run standalone in genepattern and it is not required to install the whole framework. The overall purpose of this project was to increase the usefulness of a set of bioinformatic tools by designing an user friendly interface rather than using computer code.

### 2. Background/Introduction

Genepattern(gp) is a internet analysis platform which enables scientists to develop applications/modules to analyse data produced in biological sciences. Gp enables scientists to access more than 100 tools for gene expression analysis, proteomics, SNP analysis and common data processing tasks. This is only for the main server at the broad institute. There are several other server running which supply scientist with even more analysis tools.

Computational experience is not necessary for using gp. Furthermore it enables scientists to access a programatic interface to develop and distribute there own bioinformatic tools. Currently there are three programming environments supported by gp, Java, R, Perl, and Matlab. In gp there are links provided for executing computer code in this languages. Other languages can be installed through a straightforward mechanism. This makes it possible to use any other language for your gp modules. The results of the analysis are provided by links leading to downloadable files. For each module there is a documentation available. Input parameters, output and an overview what the module dos can be retrieved through this. The documentation is available through a help feature for all modules. The extandable nature of gp makes it possible to add all sorts of analysing tools. Several concepts of gp need to be explained. A brief overview is given in the table below.

Analysis module	This kind of module is used to preproces and/or analyse data from various sources.
Visualization module	Through this modules the user is able to visualise data in tools such as heatmaps, comparative marker selection and many more.
Suites	Suites group modules and pipelines into convenient packages such that related module which can be run in a chain are easily found.
Jobs	When a module is run, gp creates a job for this run of the module. This ensures reproducibility of the result by referring and comparing job results.
Pipelines	Modules often run in a chain such as the output of one module is the input of an preceding module. This can be set up as a pipeline to make sure the findings are reproducible.

Gp is free software and everybody can setup its own server. Very detailed instructions for all features are available at:

<http://www.broadinstitute.org/cancer/software/genepattern>

### 3.Goal

To setup an Genepattern server for testing and establishing methods to setup modules for several languages and systems.

### 4.Experimental results

#### 4.1.Instalation of Genepattern(gp)

At first download the server software from

<http://www.broadinstitute.org/cancer/software/genepattern>

Gp is available for windows or unix systems. The file comes in zip format. After the files are extracted from the zip archive follow the instructions on the screen. This is also very well documented in the genepattern documentation under

[http://www.broadinstitute.org/cancer/software/genepattern/download/gp\\_server\\_instruct.html](http://www.broadinstitute.org/cancer/software/genepattern/download/gp_server_instruct.html).

Furthermore some default settings need to be changed. Gp comes with a very old version of java and R. To point gp to the latest version of java a file called "StartGenePatternServer.lax" needs to be modified. Under

```
# LAX.NL.CURRENT.VM  
# -----  
# the VM to use for the next launch
```

```
lax.nl.current.vm=jre\\bin\\java.exe
```

the "lax.nl.current.vm" variable needs to be set to the path of the latest version of java. Another possibility to archive this, which is not in the documentation, is to copy manually the java files of the latest version into the jre folder in the genepattern directory.

To point gp to the latest version of R, go to "Administration">>Server settings>> Custom.

Add 2 new server settings. The first one is called "Rxxx\_HOME", were xxx stands for the version of R gp needs to point to and add the full path of that R version as content. The second variable is what is used in the command line. Use as name Rxxx. Again, xxx stands for the version of R. As content add

```
<java> -DR_suppress=<R.suppress.messages.file> -DR_HOME=<Rxxx_HOME>  
-Dr_flags=<r_flags> -cp <run_r_path>.
```

Additionally, user with administrator rights can be specified. By default every user is an administrator. That is not a good thing. Every user can change the settings in the server. The user groups are saved in an file "userGroups.xml" in the resource folder. The default version looks as followed.

```
<!-- map of users to groups -->  
  
<userGroups>  
  
<group name="administrators">  
  <user name="*" />  
</group>
```

```
</userGroups>
```

Add user with administrator rights as followed.

```
<!-- map of users to groups -->
```

```
<userGroups>
```

```
<group name="administrators">  
  <user name="Christoph Knapp"/>  
  <user name="Cris Print"/>  
</group>
```

```
</userGroups>
```

This is also very well explained in the gp documentation.

#### **4.2.Setting up testcases.**

To become familiar with the gp platform several testcases for all supported languages were set up. This testcases are very simple. Several strings as input were printed to an standard output. Gp catches standard output produced by an module and stores it in a file "strout.txt". For Matlab we had two choices, to use compiled Matlab scripts or uncompiled. For compiled Matlab code it would have been necessary to purchase a compiler license. Thats why we decided to use uncompiled code. For uncompiled code the use of a java wrapper script was necessary. This wrapperscript is available under [http://www.broadinstitute.org/cancer/software/genepattern/tutorial/gp\\_programmer.html](http://www.broadinstitute.org/cancer/software/genepattern/tutorial/gp_programmer.html).

#### **4.3.Establish a method for conversion of framework methods in to gp modules**

The next task involved an unpublished framework of several analysis tools. This framework combines this analysis tools into one setup were all methods are available to use. The framwork is written in Matlab. To convert a single method into one module it was necessary to extract all files needed. Another Matlab wrapper script was used to start the framwork component. After that the same java wrapper script, mentioned above, was used to start the framwork wrapperscript. For some feature like exporting and cloning the module gp showed behaviour which was not compatible with the setup of the framwork component. When this feature was used subdirectories were not copied over into the generated directory or archive. This problem was solved by supplying the gp module code with an zip archive containing all subdirectories necessary. Inside the Matlab wrapper script an statement checks if the those directories are not there. If not the suplied zip archive becomes unzipped. Pseudo code for that is suplied below.

```
%in the Matlab wrapper script
```

```
if(subdirectories do not exist)  
  unpack zip archive
```

#### **4.4.Conversion of R analysis tools in to gp modules**

Several R scripts were converted in to gp modules to make them easily available for other scientists. This tools involved basic microarray analysis and tools to extract meaningful results after

microarray analysis is finished.

#### 4.4.1. Microarray analysis modules.

- 1) Microarray images and RMA conversion  
A gp module which takes an zip archive containing the cel files of an Affymetrix microarray experiment. The module produces images of all chips for quality control, performs the rma conversion of the data and stores this into an dataframe file, as a normalized log2-transformed data matrix. Furthermore several plots are generated. Violinplots of the chip data and the whole data and boxplots for comparison of the raw data versus after the RMA conversion.
- 2) Create Contrasts  
For the LIMMA analysis in R a designmatrix is necessary to identify which of the samples are going to be compared. This module offers an easy way to generate such a matrix. It is saved in a Rdata file which can be easily loaded into R using the load function.
- 3) Cluster columns of dataframe  
The data is clustered based on the columns of an dataframe and an cluster object is produce. Furthermore a dendogram of this clustering is generated. The clusterobject serves as input for the preceding module.
- 4) Produce colored dendogram  
To ensure that there is nothing wrong with either sample or controls a colored dendogram is generated which should cluster samples and controls into different subtrees.
- 5) Statistical testing of simple control vs experiment  
This module fits an linear model to the data and fits the designed contrasts to the data. According to that the data is filtered and sorted after increasing p-values and only the statisticly significant genes are kept in the data. For further decissions heat maps are generated. From this heat maps further selection of relevant genes can be made.
- 6) Gather query  
The online gather tool plays an important role for finding relevant coherence between genes. This module takes an dataframe as input were the row names are genenames and performs an gather query on them. The data should not have more than 800 rows.
- 7) Sam analysis  
SAM, significance analysis of microarray data can be performed with this module. It takes the Rdata file from the “Statistical testing of simple control vs experiment” called “designAndEsetSelxxx.Rdata” and performs SAM on this data. Were xxx stands for the p-value treshold used for filtering the significant genes. A plot is generated to visualise how many of the genes selected would be likely to represent false discoveries.
- 8) Density plots of p-values for permutated microarray data  
This module is for testing if the findings from the previous modules are correct. A number of random permutations of the contrasts is generated. A function was introduced which produces a great number of unique permutations of contrasts. In fact for most contrast setups it would generate all unique permutations. From this pool of permutated contrasts the user picks a number and a linear contrast is fitted for all permutations picked. If there are less unique permutations only the unique ones are used. For all of them a density plot is generated and plotted in one single graph. The original data is than plotted in red. If the findings are significant no other density plot has a higher peak than the original data.

#### 4.4.2. Processing tools

Three other modules were generated which work with different input after microarraanalysis is finished.

- 1) Find modulation multicore limited cor genes of gene to gene relationship by a third gene  
To estimate the relationship of three genes, this module was generated. It calculates the correlation between two genes. If there is a correlation it tries to find a third gene which influences this correlation. Such as when the third gene is up or down regulated is there a change in correlation for the other two genes. This module requires a big amount of computer power. It is designed to use the resources(multiple cores) on the computer very efficient.
- 2) Hub child analysis 1  
This modul takes an edge list stored in a file. An edgelist is an representation of a network were each line has two variable. This represent a connection or edge in the network. For each gene the number of children or neighbors is visualised in a histogram. The data is converted into dataframe for further use in the next module.
- 3) Hub Child analysis 2  
The input for this module is the dataframe from “Hub child analysis 1” module, an treshold value of the lowest number of children for an parent and the false discovery rate (FDR) treshold for filtering the gather query. The value for the minimum number of children is important to pick the parents with highest children because those promise the best results in a gather query to lead to insight of the relationships. The FDR.treshold is important to have only those results that are most likely true discoveries.
- 4) Hub Child analysis pipline  
To test the concept of a gp pipline from the Hub child analysis1 and 2 module a pipline was created.

#### 5. Conclusions/Prospects/Significance

Genepattern is a very useful tool for scientists to supply them with a user friendly graphical interface for analysis of their data. To have the possibility of using it makes the workflow in a workgroup more effectiv. Scientists do not need to worry about implementation or codewriting their analysis which is a significant source of error. Instead, they can focus on experimental design and correctnes of there analysis. Furthermore gp provides a very good way of reproducing findings. The same data can be analysed by several people making sure that the results are not changed. Creating modules contributes effectiv to decreasing human error in analysis of data. Almost everything that is now done by scientists using a programing language such as java, perl, R or Matlab can be converted into an gp module. This makes distribtion of the tools easy and helps to decrease human error when this tools are used by scientists who are not that experienced in using bioinformatic tools without supervision.