

## DNA Microarray Data Analysis

IIRIS HOVATTA, KATJA KIMPPA, ANTTI LEHMUSSOLA, TOMI PASANEN,  
JANNA SAARELA, ILANA SAARIKKO, JUHA SAHARINEN, PEKKA TIIKKAINEN  
TEEMU TOIVANEN, MARTTI TOLVANEN, MAUNO VIHINEN AND GARRY WONG  
EDITORS JARNO TUIMALA AND M. MINNA LAINE

CSC

# **DNA Microarray Data Analysis**

Editors

Jarno Tuimala

M. Minna Laine

CSC, the Finnish IT center for Science

CSC – Scientific Computing Ltd. is a non-profit organization for high-performance computing and networking in Finland. CSC is owned by the Ministry of Education. CSC runs a national large-scale facility for computational science and engineering and supports the university and research community. CSC is also responsible for the operations of the Finnish University and Research Network (FUNET).

All rights reserved. The PDF version of this book or parts of it can be used in Finnish universities as course material, provided that this copyright notice is included. However, this publication may not be sold or included as part of other publications without permission of the publisher.

© The authors and  
CSC – Scientific Computing Ltd.  
2005

Second edition

ISBN 952-5520-11-0 (print)

ISBN 952-5520-12-9 (PDF)

<http://www.csc.fi/oppaat/siru/>

<http://www.csc.fi/molbio/arraybook/>

Printed at  
Picaset Oy  
Helsinki 2005

## Preface

This is the second, revised and slightly expanded edition of the DNA microarray data analysis guidebook. As a change to the previous edition, some relatively quickly changing material such as software tutorials have been exclusively published on the book's web site. Please see <http://www.csc.fi/molbio/arraybook/> for more information and to access the extra material.

DNA microarrays generate large amounts of numerical data, which should be analyzed effectively. In this book, we hope to offer a broad view of basic theory and techniques behind the DNA microarray data analysis. Our aim was not to be comprehensive, but rather to cover the basics, which are unlikely to change much over years. Especially, we hope that researchers starting their data analysis can benefit from the book.

The text emphasizes gene expression analysis. Topics, such as genotyping, are discussed shortly. This book does not cover the wet-lab practises, such as sample preparation or hybridization. Rather, we start when the microarrays have been scanned, and the resulting images are being analyzed. Also, we take the files with signal intensities, which usually generate questions such as: "How is the data normalized?" or "How do I identify the genes which are upregulated?", and provide some simple solutions to these specific questions and many others.

Each chapter has a section on suggested reading, which introduces some of the relevant literature. Some chapters have additional information available on the web. In the first edition the software examples were included in the book, but we have now moved them into Internet. This allows us to better keep the material up to date.

Juha Haataja and Leena Jukka are warmly acknowledged for their support during the production of this book.

We are very interested in receiving feedback about this publication. Especially, if you feel that some essential technique has been missed, let us know. Please send your comments to the e-mail address [Jarno.Tuimala@csc.fi](mailto:Jarno.Tuimala@csc.fi).

Espoo, 23rd December 2005

*The authors*

## List of Contributors

Iiris Hovatta  
National Public Health Institute  
Haartmaninkatu 8  
FI-00290 Helsinki  
Finland

Katja Kimppa  
PerkinElmer Life Sciences and Analytical Sciences  
- Wallac Oy  
P.O.Box 10  
FI-20101 Turku  
Finland

M. Minna Laine  
CSC, the Finnish IT center for science  
Keilaranta 14  
FI-02101 Espoo  
Finland

Antti Lehmussola  
Tampere University of Technology  
P.O.Box 553  
FI-33101 Tampere  
Finland

Tomi Pasanen  
University of Helsinki  
P.O.Box 68  
FI-00014 University of Helsinki  
Finland

Janna Saarela  
Biomedicum Biochip Center  
Haartmaninkatu 8  
FI-00290 Helsinki  
Finland

Ilana Saarikko  
University of Helsinki  
P.O.Box 68  
FI-00014 University of Helsinki  
Finland

Juha Saharinen  
National Public Health Institute  
Haartmaninkatu 8  
FI-00290 Helsinki  
Finland

Pekka Tiikkainen  
VTT  
P.O.Box 106  
FI-20521 Turku  
Finland

Teemu Toivanen  
Centre for Biotechnology  
Tykistökatu 6  
FI-20521 Turku  
Finland

Martti Tolvanen  
Institute of Medical Technology  
Biokatu 8  
FI-33520 Tampere  
Finland

Jarno Tuimala  
CSC, the Finnish IT center for science  
Keilaranta 14  
FI-02101 Espoo  
Finland

Mauno Vihinen  
Institute of Medical Technology  
Biokatu 8  
FI-33520 Tampere  
Finland

Garry Wong  
A. I. Virtanen -institute  
University of Kuopio  
FI-70211 Kuopio  
Finland

# Contents

<b>Preface</b>	<b>4</b>
<b>List of Contributors</b>	<b>5</b>
<b>I WEB extra: Introduction</b>	<b>8</b>
<b>1 Web extra: Image analysis</b>	<b>9</b>
1.1 Image analysis using commercial software . . . . .	9
1.1.1 ScanArrayExpress . . . . .	9
1.1.2 Agilent Image analysis . . . . .	11
1.2 Freeware image analysis software . . . . .	12
1.2.1 TIGR Spotfinder . . . . .	13
1.2.2 ScanAlyze 2 . . . . .	14
1.2.3 GridGrinder . . . . .	16
<b>2 Web extra: Basic statistics</b>	<b>19</b>
2.1 Statistics using GeneSpring . . . . .	19
2.1.1 Simple statistics . . . . .	19
2.1.2 Transformations . . . . .	19
2.1.3 Scatter plot and histogram . . . . .	19
2.1.4 Correlation . . . . .	20
2.1.5 Linear regression . . . . .	20
2.1.6 One-sample t-test . . . . .	21
2.1.7 Independent samples t-test and ANOVA . . . . .	21
<b>II WEB extra: Analysis</b>	<b>23</b>
<b>3 Web extra: Preprocessing of data</b>	<b>24</b>
3.1 Examples using GeneSpring . . . . .	24
3.1.1 Importing data . . . . .	24
3.1.2 Background subtraction . . . . .	24
3.1.3 Calculation of expression change . . . . .	24
3.1.4 Replicates . . . . .	25
3.1.5 Checking linearity . . . . .	25

---

3.1.6	Normality . . . . .	26
3.1.7	Filtering . . . . .	26
<b>4</b>	<b>Web extra: Normalization</b>	<b>27</b>
4.1	Using GeneSpring for normalization . . . . .	27
<b>5</b>	<b>Web extra: Cluster analysis of microarray information</b>	<b>29</b>
5.1	GeneSpring and clustering . . . . .	29
5.1.1	Clustering tool . . . . .	29
5.1.2	Principal components analysis tool . . . . .	31
5.1.3	Predict parameter value tool . . . . .	31
<b>III</b>	<b>WEB extra: Data mining</b>	<b>33</b>
<b>6</b>	<b>Web extra: Data mining for promoter sequences</b>	<b>34</b>
6.1	Using BioMart to retrieve promoter regions . . . . .	34
6.1.1	Final warnings regarding upstream data . . . . .	37
6.2	GeneSpring and promoter analysis . . . . .	39
<b>7</b>	<b>Web extra: Biological sequence annotations</b>	<b>40</b>
7.1	Using Ensembl server to batch process of annotations . . . . .	40
7.1.1	Example 1 . . . . .	40
7.1.2	Example 2 . . . . .	41
<b>8</b>	<b>Web extra: Software issues</b>	<b>42</b>
8.1	Programming . . . . .	42
8.1.1	Perl . . . . .	42
8.1.2	Awk . . . . .	43
8.1.3	R . . . . .	43
8.2	Common code examples . . . . .	44
8.2.1	Read tabular data . . . . .	44
8.2.2	Filtering/Rows . . . . .	47
	Filtering/Columns . . . . .	48
8.2.3	File conversions/print/write . . . . .	48
	<b>Index</b>	<b>50</b>

## **Part I**

### **WEB extra: Introduction**

# 1 Web extra: Image analysis

Antti Lehmussola, Katja Kimppa and Pekka Tiikkainen

## 1.1 Image analysis using commercial software

### 1.1.1 ScanArrayExpress

ScanArrayExpress(SAE) software comes with the PE scanner, and is available in certain centers in Finland. SAE is scanning and image analysis software.

Version tested: 2.1

Environment tested: XP, 1024 Mb memory, 2 Ghz Pentium 4

Image analysis starts by opening images, one for each channel. If the images are opened simultaneously, the reference image assigning has to be checked (File -> Set control image) to ensure the calculations of ratios to be done correctly.

Quantitation of the image starts by selecting an existing quantitation protocol or using Easy Quant (define parameters for one analysis only). New quantitation protocols can be done through Configure -> Quantitation Protocols -> Add. When using an existing quantitation protocol, select the quantitation protocol and fine tune the locations of subarrays and images before starting the quantitation. When using the Easy Quant, user must define the grid either by inputting the needed values or by using a Gal-formatted file, and place the grid correctly on top of the images. The spot quantitation method and normalization method must be chosen also.

In SAE, four different spot quantification methods are implemented, fixed circle, adaptive circle, adaptive threshold and histogram methods.

- Adaptive circle: The program tries to find the edges of a spot and draws a circle around the spot. What inside the circle is the spot, and outside is the background.
- Fixed circle: The program draws a specified size circle (specified in the template) around the spot, and what's inside the circle is the spot, and outside is the background.
- Adaptive threshold: This method uses statistical algorithm to define the spot.
- Histogram: In this method the histogram of spots is calculated and used to define the spot and the background.

Status	Flag
Found	0
Not Found	1
Absent (call from GAL file)	2
Present	3
Bad (only by user interference)	4

In SAE only local background is calculated.

In SAE two different normalization methods are implemented, linear normalization method and LOWESS normalization. These are applied to the whole dataset.

SAE uses 5 different classes to classify the spots, these classes correspond in the result file Flag column values as follows:

SAE gives out 49 columns for 2 channel data, including spot locations, gene annotations, raw information of the spots in individual channels, statistical information about the spots, calculated ratios and normalized data. SAE calculates both mean and median values for spots, backgrounds and ratios.

SAE has got some tools for preliminary checkup of the data, including few scatterplot tools f.ex. MA-plots or intensity-intensity plots drawn from raw or normalized data, and distribution plots.

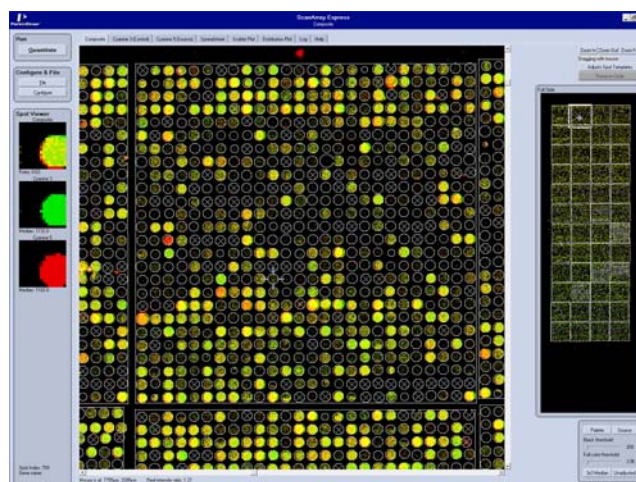
After quantification, SAE shows the data as a spreadsheet and the grids located over the images. In this step user has got the possibility to modify the automated results, moving spots or changing the status of the spot. The selected spot is shown selected in each of the different views to the data.

#### Pros

- Easy to use
- Versatile possibility to adjust grids and spots
- Interactivity after the automated analysis
- Possibility to change f.ex spot locations after analysis
- Multiple options for spot analysis method

#### Cons

- Only local background is calculated
- If the hybridization is bad, the automated spot finding may drift from correct location
- For array size of 30 000 spots, the analysis of one array takes 30-45 minutes
- The response to the commands in adjustment of grid locations is slow



**Figure 1.1:** Screenshot of SAE after spot finding. Note that the data selection is transparent through all of the windows (composite, cyanine3, cyanine5...) simultaneously. Three images of the array are also available at all times, the main analysis image (in the middle), full image of the array (on the right) and a close up for a spot (on the left).

### 1.1.2 Agilent Image analysis

Agilent Image Analysis (AIA) software is distributed with Agilent microarray scanners, and is also available for 30 day test use through the Agilent web-pages. Image Analysis software is available in centers which have got Agilent microarray scanner.

Version tested: A.7.5.1

Environment tested: XP, 1024 Mb memory, 2 Ghz Pentium 4

AIA imports only Agilent format TIFF-files, in which one file consists of stacked images for both channels. The analysis flow is the following: crop the image, place the grid on top of the array, defining the parameters and saving results.

The grid information can be imported to the AIA from Agilent's grid file, GAL-formatted file, Agilent design file, tab text file or manually. The software shows the inputted values for the grid and after that the grid may be fitted on top of the image manually or automatically. After this it is possible to fine tune the locations for whole grid, sub grids and individual spots. The grid has to be saved and after that the feature extractor can be called, in which we have to define which modules need to be ran and define the parameters for the modules, like the spot analyzing method, background subtraction method, normalization and quality measures. After this we can run the feature extractor. When it's ready, all the results will be saved automatically and it shows the visual results on top of the image. All outliers (uneven intensity or background, uneven shape or population outliers) are shown with differently colored circles on top of the spots. Dark blue circles are non-outliers.

In AIA there is two different ways to analyze spots, cookie cutter and whole

spot methods. Cookie cutter is a circle based method and the whole spot method bases on finding the edges of the spot.

Six different background calculation methods are implemented in AIA including no background subtraction, local background, average of all background areas, average of negative control features (needs information of the negative controls), minimum signal on feature and minimum signal on feature of background.

On normalization, there is linear and lowess methods, and combination of these two. In here it is also possible to select which spots are used for normalization.

On AIA you can look at intensity distributions across the slide (line of pixels) or defined smaller area, like unique spot.

AIA gives out 5 different result files: GEML, MAGE, JPEG, Text and visual results. The text file includes the results in tab-delimited format, including raw data (intensities, backgrounds, means, medians etc) and calculated values (all intermediate values from data processing steps). Depending slightly on the options selected, there are 90-100 columns of data. Note, that AIA logarithmic values are 10-based logarithms.

Pros

- Easy to use
- Versatile possibility to adjust grids and spots before spot finding
- Multiple options for spot analysis method
- Multiple options for background calculations

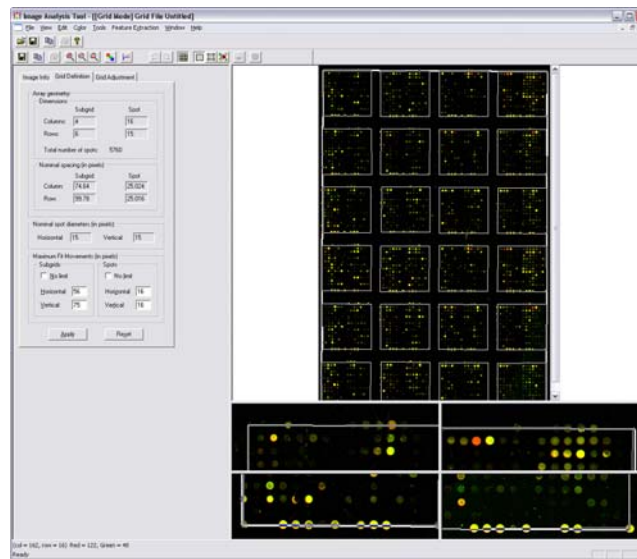
Cons

- Takes in only Agilent TIFF-files
- For array size of 30 000 spots, the analysis of one array takes 20-30 minutes
- No possibility to change spot calls or locations after analysis

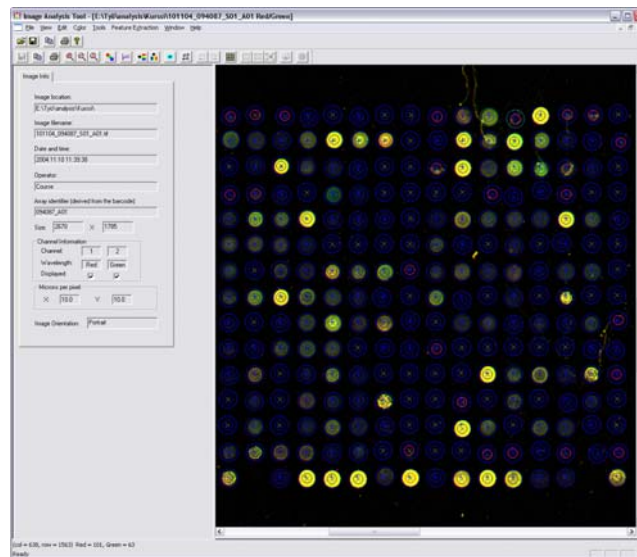
## 1.2 Freeware image analysis software

A wide variety of software for microarray image quantitation is available on the Web. For an academic user, paying thousands of euros for a license of commercial software might not be possible. For the average mortal - with limited budget funding - using a free-ware program is often the only option.

This last part of the chapter represents three such programs. In addition to being free to use, they all had to fulfill two additional criteria: independence of any other software (e.g. Matlab) and ability to run in Microsoft Windows. All programs are meant for quantitation of spotted arrays. For a comprehensive list of available quantitation software, see [http://ihome.cuhk.edu.hk/~b400559/arraysoft\\_image.html](http://ihome.cuhk.edu.hk/~b400559/arraysoft_image.html).



**Figure 1.2:** AIA in action: grid adjustments. Circles over the spots include quality information in color coding.



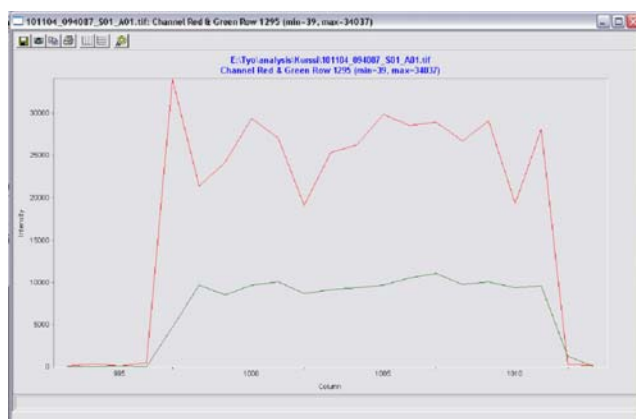
**Figure 1.3:** AIA in action: spots after analysis. Circles over the spots include quality information in color coding.

### 1.2.1 TIGR Spotfinder

Version tested: 2.2.3

Available at: <http://www.tigr.org/software/tm4/spotfinder.html>

The process to quantitate images with Spotfinder is quite straight-forward: load the images, create subgrids and overlay them on the subarrays in the image,



**Figure 1.4:** AIA view of spot intensity distribution.

process the grids and export the data. Overlaying the subgrids, or putting them on the right place, takes some 20-30 minutes for a large 20,000 spot image. If the subarrays aren't perfectly in right angle with the image, the grids can be tilted to fit them better. A pattern identification algorithm then automatically locates the borders of the spots and calculates both foreground and background intensities. Each spot is given a quality control value based on its form and intensity. This parameter can be later used for example in determining present spots. Also included is a possibility for some quick light-weight data analysis.

The learning curve for using the program is short thanks to the intuitive user interface. Earlier versions of the program used to have bugs that caused crashes demanding the user to save his work frequently. The version tested gives a few warnings every now and then but is stable.

#### Pros

- Quick to learn, fast to use
- Subgrids and cells can be twisted and resized manually for optimal results
- Automated spot identification
- Intuitive user interface enditemize

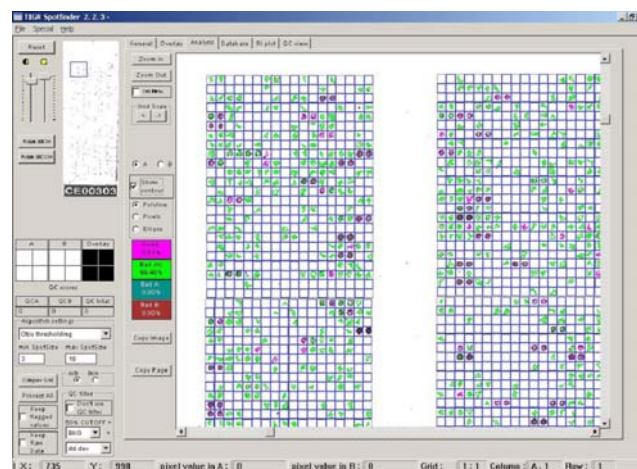
#### Cons

- Exporting directly to Excel works only for individual subgrids
- Only local background can be calculated

### 1.2.2 ScanAlyze 2

Version tested: 2.50

Available at: <http://rana.lbl.gov/EisenSoftware.htm>



**Figure 1.5:** AIA view of spot intensity distribution.

The first thing the user notices of ScanAlyze is its weird user interface. Each function from image loading to grid creation has its own window. A special Window control box is used for opening other windows. With a normal screen resolution (1024 x 768) all contents of most windows cannot be viewed at once requiring scrolling the windows up and down. All this makes using ScanAlyze a painful experience. It is nice that software designers try different approaches in user interface design but in this case originality comes with a price.

After loading images for both channels, the user needs to create the spotting grids. Strangely, only grid structures of 1, 4, 16 and 32 subgrids are available. For quantitating arrays with a different array format this is a major obstacle requiring unnecessary tricks to get the job done. Moving the grids is difficult as drag and drop with the mouse works only for individual grids. Also zooming in and out in the image is very slow. Black image background makes it very hard to determine subarray boundaries if they are closely packed and/or most spots have low intensity. Position for several spots needs to be individually assigned which means that a lot of time is required for quantitating a large array.

The program assumes that all spots are round which is not always true. Everything inside the circle is considered foreground which causes biased results if the assumption of roundness is not fulfilled.

#### Pros

- Easy to use auto-uninstall utility included

#### Cons

- Poor user interface
- Low intensity spots not shown properly when zoomed out

- Number of subgrids not freely adjustable
- Spots are assumed to be round
- Zooming and moving around the image is slow
- Quantitation takes time due to the amount of manual work needed

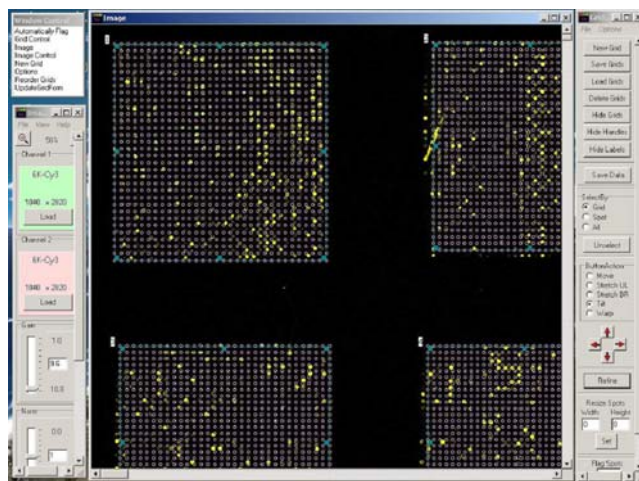


Figure 1.6: AIA view of spot intensity distribution.

### 1.2.3 GridGrinder

Version tested: 1.3

Available at: <http://gridgrinder.sourceforge.net/>

Makers of GridGrinder claim that it can automate the process of array spotting and quantitation. All the user needs to do is to tell which images to process and give some information of the subarray architecture. After doing this, you should be able to just lay back and watch the computer crush some numbers. Not only saving time and labor, this should also result in more reproducible outcome since the program does its job more objectively than human. Unfortunately, the reality isn't that care-free.

The user interface is very clear and therefore easy to learn. The images to be processed are loaded in a list, output folders set, the arrays' architecture defined and the processing started. Depending on the size and number of arrays, processing takes anything from few to several minutes on a normal desktop computer. In addition to the numerical output data file, the program also produces a copy of the original image files showing the spots it recognized. They can be used for refining parameters if the program failed to identify all spots correctly. The actual output file contains plethora of information about the spots and their foreground and background stats. That should satisfy even the most demanding data analyst.

When tested with a medium-density array with 6,144 spots, the program was able to find the location of all spots correctly after a few tries. Processing one image took one to two minutes which is reasonable. Performance was much worse when tested with an image of a large array with some 20,000 spots. The task was especially hard since most spots had no or very low intensity values. This time the processing took closer to 10 minutes and the program failed to locate the spots.

In conclusion, the program works fine with smaller arrays which have most spots present. The performance gets worse when less spots are present and noise increases. After optimized for a specific array design, the program can speed up research to some extent. It is up to the researcher if he or she finds that improvement worth it. After all, most projects today include the use of 10–20 arrays. An experienced user can quantitate those easily in a day with semi-automatic software like TIGR Spotfinder.

#### Pros

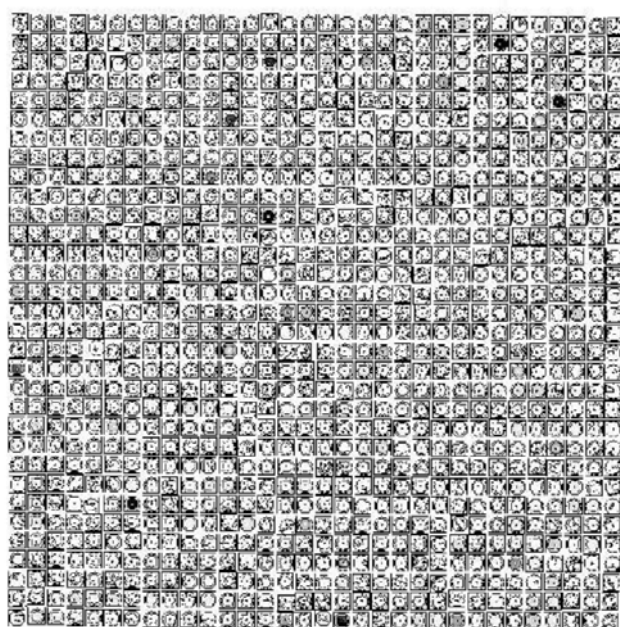
- Plain and intuitive user interface
- Automated processing suits well for some arrays
- Can quantitate several images in a batch
- Output data includes a diverse set of measurements

#### Cons

- Performance degrades if noisy and dense images are used
- The time saved with automated processing is marginal for small microarray projects

### **Suggested reading**

1. Gonzalez, R.C., Woods, R.E. (2002) Digital Image Processing, Prentice-Hall, New Jersey.
2. Russ, J.C. (1999) The Image Processing Handbook, CRC Press, Boca Raton.
3. Zhang, W., Shmulevich, I., Astola, J. (2004) Microarray Quality Control, John Wiley & Sons, New Jersey.



**Figure 1.7:** *AIA view of spot intensity distribution.*

# 2 Web extra: Basic statistics

## 2.1 Statistics using GeneSpring

A few examples of statistical data manipulation, simple statistics calculation and statistical testing using the DNA microarray data analysis software GeneSpring are given in this section.

### 2.1.1 Simple statistics

Genewise Simple statistics, like the average, minimum, maximum, standard error of the mean, and standard error can be produced through *Edit->Copy->Copy Annotated Genelist* in GeneSpring. After pasting the information in Excel or other spreadsheet program, the values will become apparent. These simple statistics can be produced for intensities of either channel (raw and control data in GeneSpring), intensity ratio and log-transformed intensity ratio (normalized data in GeneSpring).

### 2.1.2 Transformations

In GeneSpring, data transformations are linked to Experiment Interpretation. There are three options to choose from: Non-transformed data (ratio)  $\log_2$ -transformed data (log of ratio) and fold change . When one transformation is chosen, GeneSpring will automatically recalculate the data values, and use the new values for any subsequent statistical analyses (statistical group comparison, k-means clustering, etc.).

### 2.1.3 Scatter plot and histogram

A scatter plot can be produced in GeneSpring through *View->Scatter plot*. The shown axes can be modified from *View->Display Options*. A scatter plot can easily be used for testing the linearity of the data.

A histogram is displayed, if you have selected the *View->View Graph*, and additionally have set up parameters so that the quantification results are shown separately for every hybridized chip. For example, in a simple time series experiment, setting time as a non-continuous parameter would produce a histogram of expression values. You can use histograms for assessing the distribution of the data. After log-transformation the distribution of expres-

sion values should be symmetric and one-peaked.

### 2.1.4 Correlation

The Pearson correlation between chips is automatically calculated. The values of correlation coefficients can be viewed through Condition Inspector. Condition Inspector is invoked when the right-hand mouse button is clicked over one chip in the navigator bar, and Inspect is selected. From the opening window, select the Similar Conditions tab. The correlation coefficients between the selected chip and all the other chips are reported in the column Correlation (Figure 2.1).

Correlation	Experiment Name	Mouse	Signal channel
0.66275	Mouse testis again	2	3
0.59187	Mouse testis again	6	3
0.58806	Mouse testis again	5	3
0.56766	Mouse testis again	4	3
0.48144	Mouse testis again	3	3
-0.31119	Mouse testis again	4	5
-0.34872	Mouse testis again	3	5
-0.44185	Mouse testis again	5	5
-0.48318	Mouse testis again	6	5
-0.49606	Mouse testis again	2	5
-0.53548	Mouse testis again	1	5

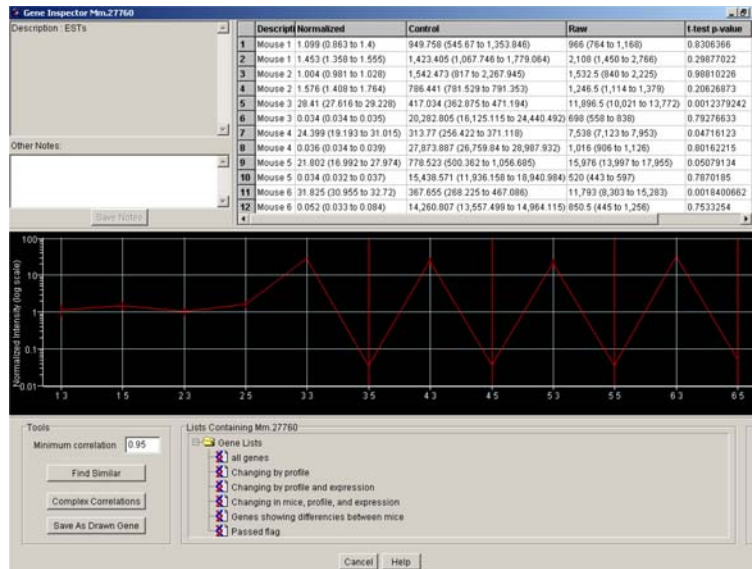
**Figure 2.1:** The Pearson correlation in GeneSpring is found under the Similar Conditions tab in Condition Inspector.

### 2.1.5 Linear regression

GeneSpring can calculate a linear regression model producing a line of best fit for a 2D scatter plot view. The line of best fit is produced from *View > Display options*. Select the Lines to Graph tab, and tick Line of Best Fit box. The linear regression line is overlaid with the scatter plot, and the regression equation of form  $y = aX + b$  is displayed at the bottom of the scatter plot view. Recall that  $a$  in the regression equation equals the correlation coefficient between the two variables plotted along the axes.

### 2.1.6 One-sample t-test

The one sample t-test in GeneSpring is automatically calculated for all the genes, whenever replicates are available. The t-test  $p$ -values can be found from the Gene Inspector, which is opened by double-clicking the left mouse button over a gene (Figure 2.2). The same  $p$ -values are also reported in the Spreadsheet, which is invoked from *File->View as spreadsheet*.

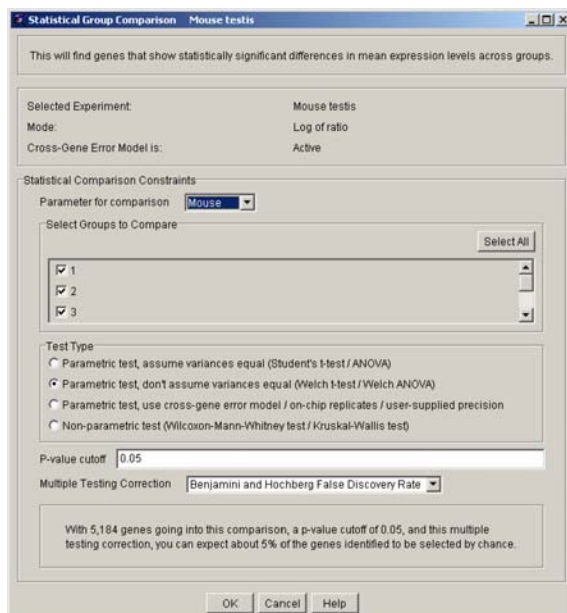


**Figure 2.2:** In GeneSpring the  $p$ -values for the one-sample t-test are found from Gene Inspector

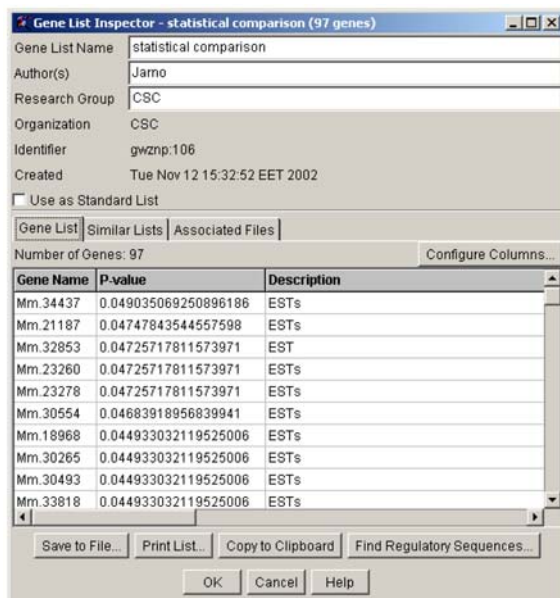
### 2.1.7 Independent samples t-test and ANOVA

The independent samples t-test and ANOVA are located in *Tools->Filtering and statistical analysis*. Clicking the right hand mouse button on an experiment in the opening window enables one to Add Statistical Group Comparison. You can specify the parameters by which the compared groups are defined, select the groups to compare, select the appropriate statistical test and adjust  $p$ -value and multiple testing correction (Figure 2.3).

The result of the statistical group comparison is a list of genes, which are statistically differentially expressed between the specified groups. The actual  $p$ -values for these genes can be found from the Gene List Inspector, which can be opened by clicking a gene list with the right hand mouse button, and selecting Inspect (Figure 2.4).



**Figure 2.3:** In GeneSpring statistical group comparison is a tool of its own.



**Figure 2.4:** The result of the statistical group comparison is stored into a genelist, which can be viewed with a Genelist Inspector.

## **Part II**

### **WEB extra: Analysis**

# 3 Web extra: Preprocessing of data

Jarno Tuimala

## 3.1 Examples using GeneSpring

This section describes some preprocessing examples using the GeneSpring program.

### 3.1.1 Importing data

Before any preprocessing can be done for the data using GeneSpring, it should be imported into the program. GeneSpring read virtually any file format if it is a tab-delimited text file. In GeneSpring you need to specify what kind of information some key columns contain. Basically, the minimal datafile contains just two columns: one for the gene name (it can also be Genbank accession number) and another one for the intensity (Affymetrix) or intensityratio (cDNA chips). Because importing data is covered very thoroughly in the online manual accessible from *Help->Online documentation*, we do not go into details here.

### 3.1.2 Background subtraction

GeneSpring subtracts the background from the foreground or spot intensities automatically, if the background intensities are present in the datafile.

### 3.1.3 Calculation of expression change

For cDNA microarrays this is one normalization option, Per Spot: Divide by control channel, but the option is not used with lowess-normalization. There are two possible transformations, log-transformation (log of ratio) and fold change. The transformation options can be accessed from *Experiments -> Experiment interpretation*.

For the Affymetrix chips, a calculational control channel is created. The expression is then calculated as with cDNA microarrays. It is also possible to calculate the expression change using a certain control chip. This is one

**Table 3.1:** *Time series experiment*

	Time point	Replicate
chip 1	1 hour	1
chip 2	1 hour	2
chip 3	2 hours	1
chip 4	2 hours	2
chip 5	3 hours	1
chip 6	3 hours	2

of the normalization options, Per Spot: Normalize to control samples.

Calculation of the expression change is coupled to normalization. Normalization only affects the control channel, which is adjusted so that the median intensity ratio of the chip is one when using Per chip: Normalize to percentile normalization. Therefore, the control channel is created for the Affymetrix chips, also. Note, that the original raw intensity values are always retained.

#### 3.1.4 Replicates

When importing the data from a chip where the same spot is present in multiple copies, GeneSpring automatically calculates an average of those replicates. This is assuming that the replicate spots have exactly the same Gene identifier in the data file. Replicate chips are also averaged, after defining the replicates in the *Experiment -> Experiment parameters* window and setting up the parameters in *Experiment -> Experiment interpretation*.

For example, if we have a time series experiment (Table 3.1) with three time points and two replicates per every time point, the parameters should be set up as follows. Two parameters, a time point and a replicate are created. In the time point, mark the time points suitably. The replicates are then set up using the other parameter. Inside one time point, the replicates are marked with a running number. Last, set up the value order for the time points. This tells GeneSpring, in which order the time points should be displayed on the screen.

If there is missing data for some genes either when importing the data or when setting up the replicate chips, GeneSpring only uses the existing data for the calculation of means.

#### 3.1.5 Checking linearity

Linearity is easily checked in GeneSpring using the M versus A plot. Go to scatter plot (*View -> Scatter plot*). Change the display options (*View -> Display options*) so that the horizontal axis is the average of raw and control (A), and the vertical axis is normalized (M). To help plot the non-linearity, from the Lines to Graph tab tick the Line of best fit box, which draws the linear regression line to the plot. Note, that the M versus A plot should be

initially produced for unnormalized data.

### 3.1.6 Normality

Normality can be checked using the histogram. A histogram can be investigated in (*View -> Graph*) window by checking *All Samples* - interpretation mode, that is automatically produced for each experiment.

### 3.1.7 Filtering

A filtering tool can be invoked from the menu *Tools -> Filtering and statistical analysis*. A new window opens, where one genelists and one experiment should be selected. The actual filtering tool is accessed by right-clicking on the selected experiment and selecting *Add expression percentage restriction* from the opening list. Often the bad quality data is first filtered out. After that, the not-changing genes are removed from the dataset.

Using the scatter plot tool, define the intensity value, under which you can't trust your data anymore on either raw or signal channel. For cDNA chips, this is often around 200-1000 and for Affymetrix chips around 200. In the expression percentage filtering, use this signal value as a minimum cut-off. Also select that it applies to all the conditions. Create one such filter for both channels. Create a new genelists of the results (*Make list button*).

In the next filtering phase we will try to find the genes that are changing and we can trust in. Using the genelists created above, set up a new filter, where you select genes, which have expression values between 0.5 (minimum) and 2 (maximum). These genes are not showing any expression changes and are uninteresting to us. This filtering should also apply to the whole dataset (all conditions). After saving the new genelists, the filtering tool can be closed.

Go to the navigator bar and right-click on the good quality gene list. From the list, pick *Venn diagram -> Left (red)*. Similarly, add the not-changing and all genes genelists to the Venn diagram. Then select the *All genes* genelists from the navigator. From the Venn diagram identify the region that contains the genes included in the reliable genelists, but not in the not-changing genelists. Right-click on that area of the diagram, and make a list of these genes.

Now you have a list of genes, which are reliable and also changing. You're ready to proceed with the analysis, for example to clustering or classification analyses.

# 4 Web extra: Normalization

Jarno Tuimala, Ilana Saarikko, and M. Minna Laine

## 4.1 Using GeneSpring for normalization

GeneSpring offers the basic repertoire of normalization methods, including both per-chip and per-gene methods. GeneSpring includes calculation of intensity ratios and dye-swap calculations as normalization methods. Dye-swap normalization in GeneSpring simply calculates the intensity ratio as (Cy5 / Cy3) instead of the usual (Cy3 / Cy5) for the specified samples.

Normalization methods in GeneSpring 5.x are

- lowess-smoothing (per-chip and per-gene)
- median polishing (per-chip and per-gene)
- median or percentile scaling (per-chip or per-gene)

Also housekeeping genes or spiked controls can be used for normalization (Per-chip: normalize to positive control genes) . If chips have been scaled in the Affymetrix MAS program to a certain mean intensity value, the transformational effect can be removed in GeneSpring using Per-chip: Normalize to a constant value option. If one-color data is analyzed, and the experiment includes one control chip that is used for the calculation of intensity ratio, this can be done in GeneSpring using Per-gene: Normalize to specific samples option. Furthermore, normalization can be applied only to certain samples, if needed.

GeneSpring automatically detects the imported data type (one- or two-color), and suggests a normalization scheme for the experiment. These schemes are following

For one-color data (Affymetrix or nylon filter):

- Data transformation: Set measurements less than 0.0 to 0.0
- Per-chip: Normalize to 50th percentile
- Per-gene: Normalize to median

For two-color data (cDNA microarrays) either:

- Per-chip and per-gene: Intensity dependent (lowess) normalization  
or
- Per spot: divide by control channel
- Per-chip: Normalize to 50th percentile

Normalization methods are accessed through *Experiments* -> *Experiment normalizations* in GeneSpring. GeneSpring gives a warning if the normalization you are trying to perform does not make any sense (red text), or that the calculation can be performed, but the results might be unreliable (orange text). Note that the normalization methods are applied in the same order as they appear in the list.

# 5 Web extra: Cluster analysis of microarray information

Mauno Vihinen and Jarno Tuimala

## 5.1 GeneSpring and clustering

GeneSpring offers five different clustering and classification algorithms, one of which is a supervised method. These are hierarchical clustering (gene and experiment trees), K-means clustering, self-organizing maps, principal component analysis, and parameter value prediction. Clustering tools are invoked from the menu *Tools->Clustering*, the supervised classification method can be found from *Tools->Predict parameter values*, and the principal component analysis is located in *Tools->Principal components analysis*.

### 5.1.1 Clustering tool

The clustering tool has four fields, which need to be filled in before running the analysis (Figure 5.1). From top to down, the first box indicates the gene list to be clustered. Initially the list is the same that was highlighted when the clustering tool was invoked, but it can be changed from the navigator bar on the left. Next box contains the information about the experiment to be clustered. This can also be easily changed from the navigator bar on the left. The pull-down menu offers the choice of three clustering algorithms (hierarchical, K-means and SOM). The setting for the current analysis is in the box below the pull-down menu. For K-means, the number of clusters needs to be specified, as well as the number of iterations and the desired measure of similarity.

GeneSpring has several different similarity measures, which fall into the following categories: correlation, confidence, and distance. The selection of the similarity measure should be given some thought, because it significantly affects the generated results. Pearson's correlation emphasizes both over- and underexpressed genes, and the Standard correlation finds especially overex-

pressed genes. Spearman's correlation is highly similar to Pearson's correlation except it uses ranks for the calculation of the correlation coefficient (and is thus a nonparametric measure of correlation). Distance measures the euclidian distance between two gene expression profiles. It is calculated as the square root of averaged squared deviation of the profiles. Spearman's confidence measures the probability of getting a correlation of  $S$  or higher by chance alone, if the true correlation is zero.

If there is only one measurement per gene (*i.e.*, one chip), only the distance measure can be used for the clustering of the genes. If there are two replicates of every gene, the Standard correlation can also be used. If there are three replicates, Pearson's correlation becomes available, and with five replicates the confidence measures can be used.

In other words, the decision about the applied similarity measure depends on the biological question you are interested in, and the amount of replicates in your dataset.

After specifying the aforementioned settings, the run can be started by clicking the Start-button on the bottom of the window. When the run has ended, you can name and save the clustering result. The result appears on the main screen of GeneSpring. Thereafter, all the results can be found from the navigator bar under the folder classification. Note that the hierarchical clustering results are stored under two different folders, Gene Trees and Experiment Trees.

After viewing the clustering results, you can get back to the original view by selecting *View->Unsplit window*. Hierarchical clustering results can be dismissed, for example, by selecting *View->blocks*.

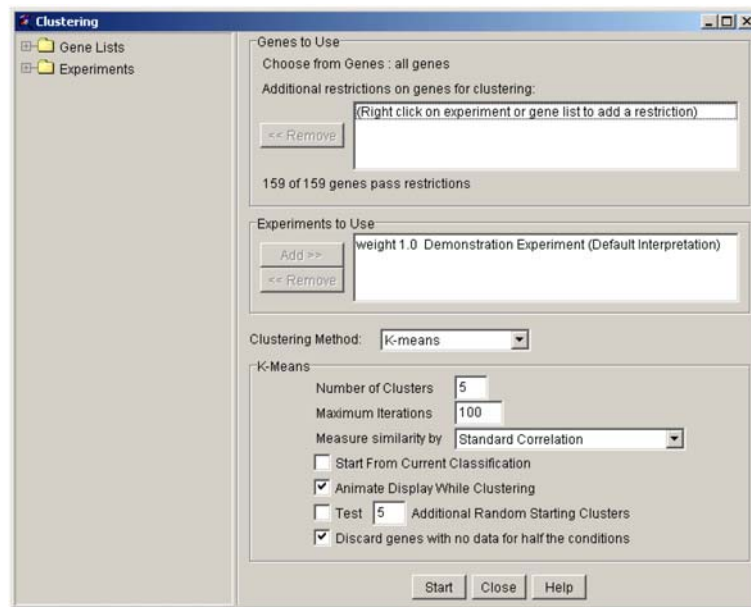


Figure 5.1: The clustering tool in GeneSpring.

### 5.1.2 Principal components analysis tool

Principal component analysis can be used for creating a set of the most significant expression patterns from the data. It can also be used for checking the results of the clustering methods. After selecting *Tools->Principal components method* the analysis is run, and the results are displayed. The opening window contains the most significant profiles. Double-clicking one profile transfers the view to the Gene Inspector, where genes with a similar expression profile can be searched.

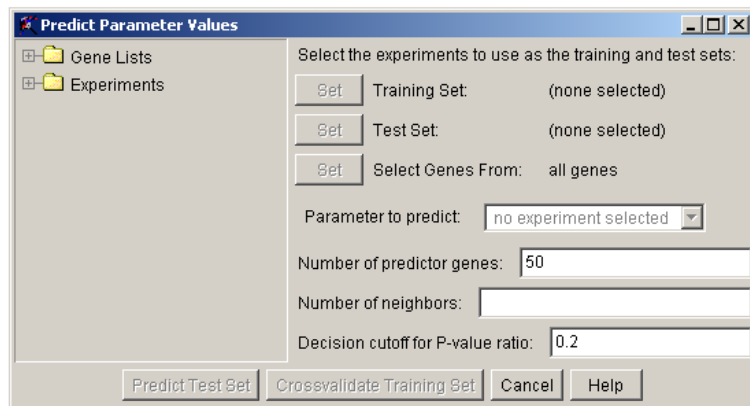
The results are displayed in a scatter plot format in the main window. If you want to compare PCA with some clustering results, you can right-click on one clustering result in the navigator bar, and select Set as coloring scheme from the appearing menu. A good clustering result often creates clusters that do not overlap with each other in the PCA scatter plot.

### 5.1.3 Predict parameter value tool

The predict parameter value tool (Figure 5.2) is used in situations, where we have a certain set of known samples, and based on these, we want to predict in which group the new, unknown samples fall. For example, if we have information about the leukemia type of 60 samples, we can find the genes, which differentiate these leukemia types from each other. After finding the suitable set of genes, the identity of the unknown samples can be predicted. GeneSpring uses the K-means algorithm described by Golub *et al.*

Training and test sets (experiments) need to be specified. GeneSpring also needs to know which parameter contains the information about the groups

to be compared (parameter to predict). After cross-validating the test set, the test set can be predicted. The result of the analysis is a prediction of the test set sample identities and a set of genes differentiating the selected groups.



**Figure 5.2:** *The predict parameter value -classification tool in GeneSpring.*

## **Part III**

# **WEB extra: Data mining**

# 6 Web extra: Data mining for promoter sequences

Martti Tolvanen, Jarno Tuimala and Mauno Vihinen

## 6.1 Using BioMart to retrieve promoter regions

The preferred option for retrieving upstream sequences is found at the BioMart service (<http://www.ensembl.org/Multi/martview>) of the Ensembl project. This is limited only by the number of Ensembl genes that are annotated and by the accuracy of the annotation. Many genes still have many multiple entries in Ensembl, and some entries are for pseudogenes. However, the service is very easy to use. Note that you will want to use primarily Ensembl data, not Vega (which is manually annotated but still very incomplete data).

For some microarrays (especially Affymetrix chips), BioMart provides direct mappings. Such mappings with microarray contents have become more common in the genome sites, both at Ensembl and at UCSC, and the annotations provided by the manufacturers have improved, too. Therefore, retrieving the upstream sequences is less of a technical problem now, but the problem of correct transcription start sites remains a serious one, especially in the case of alternative promoters..

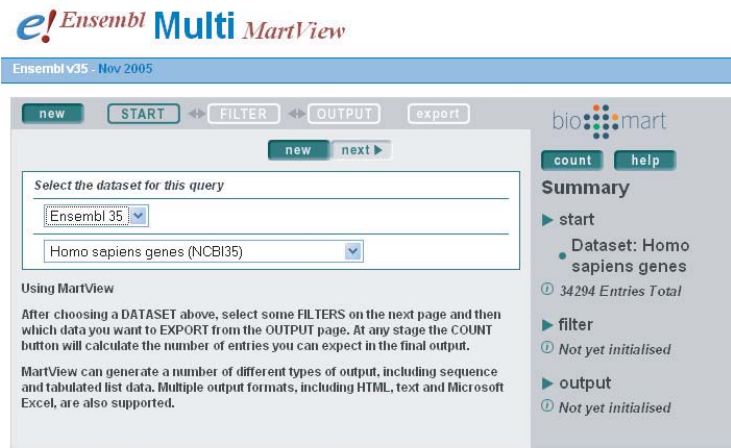


Figure 6.1: BioMart start screen.

Next, enter your list of gene identifiers in the Filter step:

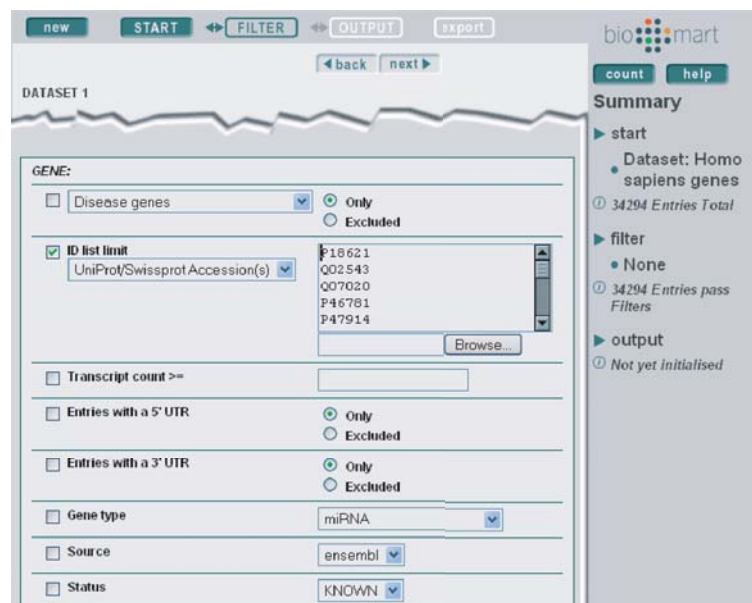
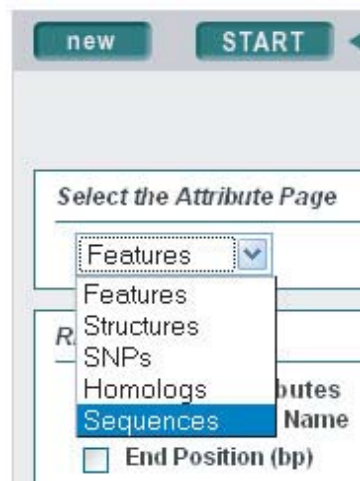


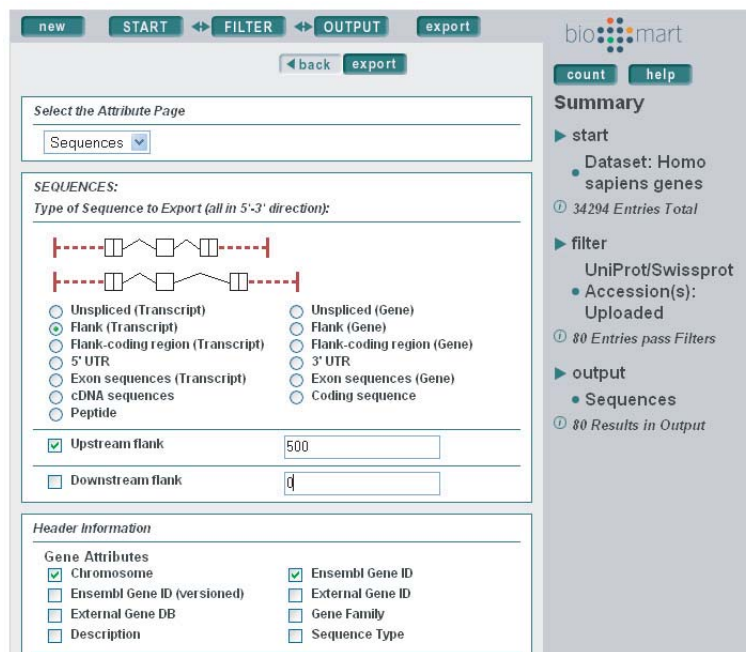
Figure 6.2: BioMart filtering. There are many options, and you should be aware that some mappings are more complete than others. Next to internal Ensembl references, RefSeq is your best choice. This example comes from data in which reliable SwissProt codes were provided. A long list of other filtering options is omitted from the figure.

Then, in the following output phase, you can optionally first choose your output as Features, to check the completeness and consistency of your results, and finally as Sequences to retrieve the data. The Features export

gives a tabular output which can be imported easily to other programs, e.g. by a direct copy/paste to MS-Excel..



**Figure 6.3:** Changing between Features and Sequences.



**Figure 6.4:** Sequence output options for obtaining only the 5'-upstream flank of your found genes in BioMart. You may want to add more options in the Header information and/or select a longer sequence region. Additional options for data compression, saving locally etc. are not shown.

```

>7| ENSG00000075624
ACTGCCTGGCCACTCCATGCCCCCAAGAGCTCCTTCTGCAAGGAGCGTACAGAAACCCAGGGCCCTGGCACCCCGTGCAGAC
CCTGGCCCCACCCACCTGGGGCGCTCAGTGCCCAAGAGATGTCCACACCTAGGATGTCCCGGGTGGGTGGGGGGCCCGAG
AGACGGGCAGCCCGGGGAGGCCATGCGGGGCGAACCAGGGCACTGCCAGCGTGGGGCGCGGGGGCCACGGCG
CGGCCCC&AGCCCGGGCCAGCACCC&AAGCGGCC&ACGCC&AA&ACTCCTCCCTCCTCCTCCTC&A&TCTCGGTC
TCGCTCTTTTTTTTTTCGCA&A&AGGAGGGGAGAGGGGGT&AAAA&ATGCTGCACTGTGCGGG&AAGCCGCT&AGT&GAGC
GGGCGGGGGCA&ATCAGCGTGC&CGCTTCC&AA&AGTTG&CCTTTT&ATGGCTCG&AGCGGGCGCGGGGGGGCC&T&AAAA&CC
CAGCGGCGCGACGCGCCACC

>7| ENSG00000075624
CTTTGGGCGCTAACTGCGTGC&CGCTGG&A&ATTGGCGCTAA&TTGCGCGTGC&CGCTGG&ACTCA&AGGCGCTAA&CTGCGG
TGCGTTCTGGGGCCGGGGTGC&CGGCGCTGGGCTGGGGCG&A&AGCGGGCTCGGGCG&A&AGGGTGGGGTGC&CGGCGCT
CCGGGCGCTTGC&CGCA&CTTCTG&CGGCG&CGCTGG&CGCGCC&GAGGGTGTG&CGCGCTG&CGCTGC&CGCGCGCG&ACCG
GCGCTGTTG&AACCGGGCG&AGGCGGGCTGG&CGCCGGTGG&AGGGGGTGGGGCTGGCTTCTG&CGCGCGCGCGG
GGGACGCTCCG&ACCAGTGT&TTG&CCTTTT&ATGGT&A&TA&ACGCGGGCGGGCGCGCTTCTTTG&TCC&CA&ATCTGGGCGCGC
GCGCGCGCCCTTGGCGGCT&A&AGG&ACTCGGCGCGCGG&A&AGTGGCC&AGGGCGGGGGCG&ACTCGGCTC&AC&AGCGCGCC
GGCTATTCTCG&AGCT&ACC

>20| ENSG00000101361
GGGAGG&AA&AGGATGC&ACCGCC&ACCCGGCGCGGGGAGGCTCGTCCCGAGTAGGGGACGG&AACCGCTGGGTC&CGGGG
ACGTGGGGCGC&AGG&TCTGGGGCC&GAGGCC&ACG&ACG&AA&GCGGG&AA&CGCGCT&AGG&AGCGG&TCTGTCTGG&ATG&GA
CGCCCGCGCC&ACCGGG&ACTCCCGCC&CG&ACTTCA&CC&AGCTCTGCTG&CGCGTAA&CTC&AT&AT&G&CA&A&G&A&C&AT&C&AG&AA
CCGG&ACT&AT&A&CC&CG&AA&AGGG&CCTTCC&CA&AGT&CGTTT&CGCGCGCTG&C&AGT&G&CG&AC&CCGGGG&CGCGCGTCC&CG&CA
ACC&AC&CGCC&CGCG&C&AGCTG&C&AG&GG&CG&AA&AGGG&CGCGCTCG&CGCGCGCTCG&G&ACC&ACC&AG&CGCGCGCTCC&GG
CG&AT&ACC&CCCTGG&CGGG&CCCTC&CA&A&AGG&CG&AG&AT&G&T&CGT&C&CGCGCGCTCG&G&ATT&G&T&GG&GGGG&CGGGGGG
TGCCTCTG&G&AGCT&G&GTTT

```

**Figure 6.5:** Sequence data in Fasta format, after the Export step.

For some microarrays (especially Affymetrix chips), BioMart provides direct mappings, so you can skip most of the previous steps. Instead, you can list all the genes on the chip directly from the Gene menu (Fig. 6), or choose a subset of them in the Filter step:.

The screenshot shows a web form with the following elements:

- A label "GENE:" in blue text.
- A checked checkbox followed by a dropdown menu showing "with AFFY-HG-U133 ID(s)".
- Two radio buttons: "Only" (which is selected) and "Excluded".

**Figure 6.6:** Selecting all genes that are represented on AFFY-HG-U133.

Such mappings with microarray contents have become more common in the genome sites, both at Ensembl and at UCSC, and the annotations provided by the manufacturers have improved, too. Therefore, retrieving the upstream sequences is less of a technical problem now, but the problem of correct transcription start sites remains a serious one, especially in the case of alternative promoters.

### 6.1.1 Final warnings regarding upstream data

- in some cases the genome assembly may be broken close to the start of your gene, so that you do not get a full 1000 (or whatever) nucleotides of sequence. In Ensembl this shows as a long string of N's in the sequence
- the start of a RefSeq mRNA may not always correspond to a true transcription start site. There are many mRNAs which give evidence of

longer transcript variants than the current RefSeqs. UCSC genome browser relies directly to the RefSeq start sites in their upstream data sets.

- Ensembl tries to extend the 5'-end as far as possible, but in a few cases this leads erratically long sequences, due to seemingly mis-spliced mRNA versions

## 6.2 GeneSpring and promoter analysis

GeneSpring includes a promoter analysis tool, which can be used for finding novel common regulatory sequences in a gene list, or to search for a known sequence. The tool can be invoked from *Tools->Find potential regulatory sequences*. In order to search for potential regulatory sequences, you need to have a whole genomic sequence of the organism under study. In principle, if only a partial genome of the organism is known, it is not possible to search for regulatory elements (GeneSpring forbids the use of the tool), because the statistical support and frequency values of the elements would be erroneous. However, there is a trick, which allows the analysis of partial genomes. For more information, see the tech note at [http://www.silicongenetics.com/cgi/TNgen.cgi/GeneSpring/GSnotes/Notes/how\\_contig](http://www.silicongenetics.com/cgi/TNgen.cgi/GeneSpring/GSnotes/Notes/how_contig).

The tool opens a new window (Figure 6.7). First, you need to select a genelist you want to study, but do not use the “all genes” or “all genomic elements” list, because then you would compare the whole genome against itself, which is not a viable analysis. From the pull-down menu, select whether you want to search for new sequences or for a specific sequence. You can also select the length of the sequence to be considered a promoter region, how long a regulatory element is being searched, and how many unknown bases are allowed. The longer the sequence, and the larger the number of unknown bases, the longer the analysis time. You have control over the probability statistics: The  $p$ -value cut-off for a significant pattern can be modified. Whether the sequence is relative to the sequence upstream of other genes or relative to the whole genomic sequence can also be modified. The first option is far more common.

After the analysis have completed, or you stop the search, the results are reported. They appear on right side of the toolbox. Potential regulatory sequences, the number of genes they were detected in, and the detection  $p$ -value are reported. The best findings are reported first.

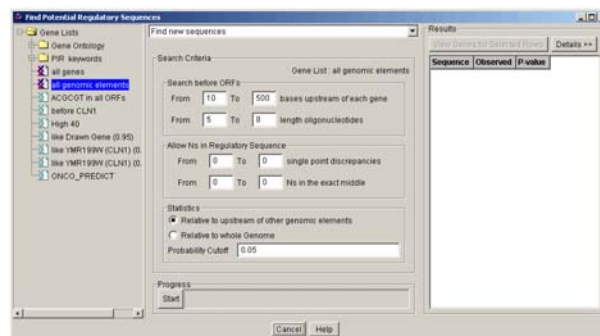


Figure 6.7: Find potential regulatory elements -tool in GeneSpring.

# 7 Web extra: Biological sequence annotations

Juha Saharinen

## 7.1 Using Ensembl server to batch process of annotations

Here are two small examples using these alternative interfaces to the Ensembl data. In example 1, 10 probe sets from Affymetrix U133A chip are queried from Ensembl *Homo sapiens* database and their Ensembl GeneStableIDs, and gene's sequence locations are returned. Example 2 shows a small Perl script, using EnsemblAPI, to retrieve the genomic sequence from a given organism, chromosome and physical position.

### 7.1.1 Example 1

```
bbu-juhaad@Opteronix:~> mysql -h ensembl.db.ensembl.org -u anonymous homo_sapiens_core_27_35a
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1217078 to server version: 4.0.18-standard-log
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

```
mysql> select xref.dbprimary_acc, gene_stable_id.stable_id, seq_region.name, gene.seq_region_start, gene.seq_region_end,
gene.seq_region_strand
from xref
join external_db on external_db.external_db_id = xref.external_db_id
join object_xref on object_xref.xref_id = xref.xref_id
join translation on translation.translation_id = object_xref.ensembl_id
join transcript on transcript.transcript_id = translation.transcript_id
join transcript_stable_id on transcript_stable_id.transcript_id = transcript.transcript_id
join gene on gene.gene_id = transcript.gene_id
join gene_stable_id on gene_stable_id.gene_id = gene.gene_id
join seq_region on seq_region.seq_region_id = gene.seq_region_id
where (external_db.db_name = 'AFFY_HG_U133A')
limit 10;
```

dbprimary_acc	stable_id	name	seq_region_start	seq_region_end	seq_region_strand
209137_s_at	ENSG00000102034	X	128924393	128970188	-1
209136_s_at	ENSG00000102034	X	128924393	128970188	-1
202264_s_at	ENSG00000196586	6	76515703	76682665	1
202264_s_at	ENSG00000118260	2	208220192	208289067	1
AFFX-HSAC07/X00351_5_at	ENSG00000196586	6	76515703	76682665	1
AFFX-HSAC07/X00351_5_at	ENSG00000166261	11	123100699	123117573	-1
217991_x_at	ENSG00000166261	11	123100699	123117573	-1
209137_s_at	ENSG00000125447	17	70744290	70769299	-1

```

| 209136_s_at      | ENSG00000125447 | 17 |      70744290 |      70769299 |      -1 |
| 217988_at      | ENSG00000163251 | 2  |      208456223 |      208459624 |      -1 |
+-----+-----+-----+-----+-----+-----+
10 rows in set (0.11 sec)

mysql>
mysql> quit
Bye

```

## 7.1.2 Example 2

```

#!/usr/bin/perl

use Bio::EnsEMBL::DBSQL::DBAdaptor;

sub main
{
    $db = Bio::EnsEMBL::DBSQL::DBAdaptor->new
        (-host => 'ensemldb.ensembl.org',
         -dbname => $ARGV[0],
         -user  => 'anonymous');

    # get the slice adaptor and fetch a slice on a given region
    $slice_adaptor = $db->get_SliceAdaptor();
    $slice = $slice_adaptor->fetch_by_region('chromosome', $ARGV[1], $ARGV[2], $ARGV[3]);

    # print out the sequence from this region
    print $slice->seq();
}

#start program
{
    if ($#ARGV != 3) {
        print "\nGenoFetcher - Juha Saharinen, Biomedicum Bioinformatics Unit & KTL, 2004\n\n";
        print "Use of GenoFetcher is \n";
        print "genofetcher <database> <chromosome> <chromosome_start> <chromosome_end>\n\n";
        print "Like genofetcher homo_sapiens_core_27_35a 12 1000000 1001000\n\n";
        print "This will return slice of human chromosome 12 from 1.000.000 to 1.001.000\n\n";
    }
    else {
        main;
    }
}

```

# 8 Web extra: Software issues

Jarno Tuimala and Teemu Toivanen

## 8.1 Programming

When the capacities of the spreadsheet program are not sufficient or many similar files need processing, some programming tools can be used instead for the datafile management purposes or for data analyses. There are actually several programming languages that are especially suited for the data file formatting and other text file manipulations.

### 8.1.1 Perl

One of the most common languages is Perl, which has very powerful and easy to use text manipulation tools. Perl is available for free for UNIX, Linux <http://www.perl.org> and PC machines <http://www.activeperl.com>. Perl is a programming language, which means that getting to know it takes some time and effort. However, if data conversion tools are needed everyday, it would definitely be worthwhile to befriend Perl.

To illustrate how easily text can be manipulated using Perl, we present a short example. The next code produces a complementary DNA sequence from the original sequence, which has been stored into the text string `$sequence`. Both the original sequence and its complement are printed on the computer screen.

The program starts with a line that tells the computer where The Perl software can be found. The next four lines contain the actual commands that manipulate the DNA sequences. Note that every command line has to end with a semicolon. Function `tr` makes a complementary sequence from the original one, and function `print` outputs the result to the screen.

```
#!/usr/bin/perl
$sequence="aaattcgagtaggtcaggcat";
print "Original:      $sequence\n";
$sequence=~ tr/acgtACGT/tgcaTGCA/;
```

```
print "Complementary:      $sequence\n";
```

You can use pico editor on CSC's Cedar server to create a similar file and test the example yourself. Perl programs are started in Cedar with a command `perl filename`.

### 8.1.2 Awk

Awk is a standard UNIX and Linux tool, which is available on CSC's servers. With Awk, individual columns can be easily extracted from tab-delimited text files. Using other standard UNIX tools, these individual columns can be saved into a new text file. For example, the next script takes the first column from a specified datafile and saves it into a new file.

```
Awk '{print$1}' datafile > newfile
```

Two columns can be "awked" into a new file next to each other separated with a space:

```
awk '{print$1, $2}' datafile > newfile
```

or one after another:

```
awk '{print$1}{print$2}' datafile > newfile
```

### 8.1.3 R

R is a free statistical analysis tool and a self-sufficient programming language. It is available for UNIX, Linux, Macintosh and PC platforms. In R, scripts for analyses and data file manipulations can be easily constructed. R has many add-on packages for cluster analysis, self-organizing maps, and neural networks, among others. There are also many packages available that have been specifically tailored for DNA microarray data analysis: Bioconductor project develops and updates many of these packages.

Here is an example on how to read the tab-delimited datafile to R and how to process it into a new table, which is then written out to a new file. The function `read.table` reads in the specified file with the headers. The table is then saved in a variable `data`. Two columns are extracted from the data, and saved into new variable (`x` and `y`). The new variables are used for the creation of a new table (`dataout`), which is then written to a new text file (`filenameout.txt`). Such a script can easily be automated using R, and the analyses can simultaneously be integrated with the datafile conversions.

```
data<-read.table("filename.txt", header=T)  
x<-data$greenintensity
```

```

y<-data$redintensity
dataout<-cbind(x,y)
sink("filenameout.txt")
dataout
sink()

```

Many images included in chapters 5–8 have been produced by R using real DNA microarray datasets.

## 8.2 Common code examples

This section covers common problems when manipulating text formatted datafiles that most programs export and import. Problems is that those file formats are program specific and using them for analysis is hard without a lot of editing. These are on Linux systems, but same code should work on all UNIX flavours and even in windows with minor changes.

For the actual code examples, see online material at <http://www.csc.fi/oppaat/siru/>.

### 8.2.1 Read tabular data

This is the basis for rest of the examples (copy/paste this first and then add what you want from the latter examples). Reads tabular files (change variables for sectioned tabular data). Please note that examples are in both Perl and Python languages and cannot be intermixed.

#### Perl

```

1  %%
2  %% This is file '.tex',
3  %% generated with the docstrip utility.
4  %%
5  %% The original source files were:
6  %%
7  %% fileerr.dtx (with options: 'return')
8  %%
9  %% This is a generated file.
10 %%
11 %% Copyright 1993 1994 1995 1996 1997 1998 1999
12 %% The LaTeX3 Project and any individual authors listed elsewhere
13 %% in this file.
14 %%
15 %% This file was generated from file(s) of the Standard LaTeX 'Tools Bundle'.
16 %% -----
17 %%
18 %% It may be distributed and/or modified under the
19 %% conditions of the LaTeX Project Public License, either version 1.2
20 %% of this license or (at your option) any later version.
21 %% The latest version of this license is in

```

```

22 %%   http://www.latex-project.org/lppl.txt
23 %% and version 1.2 or later is part of all distributions of LaTeX
24 %% version 1999/12/01 or later.
25 %%
26 %% This file may only be distributed together with a copy of the LaTeX
27 %% 'Tools Bundle'. You may however distribute the LaTeX 'Tools Bundle'
28 %% without such generated files.
29 %%
30 %% The list of all files belonging to the LaTeX 'Tools Bundle' is
31 %% given in the file 'manifest.txt'.
32 %%
33 \message{File ignored}
34 \endinput
35 %%
36 %% End of file '.tex'.
#!/usr/bin/perl -w
#filename to read, get from commandline
my $filename=shift;
#open file (F is the filehandle)
open(F, $filename);
#specify data delimiter \t = tab, \; = semicolon
$delim="\t";
#this will have the data matrix
@data=();
#put this to 1 if first row is a header
$hasHeader=1;
#this includes header
@headerData=();
#use filter to specify when data section starts and ends
#not needed, for normal tabular data
$useFilter=0;
#search for this string in the beginning of the line to start data section
$startFilter="BEGIN DATA";
#search for this string in the beginning of the line to end data section
$endFilter="END DATA";
#internal variable so that we know when we are at data section
$doInsert=0;
#loop while there's data
while($line=<F>) {
    chomp($line);
    if ($useFilter==1) { # go here if we are using filters
        if ($doInsert==0) { # not in data section
            if ($line =~ /$startFilter/) { #search for beginning of data
                $doInsert=1; #go to data phase
            }
            next;
        } else { # we are in data phase
            if ($line =~ /$endFilter/) { #search for the end of data
                $doInsert=0; #exit data phase
                next;
            }
        }
    }
}
}

```

```

#data insertion section
if ($hasHeader==1) {
    $hasHeader--1; # we come here only once
    #split header row with $delim to a vector
    @headerData=split($delim, $line);
    next; #read next line
}
#add new row to matix data, that has $line split with $delim
#to a vector (which is that row of the matrix)
push(@data, [split($delim, $line)]);
}
#now we have data in @data matrix and possible header info in @haderData

```

## Python

```

#!/usr/bin/python
from sys import argv #for filename
import re,string # regular expressions and strings
#filename to read, get from commandline
filename=argv[1]
#open file (f is the filehandle)
f = open(filename, 'r')
#specify data delimiter \t = tab, \; = semicolon
delim="\t"
#this will have the data matrix
data=[]
#put this to 1 if first row is a header
hasHeader=1
#this includes header
headerData=[]
#use filter to specify when data section starts and ends
#not needed, for normal tabular data
useFilter=0
#search for this string in the beginning of the line to start data section
startFilter="BEGIN DATA"
#search for this string in the beginning of the line to end data section
endFilter="END DATA"
#internal variable so that we know when we are at data section
doInsert=0
#loop while there's data
line=f.readline()
while len(line) != 0:
    next=0
    line=re.sub("\n|\r","",line)
    if (useFilter==1):# go here if we are using filters
        next=1
        if (doInsert==0): # not in data section
            if (re.search(startFilter, line) != None): #search for beginning of data
                doInsert=1 #go to data phase
                next=1 #do not append data
        else: # we are in data phase
            next=0

```

```

        if (re.search(endFilter, line) != None): #search for the end of data
            doInsert=0 #exit data phase
            next=1 #do not append data
#data insertion section
if (hasHeader==1):
    hasHeader=-1 # we come here only once
    #split header row with $delim to a vector
    headerData=string.split(line, delim)
    next=1 #do not append data

#add new row to matrix data, that has $line split with $delim
#to a vector (which is that row of the matrix)
if next != 1:
    data.append(string.split(line, delim))
line = f.readline()

#now we have data in a matrix and possible header info in headerData

```

### 8.2.2 Filtering/Rows

Select what column to filter (column) and what to search (search).

#### Perl

```

#column number to use for filtering (columns numbered from 0,1,2...)
$colnum=0; # first column
#what to search
$search="1";
#check each data row
@filteredData=();
for my $row (@data) {
    #exact match 'eq', numeric match '==', numeric greater or equal >=
    # =~ /$search/ for regexp match
    if ($row->[$colnum] eq $search) {
        push(@filteredData, $row);
    }
}
#replace @data with filtered data @ret
@data=@filteredData;

```

#### Python

```

#column number to use for filtering (columns numbered from 0,1,2...)
colnum=0 # first column
#what to search
search="1"
#check each data row
filteredData=[]
for row in (data):
    #exact match '==', numeric match '==', numeric greater or equal >=
    # re.search(pattern, string for regexp match
    if (row[colnum] == search):

```

```

        filteredData.append(row)
#replace data with filtered data ret
data=filteredData

```

## Filtering/Columns

### Perl

Change "cols"list to reflect wanted columns

```

#column number in a list (rest is filtered out, columns nubered from 0,1,2...)
@cols=(0,2); # first and third column
@filteredData=();
#header cols
if (scalar(@headerData)>0) { #check that header is not empty
    @headerData=@headerData[@cols];
}
for my $r (@data) {
    my @row=@{$r};
    my @tmp;
    @tmp=@row[@cols];
    push(@filteredData, \@tmp);
}
#replace @data with filtered data @ret
@data=@filteredData;

```

### Python

```

#column number in a list (rest is filtered out, columns nubered from 0,1,2...)
cols=[0,2] # first and third column
filteredData=[]
#header cols
if (len(headerData)>0): #check that header is not empty
    tmp=[]
    for x in cols: # append all columns
        tmp.append(headerData[x])
    headerData=tmp
# for each row
for row in (data):
    tmp=[]
    for x in cols:
        tmp.append(row[x]) # append all columns
    filteredData.append(tmp)
#replace data with filtered data ret
data=filteredData

```

### 8.2.3 File conversions/print/write

First print header (if exists) and then rest of the data with selected delimiter.  
 Use > 'filename' to print to a file.

### Perl

```
# open(FILEOUT, "> outfile.txt"); #remove leading hash for file writing
# also change STDOUT to FILEOUT later on this file
#output delim
$outDelim=";";
if (scalar(@headerData)>0) { #check that header is not empty
    #print header with $outDelim as delimiter
    print STDOUT join($outDelim,@headerData);
    print STDOUT "\n"; #new line
}
#for each row
for my $row (@data) {
    #print row vector with $outDelim as delimiter
    print STDOUT join($outDelim,@{$row});
    print STDOUT "\n"; #new line
}
# close(FILEOUT); #see open(..) comments
```

### Python

```
#output delim
outDelim=";";
if (len(headerData)>0): #check that header is not empty
    #print header with outDelim as delimiter
    print string.join(headerData,outDelim)
#for each row
for row in data:
    #print row vector with outDelim as delimiter
    print string.join(row,outDelim)
```

# Index

## A

annotations  
retrieving, 40

## G

GeneSpring  
ANOVA, 21  
average, 19  
background correction, 24  
classification, 31  
clustering, 29  
experimental parameters, 25  
expression change, 24  
filtering, 26  
fold change, 19  
genelist, 26  
histogram, 19, 26  
importing data, 24  
linear regression, 20  
linearity check, 25  
log of ratio, 19  
maximum, 19  
minimum, 19  
normalization, 27  
  constant value, 27  
  dye-swap, 27  
  one-color data, 27  
  per chip, 27  
  per gene, 27  
  positive control genes, 27  
  specific samples, 27  
  two-color data, 27  
  warnings, 28  
one sample t-test, 21  
Pearson correlation, 20  
principal component analysis, 31  
promoter analysis, 39  
ratio, 19

replicates, 25  
scatter plot, 19, 26  
standard error, 19  
standard error of the mean, 19  
two-sample t-test, 21

## P

promoter data mining  
retrieving sequences  
  Biomart, 34

## S

software  
awk, 43  
Perl, 42  
R, 43