Linköping University

Bayesian Learning

732A46

Project report: Bayesian analysis of DNA microarray data

Author:

Professor:
Mattias Villani

December 22, 2013

Contents

Introduction

Over the past twenty years, DNA microarray technology has grown substantially to measure the expression levels of many genes simultaneously. This technology now allows to record and analyze the expression levels of tens of thousands of genes in many patients, providing large datasets, with a number of features (i.e. genes) much larger than the number of samples (i.e. patients). However, due to different (and sometimes unknown) sources of variation, the analysis of these datasets can prove challenging. In this project, an analysis of such a dataset coming from patients with cancer is performed. More specifically, the expression of the genes regarding a certain 'phenotype' is discussed. The phenotype chosen is the 'cancer stage' of the patients[1], which can take the value 1, 2, 3 or 4. However, there is too few samples from stages 3 and 4 to draw conclusions from these 'populations', and this project will focus on stages 1 and 2, so that the outcome is binary (stage=1 or stage=2). Therefore, the patients in stages 3 or 4 are discarded[2]. The analysis is performed at three different levels of increasing complexity [1], constituting the three chapters of this report.
The first part of this project consists in an attempt to find the genes which are expressed differentially in the different stages. A Bayesian method is used instead of the classic t-test, and the algorithm is run over all the genes to identify the most discriminant ones for this phenotype.
In the second part of this project, a Bayesian hierarchical clustering on the samples is performed, using the variables (i.e. genes) found in the previous part. This part is 'secondary', and is actually just a mean to test the results of the part one. The principles behind the Bayesian hierarchical clustering and the function 'bclust()' used are described.
Finally, the last part chapter of this report focuses on a probit Bayesian linear regression with variables selection. Since the number of variables is too large to run the algorithm with the entire dataset, a 'pre-selection' is performed. This 'pre-selection' consists in the most discriminant genes of the first part and many random genes (in order to see if the algorithm works properly).
All the programs for this project have been done in R.

[1]The variable 'sex' which was chosen first has turned to be too easy to predict given the expression of some genes, since one single gene was enough to classify all the samples.

[2]The methods used in this project can be extended to more than two variables, but it requires some modifications.

Chapter 1

Individual gene analysis

In this chapter, the expression of each gene, given the stage, is analyzed, like in a classic t-test but with a Bayesian perspective. This technique, and most of the developments that follow, comes from a work of Gonen et al. [2]. This is done by comparing two models with the Bayes factors. In the first model, the mean expression μ is the same for both stages, such that $H_0 : \mu_1 = \mu_2 = \mu$, with μ_1 and μ_2 the mean expression for the gene in stage 1 and 2, respectively. In the other model, the mean expression is assumed to be different in both stages, $H_1 : \mu_1 \neq \mu_2$ and the data are then assumed to come from two different distributions. In both case, the data are assumed to be independent and normally distributed (with mean μ for the first model, and with mean μ_1 or μ_2 in the second, depending on the stage).

To perform this 'model selection', the posterior probabilities of each model, given the data y, have to be computed:

$$P(M = i|y) = \frac{P(y|M = i)P(M = i)}{\sum_i P(y|M = i)P(M = i)}$$

with $P(M = i)$ the prior probability of model i, and $P(y|M = i)$ the likelihood. With two models (i=1,2), the posterior odds for model $M = 1$ (i.e. H_0) is then given by:

$$\frac{P(M = 1|y)}{P(M = 2|y)} = \frac{P(y|M = 1)P(M = 1)}{P(y|M = 2)P(M = 2)}$$

where $B_{12} = \frac{P(y|M=1)}{P(y|M=2)}$ is the Bayes factor. Without any prior idea of which model should be chosen, a constant prior is often used, such that $P(M = 1) = P(M = 2) = 1/2$, and the posterior odds ratio is reduced to the Bayes factor. Computing this Bayes factor requires to find the marginal posterior distribution:

$$p(y|M = i) = \int f(y|\theta_i)p_i(\theta_i)d\theta_i$$

with θ_i the parameter vector for model i, and $p_i(\theta_i)$ the prior distribution of the parameters in the model M_i.

To find an expression for the marginal posterior, prior distributions have to be defined for the parameters: μ, σ, $\delta = \mu_1 - \mu_2$ (only for the second model)

and $P(M = i)$.
The hierarchical model used to define these priors is as follow:

$$\frac{\delta}{\sigma}|(\mu, \sigma, M_2) \sim N(\lambda, \sigma_\delta^2)$$

where M_2 is associated with the H_1 hypothesis ($\delta \neq 0$).

$$p(\mu, \sigma) \propto 1/\sigma^2$$

a 'non-informative' prior over μ, σ.

$$P(M = i) = 1/2 \qquad i = 1, 2$$

as discussed previously.

With the prior defined, it's now possible to compute the marginal posterior and to find the expression of the Bayes factor[1]. This (long and tedious) step will not be reproduced in this report and can be found in [3], in the appendix, with $\nu_0 = 0$ so that the prior for the variance is flat. However, the main idea is to integrate the posterior over μ and σ in order to find the marginal posterior for δ, given the data.
This gives:

$$p(\delta|y) \propto \left(1 + \frac{(\delta - (\overline{y_2} - \overline{y_1}))^2}{\nu_n(\frac{1}{n_2} + \frac{1}{n_1})\sigma_n^2}\right)^{(\nu_n+1)/2}$$

which is the form of the pdf (probability density function) of the t-distribution[2] with ν_n degrees of freedom (without the constant factor, which only depends on ν_n), the variable ('x' in the wikipedia page) being $\frac{\delta - (\overline{y_2} - \overline{y_1})}{\sigma_n \sqrt{\frac{1}{n_2} + \frac{1}{n_1}}}$.
So,

$$\frac{\delta - (\overline{y_2} - \overline{y_1})}{\sigma_n \sqrt{\frac{1}{n_2} + \frac{1}{n_1}}}|y \sim t_{\nu_n} \tag{1.1}$$

with:

- $\nu_n = n_1 + n_2 - 2$, the number of degrees of freedom
- $\sigma_n^2 = (s_1^2 * (n_1 - 1) + s_2^2 * (n_2 - 1))/\nu_n$
- $\overline{y_i} = \frac{\sum_{j=1}^{n_i} y_j}{n}$
- $s_i^2 = \frac{\sum_{j=1}^{n_i} (y_j - \overline{y_i})^2}{n_i - 1}$

One can notice that this expression has the same form as the t of the classic t-test with two samples. Therefore, the Bayes factor for testing $H_0 : \delta = 0$ against $H_1 : \delta \neq 0$ becomes [2]:

$$B_{12} = \frac{T_\nu(t|0, 1)}{T_\nu(t|n_\delta^{1/2}\lambda, 1 + n_\delta\sigma_\delta^2)}$$

[1] Note that the Bayes factor does not give an information of 'how good is a model', but just of 'how good is the model in regards to the other'.

[2] 'Student's t-distribution', Wikipedia, http://en.wikipedia.org/wiki/Student%27s_t-distribution, 2013

where $T_\nu(t|a,b)$ represents the non-central t probability density function with the parameters 'location' a, 'scale' $b^{1/2}$ and ν degrees of freedom for the variable t defined by equ. 1.1.

This has been implemented in R in the file 'bayesian_project_1.R'. However, since the pdf of the non-central t distribution has a scale parameter equals to 1 (b=1), the expression has to be slightly modified [2], and becomes $T_\nu(t/b^{1/2}|a/b^{1/2},1)/b^{1/2}$ (which is equals to $T_\nu(t|a,b)$). This gives the following command line in R:

$$BF = dt(t, n_1 + n_2 - 2)/(dt(t/sqrt(postv), n_1 + n_2 - 2, n_c)/sqrt(postv))$$

with:

$$postv = 1 + n_\delta \sigma_\delta^2$$

$$n_c = n_\delta^{1/2} \lambda / (1 + n_\delta \sigma_\delta^2)^{1/2}$$

A Bayes factor for gene i larger than 0 means that the H_0 hypothesis ($\delta = 0$) is 'more likely' than H_1, while, at the opposite, a Bayes factor smaller than 0 supports more H_1 ('two different populations', one for each stage). Since the purpose is to find the genes which are expressed differentially in both stages, the genes kept are those with the smallest Bayes factor (this is done by sorting the genes according to their Bayes factor after running the algorithm).
Before running the algorithm, the hyper-parameters λ and σ_δ have to be specified. λ has been set to 0, since there is (a priory, and for non-expert) no reason to think that genes should generally be more expressed in the first or the second stage. The value of σ_δ has first been fixed to 0.3, but a sensitivity analysis of this hyper-parameter is then performed to see the influence of this value.

The results are shown in figure 1.1. One can see that some genes are associated with a very small Bayes factor, thus giving 'evidence' of a different expression in both stages. However, given that the number of genes is very large (almost 25000), these results have to be watch carefully since it would not be surprising that, over all genes, some of them are associated with small Bayes factors by chance and are in fact 'false positive'. The values of the smallest and largest Bayes factors for other values of σ_δ can be found in appendix.

To analyze the influence of the hyper-parameter σ_δ, the algorithm is run for 40 different values of σ_δ (from 0.05 to 2). For each run, the number of Bayes factors smaller than 0.1 and the number of Bayes factors larger than 2 are computed. The results are shown in figures 1.2 and 1.3. These graphs tend to show that the hypothesis of 'two different populations' is more supported for a value of σ_δ of about 0.5.

```
> source('bayesian_project_1.R')
[1] "smallest Bayes factors:"
  cg21120063   cg22199080   cg22980079   cg06242000   cg24619694   cg23620639
0.0005920472 0.0040394316 0.0045071030 0.0051150804 0.0054688058 0.0061297559
  cg24735937   cg15679095   cg08827391   cg06290096   cg27433088   cg04726200
0.0089529093 0.0113194704 0.0120904222 0.0125754815 0.0149220882 0.0156924242
  cg14142521   cg26884581   cg03662459   cg18003231   cg09705062   cg06771126
0.0163476860 0.0172813976 0.0175953123 0.0207684841 0.0213729497 0.0219904116
  cg11653466   cg11081833
0.0228930702 0.0229382798
[1] "largest Bayes factors:"
cg16790239 cg21182322 cg27383744 cg25023994 cg18042079 cg06765514 cg23267110
  1.726267   1.726267   1.726267   1.726267   1.726267   1.726267   1.726267
cg10348863 cg02273078 cg16358826 cg20252016 cg21351102 cg01866162 cg19324627
  1.726267   1.726267   1.726267   1.726267   1.726267   1.726267   1.726267
cg04301614 cg17080180 cg11059483 cg15166089 cg04713352 cg27553637
  1.726267   1.726267   1.726268   1.726268   1.726268   1.726268
```

Figure 1.1: The smallest and largest Bayes factors computed, with the gene associated to each value (with $\sigma_\delta = 0.3$).

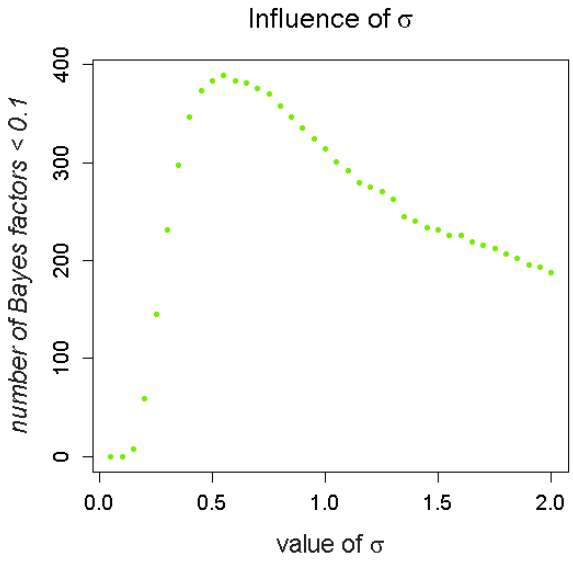

Figure 1.2: Evolution of the number of genes associated with a Bayes factor smaller than 0.1 with σ_δ.

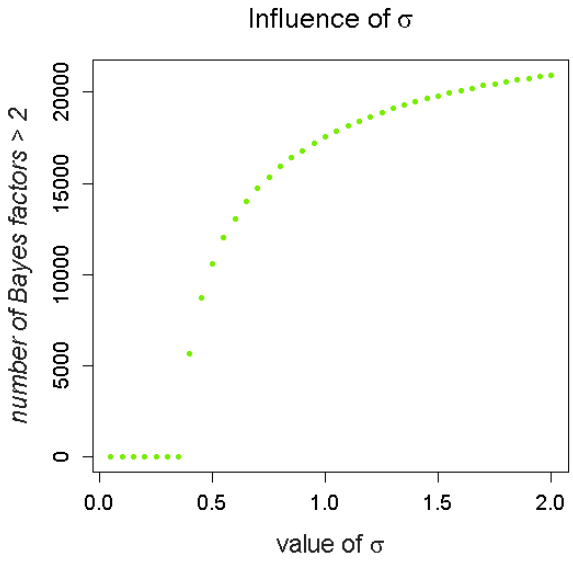

Figure 1.3: Evolution of the number of genes associated with a Bayes factor larger than 2 with σ_δ.

Chapter 2

Bayesian hierarchical clustering

To 'test' the results of the first section, a Bayesian hierarchical clustering can be performed with the genes found. The idea is to see if a hierarchical clustering with the 'most discriminant' genes (i.e. those associated with the smallest Bayes factors) enables to correctly classify the patients of each stage. So, if the expression of the most discriminant genes mainly vary with the stage, the samples should, on the highest level of the tree (i.e. with only two leaves), naturally be regrouped according to their stage (since we perform the hierarchical clustering with the genes that 'code' the stage).

Bayesian hierarchical clustering has many advantages compared to a 'classical' hierarchical clustering approach [4]. Indeed, in a classical approach, a distance metric has to be chosen (which is often difficult to choose) and different distance metric can lead to very different results. In addition, there is no notion of 'uncertainty' for the tree obtained as well as no 'rule' to know where the tree should be cut (i.e. to know the number of clusters to choose given the tree). By considering a probabilistic model to build the tree, it's possible to address these drawbacks.
A bottom-up[1] approach is used, and the clusters are merged according to 'the probability that the data of the two sub-clusters come from one single cluster (i.e. are drawn from the same 'population')'. This probability is given by the posterior probability of the 'merged hypothesis', given the data, which is computed using (surprise...) Bayes rule [4].

In summary, when considering the merge of two clusters D_1 and D_2 to form one single cluster D_{12}, two hypothesis are compared, denoted H_1 and H_2. The first hypothesis H_1 considers one single probabilistic model, from which are drawn all the data of the two clusters:

$$p(x_{12}|H_1) = \int p(D_{12}|\theta)p(\theta|H_1)d\theta$$

[1] This means that the algorithm starts with each data in a cluster and merge two clusters at each step until all the data are in one cluster. On the other hand, the 'top-down' approach considers all the data in one cluster at the beginning and iteratively 'cuts' a cluster in two clusters, and so on until the number of clusters is equal to the number of data.

with x_i, the data in the cluster D_i (so that $x_{12} = x_1, x_2$), θ, the vector of the (unknown) parameters of the model, $p(x_i|\theta)$ the likelihood and $p(\theta|H_1)$ the prior of θ under H_1 (which depends of hyper-parameters).
The second hypothesis, H_2, is that the data come from two different probabilistic models (the two sub-trees). The probability of the data given this hypothesis is simply given by: $p(x|H_2) = p(x_1|T_1)p(x_2|T_2)$ (with $p(x_i|T_i)$ the probability of the data of the tree i under the tree i), i.e. the product of the probabilities of the data of each sub-tree under each sub-tree.
The marginal probability of the data in tree T_{12} can thus be derived:

$$p(x_{12}|T_{12}) = p(H_1)p(x_{12}|H_1) + (1 - p(H_1))p(x_1|T_1)p(x_2|T_2)$$

with $p(H_1)$ the prior probability of H_1, and the posterior probability of the merged hypothesis can be computed as:

$$p(H_1|x_{12}) = \frac{p(H_1)p(x_{12}|H_1)}{p(x_{12}|T_{12})}$$

This posterior probability is used at each iteration to determine which clusters should be merged (the clusters i,j with the highest $p(H_1|x_{ij})$). No distance metric is thus needed, and the resulting tree can be cut at the leaves where the probability $p(H_1|x_{12})$ is less than 0.5. A summary of the algorithm is given in the appendices.

A function 'bclust()', from the 'bclust' package[2], is already implemented in R to perform a Bayesian hierarchical clustering. The program used can be found in the file 'bayesian_project_1.R'. Since the purpose of the previous section was to find the relevant genes to predict the stage, the 'variable selection' available for this function is not used. The description of the parameters of this function and their value is discussed in the appendices.
The resulting tree, keeping the 5 most discriminant genes for the phenotype of interest (i.e. stage), is displayed in figure 2.1. One can see that the classification gives pretty good results (with most of the samples in stage 0 in the left branch and most of the samples in stage 1 in the right branch), but is far from perfect.

If this classification is not perfect, one can assume that the expression, even for the 'most relevant genes', is not so different given the phenotype. Figure 2.2 shows the expression for the most discriminant gene in all the samples. One can see that, as expected, a perfect classification clearly cannot be achieved only with this gene[3] and it's not surprising that a perfect classification cannot be achieved even when considering the 5 most relevant genes[4]. This result proves that the 'stage' is not the only source of variation of the expression, even for the most relevant gene for this phenotype.

[2]For documentation about this package, see [5].

[3]The opposite was observed for the phenotype 'sex', where a perfect classification can be achieved by considering the expression of one of the hundred most relevant genes.

[4]The number of genes could of course be increased but that will increased the chances to 'include other factors' than the stage in the classification.

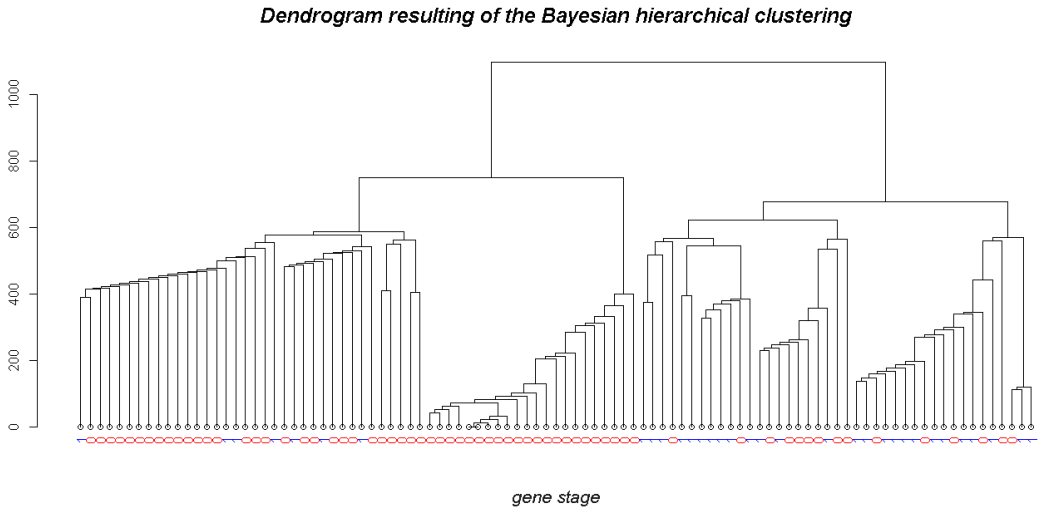

Figure 2.1: Tree obtained with the function 'bclust()', with the 5 genes associated with the smallest Bayes factors (and without variable selection for the hierarchical clustering). In red ('0') the patients in stage 2, in blue ('1') the patients in stage 1.

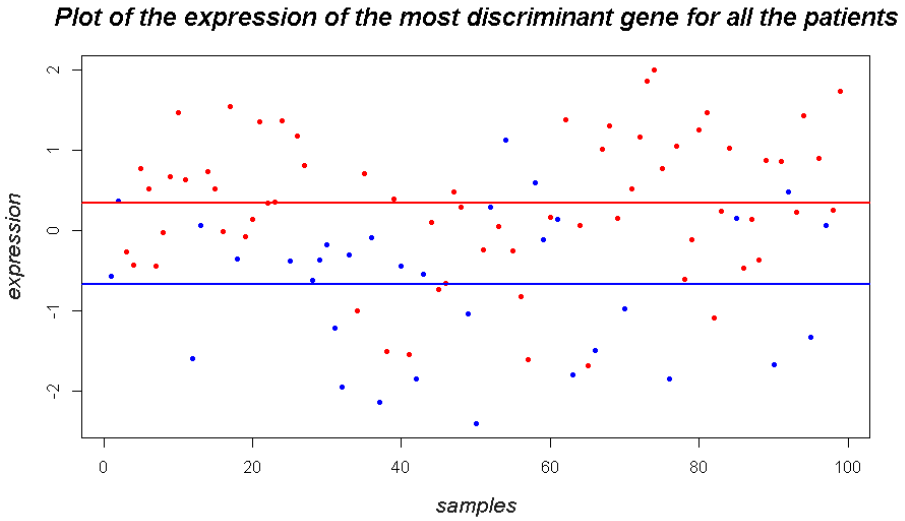

Figure 2.2: Expression of the most discriminant gene for all the patients. In red, those associated with the phenotype 'stage 2', in blue with the phenotype 'stage 1' (the lines are the means for each stage).

Chapter 3

Bayesian probit regression

This final part of the project focuses on a Bayesian probit regression with variables selection, to try to 'predict' the stage given the expression of some genes. The basic idea is to perform a linear regression, the binary variable 'stage' being replaced by a continuous variable α [6], without taking too many features (i.e. genes) since this could lead to overfit. This variable selection part is implemented by defining a vector of indicators, $'Ind'$, of length equals to the number of genes, which elements take value 1 (if the gene is selected) or 0 (if the gene is not selected) [7]. Finding which genes are 'important' for the regression and the coefficient β associated to these genes requires to simulate from the posterior distribution, which is unknown. However, the conditional posterior of these parameters[1] can be derived and therefore a Gibbs sampling can be used.

3.1 Probit model and variable selection

The probit model is defined as:

$$P(y_i = 1|x_i) = \phi(x_i'\beta)$$

with:
y_i $(i = 1, \dots n)$: the stage $(y_i \in 1, 2)$
x_i $(i = 1, \dots n)$: a vector of length p, the components of which are the expressions of the genes
β : a vector of length p, the components of which are the coefficient associated to each gene
$\phi(.)$: the cumulative normal distribution function

This model can be formulated more easily by defining a variable vector α such that $\alpha_i \sim N(x_i'\beta, 1)$ and $\alpha_i > 0$ if $y_i = 1$ and $\alpha_i \leq 0$ if $y_i = 0$. This variable thus transform the probit model into a linear model $(\alpha_i = x_i'\beta + \epsilon)$. However, since the variable α is unknown, a Gibbs sampling must be used to simulate form the posterior $p(\alpha, \beta|y, x)$. This Gibbs sampling iteratively simulate from the conditional posteriors $p(\beta|\alpha, y, x)$ and $p(\alpha|\beta, y, x)$.

[1]The term 'conditional posterior' means the posterior distribution a parameter given the other parameters. The conditional posterior of each parameter is requested for Gibbs sampling.

To include a variable selection process in this model, a new vector of binary variable Ind is introduced with $Ind_j = 1$ $(j = 1, \dots p)$ if the gene is used in the regression and $Ind_j = 0$ if it is not[2]. Given a simple Bernoulli model as prior, the conditional of $p(Ind|\alpha, \beta, x)$ can be derived[3] and this variable can easily be introduced in the conditional posteriors of the other parameters by restricting the coefficients β_j (and thus the data matrix) used to those associated with an indicator Ind_j equals to 1. In practice, the indicators Ind_j are drawn one by one.

An approach combining the idea for the probit regression and for variables selection can be found in the work of Lee K. E. [8], from which this part of the project is drawn.

3.2 Prior and conditional posterior

The priors of the model are the following:

$$p(Ind_j = 1) = Bernoulli(\pi_j)$$

with $0 \leq \pi_j \leq 1$ $(j = 1, \dots p)$ the prior probability of including the gene j in the model. The probabilities of selection for the different genes are assumed to be independent.

$$\beta_{Ind} \sim N(0, c(X'_{Ind} X_{Ind})^{-1})$$

where β_{Ind} is the vector of the coefficients β_j restricted to the coefficients associated to an indicator $Ind_j = 1$ (the others coefficients are thus zero) and similarly X_{Ind} is the data matrix obtained by keeping only the genes (columns) associated to an indicator $Ind_j = 1$. The scalar c is a (positive) hyper-parameter associated with the scale and is first fixed to 100 [9].

Given these priors, the conditional posteriors can be computed. The steps to derive the expressions of the conditional posteriors can be found in the appendix of [8] and will not be reproduced here. The results are the following:

$$p(Ind_j = 1|\alpha, Ind_{k \neq j}) = \frac{1}{1 + \frac{1-\pi_j}{\pi_j}\sqrt{(1+c)} * exp(-1/2 * (S(Ind^0) - S(Ind^1)))}$$

where:

- $Ind^0 = (Ind_1, \dots, Ind_j = 0, \dots, Ind_p)$
- $Ind^1 = (Ind_1, \dots, Ind_j = 1, \dots, Ind_p)$
- $S(Ind^l) = \alpha'\alpha - \frac{c}{1+c}\alpha X_{Ind^l}(X'_{Ind^l} X_{Ind^l})^{-1} X'_{Ind^l}\alpha$, with X_{Ind}, the restricted data matrix (see above). Note this posterior is obtained by integrating over β, which makes the algorithm faster and doesn't change the results [8].

[2] The genes include in the model can change at each iteration.

[3] Note that this posterior is only used to compute and compare the probabilities $p(Ind_j = 1)$ and $p(Ind_j = 0)$.

$$\beta_{Ind}|\alpha \sim N(\mu_n, V_n)$$

with:

- $\mu_n = (1 + c^{-1})^{-1}(X'_{Ind}X_{Ind})^{-1}X'_{Ind}\alpha$
- $V_n = (1 + c^{-1})^{-1}(X'_{Ind}X_{Ind})^{-1}$

(Note that these expressions are the same than given in the course [10], slide 9/12, with σ fixed to 1, $\mu_0 = 0$ and $\Omega_0 = \frac{1}{c}X'_{Ind}X_{Ind}$.)

$$\alpha_i|(\beta, x_i, Y = 1) \sim N(x'_i\beta, 1) \text{ truncated to } [0, +\infty]$$
$$\alpha_i|(\beta, x_i, Y = 0) \sim N(x'_i\beta, 1) \text{ truncated to } [-\infty, 0]$$

The influence of Ind on α is indirect since Ind restrict the number of β coefficient and therefore has a big influence on the mean $x'_i\beta$ of the distribution from which *alpha* is drawn.

3.3 Algorithm

The Gibbs sampler with n iterations implemented in R works as follow:

1. Initialize value for Ind, $\beta|Ind$ and $\alpha|\beta$
2. for t = $1 : n_{iter}$, do
   for u = 1:p, do
   Simulate from the conditional posterior of $Ind_u|Ind_{v\neq u}, \alpha^{(t-1)}$
   Simulate from the conditional posterior of $\alpha|X^t_{Ind}\beta^{(t-1)}$
   Simulate from the conditional posterior of $\beta|Ind^t\alpha^{(t)}$
3. After the n iterations, 'burn' the x first iterations (since the Gibbs sampler need many iterations to converge) and consider the (n-x) remaining iterations as draws from the posterior of the model.

3.4 Results

The program implemented is explained in appendices. However, the time complexity[4] of the program strongly limits the number of genes that can be used. This high time complexity is mainly due to the 'for loop' over the parameters and the many matrix inversions which are performed for one iteration. Because of this, the time complexity is not linear with the number of genes in input.
In order to test the algorithm, the genes kept as features are composed of the ten genes associated with the smallest Bayes factors and fifty random genes chosen among the others. The number of iterations was fixed to 25000 where the first 10000 are burnt, as in [8][5]. The hyper-parameter c was fixed to 100

[4] And, to be honest, the fact that my program was working properly only two days before the deadline. I first considered an implementation with 'RStan' but problems arose for the variable selection part because it turns out that 'RStan' doesn't support discrete parameters yet.

[5] Normally, a convergence analysis should be performed in order to determine the number of iterations required for the convergence of the algorithm.

and the prior probability of selection of a gene π_i (which is of course also a hyper-parameter) was set to 1/10. The frequency of apparition (i.e. the number of time Ind_i was equal to one divided by the number of iterations) for all the genes used was printed.
Results for the frequency of 'apparition' of the genes are shown in figure 3.1.

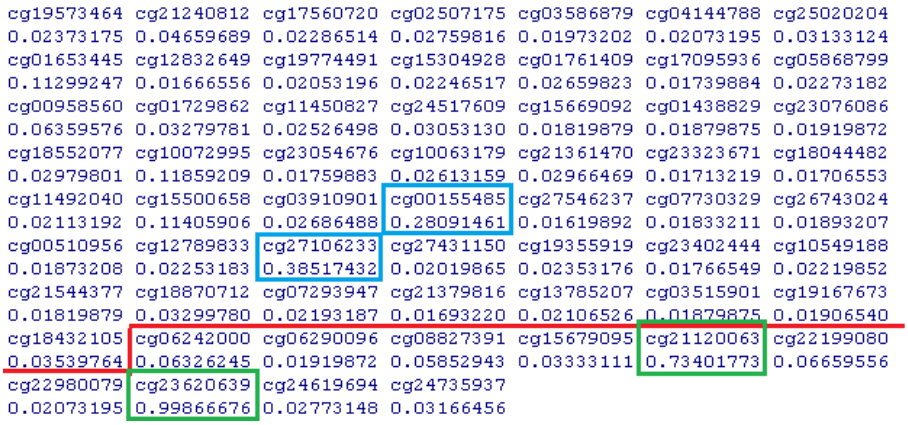

Figure 3.1: Frequency of 'selection' of each gene for the regression (with hyper-parameters $\pi_j = 1/10$ and $c = 100$). The red line divides the 50 random genes of the 10 genes associated with the smallest Bayes factors (note that these genes are not in the same order than their Bayes factors since the function 'which()' used in R ordered the genes according to their 'name'). In green, the two genes with small Bayes factors and high frequency of selection for the regression model. In blue, the two 'random' genes with a quite important frequency of selection.

One can see that two of the ten genes (in green in figure 3.1) with small Bayes factors have a very high frequency of selection for the regression model (in green). These two genes $'cg211263'$ and $'cg23020639'$ correspond to the first (i.e. smallest) and the 6th Bayes factors, respectively. It should be noted that the frequency for one gene depends on the other genes given as inputs, and since many of these genes are chosen randomly, the results between two simulations can lead to different frequencies for the same gene. In addition to these genes, other 'random' genes seem to be useful (in blue in figure 3.1) since their frequency of selection is quite important.
It's not surprising that all the genes with a small Bayes factor are not associated with a high frequency of selection since the criterion to be relevant for a regression task (with many genes) is quite different from the criterion where one single gene is considered. Two or more genes may be relevant but contain the same 'information' and thus only one of them is selected.
Note that only the frequency of each gene is showed here, but this frequency doesn't express the 'importance' of the gene in the regression model. Indeed, a gene with a high probability of selection can be associated to a small coefficient β_j. The mean coefficient β_j for gene j can be visualized by first removing all the iteration where Ind_j is equal to zero (thus associated with a β_j equals to zero too) and then perform the mean.

These results can be used to 'classify' a new sample i based on the genes expression (without knowing the true stage) by considering all the draws (one for each iteration) after the convergence of the algorithm (i.e. the draws which are not 'burnt'). For each draw, the β^t (where 't' denotes the iteration 't') are used to compute $x_i'\beta$ which is then used as the mean to draw α[6]. Finally, the cumulative normal distribution function is applied on α and gives the probability of $stage = 1$. The mean (over all the draws) is then used to find the actual probability than $y = 1$.
These steps can be summarized as follow:

1. For t=10000:n_{iter}, do

   Take β^t and compute $m = x_i'\beta$

   Draw $\alpha_i \sim N(m, 1)$

   Compute $p^t(i = 1) = \phi(\alpha_i)$

2. Compute the mean over t of p^t

3. Result: probability for i to be of stage 1

Note that the influence of Ind at each iteration is indirect via the β since the β_j associated to an indicator Ind_j equals to zero is null.

3.5 Sensitivity analysis

The hyper-parameters of the model are π_j, the prior probability of selection of gene j, and c, the hyper-parameter associated with the 'scale'.
The probability π_j plays a key role in the frequency of selection of gene j. In theory, we could define π_j to be larger for the genes which are more likely to be useful for the regression model. For instance, based on the probability of the results of section 1, we could define π_j to be a decreasing function of the Bayes factor of gene j (so that the smaller the Bayes factor, the higher the prior probability). However, to see the 'global' influence of this hyper-parameter, it's kept the same for every gene (i.e. $\pi_j = k \ \forall j$). Two simulations with different values of π_j have been performed. The results are shown in figures 3.2 and 3.3.

One can see that the mean frequency of selection of a gene increases with π. Indeed, the resulting mean are shown in array 3.1. Note that these results can be affected by the fact that the genes are not the same in each simulation[7]. But the global tendency should not be different, since this results corresponds to the one of Lee K. [8].

	$\pi_j = 1/5$	$\pi_j = 1/10$	$\pi_j = 1/20$
sum of the frequencies obtained	5.739	4.141	2.828
mean frequency obtained	0.0956	0.0690	0.0471

Table 3.1: Influence of π on the mean frequency obtained for the genes.

[6]For me draw α or just used $x_i'\beta$ should lead to the same classification but can change the probability, i.e. the uncertainty.

[7]Unfortunately, I noticed that too late to perform new simulations with the same genes.

```
cg14225485 cg21061811 cg04703844 cg12559031 cg27222589 cg24866437 cg11613875
0.04779681 0.04166389 0.04933004 0.03819745 0.04939671 0.04173055 0.04619692
cg18469326 cg06590610 cg20636078 cg25018329 cg13451483 cg13401681 cg11398680
0.06099593 0.04986334 0.04106393 0.03759749 0.04573028 0.04313046 0.06272915
cg08480367 cg16404106 cg12422968 cg03231024 cg04278905 cg21844956 cg09243900
0.05699620 0.03833078 0.05092994 0.83414439 0.03999733 0.05926272 0.05499633
cg03909500 cg02901159 cg00896220 cg26843008 cg26207503 cg16604136 cg04993257
0.04026398 0.04973002 0.03499767 0.05726285 0.04886341 0.04326378 0.04246384
cg11646757 cg14404298 cg09800519 cg08831744 cg08081725 cg20041381 cg05709923
0.04359709 0.04093060 0.04479701 0.04479701 0.04359709 0.04093060 0.04179721
cg10501128 cg15541315 cg13619408 cg17020834 cg15407570 cg06218044 cg21509023
0.04879675 0.04126392 0.04339711 0.03819745 0.04773015 0.06539564 0.03706420
cg04310824 cg26426582 cg12619162 cg22552669 cg27144395 cg06421800 cg04478795
0.04773015 0.05466302 0.10052663 0.04426372 0.04499700 0.03793080 0.04219719
cg05275231 cg06242000 cg06290096 cg08827391 cg15679095 cg21120063 cg22199080
0.04686354 0.14792347 0.04159723 0.14452370 0.04959669 0.98140124 0.10852610
cg22980079 cg23620639 cg24619694 cg24735937
0.03879741 1.00000000 0.03633091 0.04213052
```

Figure 3.2: Frequency of 'selection' of each gene for the regression, with hyperparameter $\pi_j = 1/5$ (and c=100).

```
 cg07656391  cg02742971  cg14718680  cg21020082  cg05671350  cg00425792
0.009066062 0.009999333 0.022198520 0.008866076 0.008266116 0.008866076
 cg15185286  cg08715135  cg06383088  cg11385473  cg21256649  cg11822772
0.009866009 0.048863409 0.008999400 0.010265982 0.011132591 0.012532498
 cg22767079  cg06615861  cg25033144  cg00400263  cg24319545  cg09890200
0.009399373 0.010332644 0.007466169 0.015798947 0.026864876 0.007266182
 cg11047295  cg02945646  cg24995381  cg16185365  cg02064106  cg01169610
0.024598360 0.007266182 0.036330911 0.013799080 0.015598960 0.008266116
 cg15615135  cg01531431  cg10523671  cg18589858  cg25928444  cg23355725
0.018265449 0.009199387 0.009399373 0.044130391 0.009266049 0.030797947
 cg08293367  cg19630506  cg15361750  cg04963951  cg16544272  cg15467759
0.007332844 0.010799280 0.007532831 0.051596560 0.014265716 0.009466036
 cg11378484  cg00876704  cg11080718  cg01036779  cg06941093  cg20022122
0.008199453 0.008999400 0.026864876 0.014665689 0.016398907 0.019132058
 cg24901042  cg10035922  cg19307060  cg24884084  cg25294646  cg17791936
0.011532564 0.009999333 0.009332711 0.007732818 0.010332644 0.008732751
 cg02551396  cg24182328  cg06242000  cg06290096  cg08827391  cg15679095
0.009066062 0.007532831 0.026998200 0.010399307 0.017932138 0.018998733
 cg21120063  cg22199080  cg22980079  cg23620639  cg24619694  cg24735937
0.947536831 0.033597760 0.011265916 0.985600960 0.012865809 0.020331978
```

Figure 3.3: Frequency of 'selection' of each gene for the regression, with hyperparameter $\pi_j = 1/20$ (and c=100).

The second hyper-parameter is the scalar c (the 'scale'), which was set to 100 in the previous simulations. This parameters could, I think, be replaced by a parameter σ which would follow a distribution (an Inverse-chi-squared for example) and could be included in the model. A try has been tested but the result were similar to those found by fixing c. Two tests, the results of which are shown in figures 3.4 and 3.5, have been performed with $c = 10$ and $c = 50$. However, the influence of c is not clear based on these tests, and according to Lee K. [8] the influence is small.

```
cg27600794 cg01062029 cg08705994 cg07934026 cg18403396 cg17870792 cg23164327
0.05592960 0.05259649 0.05599627 0.05186321 0.05179655 0.05079661 0.05786281
cg19162106 cg17710288 cg20878850 cg11065518 cg08002996 cg06350796 cg08260891
0.04859676 0.04886341 0.06099593 0.05666289 0.04786348 0.08479435 0.07446170
cg16090392 cg16209630 cg22110261 cg25755261 cg10348863 cg00309056 cg09637363
0.05046330 0.08226118 0.05932938 0.07179521 0.09659356 0.06799547 0.04733018
cg26848248 cg20322876 cg19751300 cg19279529 cg18138500 cg26601286 cg16779976
0.04679688 0.04446370 0.04686354 0.06072928 0.04993000 0.05912939 0.04499700
cg11979312 cg11609366 cg25854162 cg19427472 cg25203980 cg11052068 cg06285340
0.04593027 0.04813012 0.05392974 0.05512966 0.05746284 0.05459636 0.07419505
cg16148403 cg11438428 cg25092283 cg03846288 cg19320612 cg24305835 cg09736162
0.05052996 0.05072995 0.05799613 0.07312846 0.12479168 0.04719685 0.04933004
cg10226546 cg16797831 cg07827038 cg27573888 cg15933546 cg15645784 cg05798712
0.04593027 0.04559696 0.05052996 0.05172988 0.06339577 0.05766282 0.04833011
cg20164558 cg06242000 cg06290096 cg08827391 cg15679095 cg21120063 cg22199080
0.05106326 0.26811546 0.07252850 0.16472235 0.09219385 0.66722219 0.26378241
cg22980079 cg23620639 cg24619694 cg24735937
0.07999467 0.67748817 0.08126125 0.09066062
```

Figure 3.4: Frequency of 'selection' of each gene for the regression, with hyper-parameter c=10 (and $\pi_j = 1/10$).

```
cg24654350 cg17183991 cg18267381 cg15890111 cg17787710 cg21304187 cg14984307
0.02326512 0.04386374 0.03119792 0.03919739 0.02306513 0.03013132 0.04553030
cg03573747 cg22101147 cg10485472 cg25177139 cg25018329 cg25953146 cg06543018
0.02359843 0.02639824 0.02946470 0.03099793 0.02259849 0.02813146 0.02833144
cg00226923 cg00107488 cg07310951 cg11291200 cg23029494 cg13535217 cg05947740
0.02393174 0.02686488 0.02333178 0.02346510 0.02459836 0.11345910 0.02306513
cg08680102 cg23756272 cg14279899 cg09920632 cg19972859 cg11935147 cg09194436
0.02706486 0.03639757 0.02919805 0.02753150 0.02739817 0.02886474 0.02926472
cg07293947 cg20367961 cg24222435 cg20427865 cg14572111 cg10173075 cg08865050
0.03373108 0.02306513 0.02233184 0.02819812 0.09412706 0.02633158 0.02193187
cg11373746 cg14039952 cg06688396 cg09391966 cg26125443 cg24158546 cg21802726
0.02386508 0.02833144 0.02346510 0.03133124 0.02646490 0.03779748 0.02406506
cg07073964 cg02449608 cg20804555 cg18338296 cg19382714 cg00645579 cg00464269
0.02613159 0.02219852 0.02479835 0.05826278 0.03059796 0.03013132 0.02719819
cg01599709 cg06242000 cg06290096 cg08827391 cg15679095 cg21120063 cg22199080
0.02226518 0.08772748 0.02586494 0.04606360 0.04413039 0.94547030 0.10719285
cg22980079 cg23620639 cg24619694 cg24735937
0.02713152 0.98073462 0.03359776 0.04419705
```

Figure 3.5: Frequency of 'selection' of each gene for the regression, with hyper-parameter c=50 (and $\pi_j = 1/20$).

Finally, the variance of the truncated normal for the α could also, for me,

be studied[8].

[8]But I'm not sure about the influence, if any, of this parameter and didn't find any work where it was done.

Chapter 4

Conclusion and further work

In this project, a analysis of DNA microarray data regarding a certain phenotype (the 'stage') has been performed at different levels, using some Bayesian methods.
First each gene was considered individually to see if any difference in the expression could be observed between the two stages. To do so, the Bayes factor comparing the hypothesis of a single 'distribution' and the hypothesis of a distribution for each stage was computed. Even if some genes are associated with very small Bayes factors, these results must be taken carefully because of the large number of gene (which increases the probability that some genes are associated with small Bayes factors 'by chance') and because of the influence of the hyper-parameter on the results.
In the second chapter, the 'most relevant' genes found in the first part were selected to perform a Bayesian hierarchical clustering. This leads to a quite satisfying tree, which that the phenotype 'stage' is one of the main factor influencing these genes for our data (note that it doesn't reject the hypothesis that it's only by chance).
In the final part of this project, a Bayesian logit regression with variable selection was performed, to try to 'model' the stage based on the gene expression. However, because of the time complexity of the program, the number of genes considered was quite restricted, which limits the use of the results. A (limited) sensitivity analysis was also performed in order to study the influence of the hyper-parameters.

In order to test the model built in the chapter 3, new data would be needed[1]. Other regression models could also have been considered, and then compared with a Bayes factor.

The purpose of this project was to see how Bayesian methods could be used to analyze data at different levels, and if focuses more on the methods (and

[1]The data could also have been 'split' in two, one for the learning part and one for the test part, but this limit the training part or use 'time-consuming' methods which requires to run the algorithm many times.

maybe not enough on the results). Hence, in order to test these three levels (and thus have a broad view of Bayesian learning applications), the results for each part have been analyzed quite fast (the sensitivity analysis, for instance, should be improved), and further analysis should be performed.

Appendices

Appendix A

Individual gene analysis

A.1 Influence of the hyper-parameter σ_δ on the values of Bayes factors

```
> source('bayesian_project_1.R')
[1] "smallest Bayes factors:"
cg21120063 cg22199080 cg22980079 cg06242000 cg24619694 cg23620639 cg24735937
 0.1436176  0.2294672  0.2357835  0.2433098  0.2473896  0.2545194  0.2797992
cg15679095 cg08827391 cg06290096 cg27433088 cg04726200 cg14142521 cg26884581
 0.2967760  0.3017393  0.3047443  0.3181889  0.3222630  0.3256148  0.3302250
cg03662459 cg18003231 cg09705062 cg06771126 cg11653466 cg11081833
 0.3317340  0.3459810  0.3485114  0.3510427  0.3546513  0.3548293
[1] "largest Bayes factors:"
cg16790239 cg21182322 cg27383744 cg25023994 cg18042079 cg06765514 cg23267110
  1.104536   1.104536   1.104536   1.104536   1.104536   1.104536   1.104536
cg10348863 cg02273078 cg16358826 cg20252016 cg21351102 cg01866162 cg19324627
  1.104536   1.104536   1.104536   1.104536   1.104536   1.104536   1.104536
cg04301614 cg17080180 cg11059483 cg15166089 cg04713352 cg27553637
  1.104536   1.104536   1.104536   1.104536   1.104536   1.104536
```

Figure A.1: The values of the smallest and largest Bayes factors computed with $\sigma_\delta = 0.1$).

```
> source('bayesian_project_1.R')
[1] "smallest Bayes factors:"
  cg21120063   cg22199080   cg22980079   cg06242000   cg24619694   cg23620639
0.0005920472 0.0040394316 0.0045071030 0.0051150804 0.0054688058 0.0061297559
  cg24735937   cg15679095   cg08827391   cg06290096   cg27433088   cg04726200
0.0089529093 0.0113194704 0.0120904222 0.0125754815 0.0149220882 0.0156924242
  cg14142521   cg26884581   cg03662459   cg18003231   cg09705062   cg06771126
0.0163476860 0.0172813976 0.0175953123 0.0207684841 0.0213729497 0.0219904116
  cg11653466   cg11081833
0.0228930702 0.0229382798
[1] "largest Bayes factors:"
cg16790239 cg21182322 cg27383744 cg25023994 cg18042079 cg06765514 cg23267110
  1.726267   1.726267   1.726267   1.726267   1.726267   1.726267   1.726267
cg10348863 cg02273078 cg16358826 cg20252016 cg21351102 cg01866162 cg19324627
  1.726267   1.726267   1.726267   1.726267   1.726267   1.726267   1.726267
cg04301614 cg17080180 cg11059483 cg15166089 cg04713352 cg27553637
  1.726267   1.726267   1.726268   1.726268   1.726268   1.726268
```

Figure A.2: The values of the smallest and largest Bayes factors computed with $\sigma_\delta = 0.3$).

```
> source('bayesian_project_1.R')
[1] "smallest Bayes factors:"
  cg21120063   cg22199080   cg22980079   cg06242000   cg24619694   cg23620639
7.706501e-05 9.900286e-04 1.144135e-03 1.352052e-03 1.476690e-03 1.716306e-03
  cg24735937   cg15679095   cg08827391   cg06290096   cg27433088   cg04726200
2.825404e-03 3.844611e-03 4.191755e-03 4.413688e-03 5.523224e-03 5.899631e-03
  cg14142521   cg26884581   cg03662459   cg18003231   cg09705062   cg06771126
6.224263e-03 6.693670e-03 6.853239e-03 8.512902e-03 8.838207e-03 9.173375e-03
  cg11653466   cg11081833
9.668448e-03 9.693400e-03
[1] "largest Bayes factors:"
cg16790239 cg21182322 cg27383744 cg25023994 cg18042079 cg06765514 cg23267110
  2.549508   2.549508   2.549508   2.549508   2.549508   2.549508   2.549509
cg10348863 cg02273078 cg16358826 cg20252016 cg21351102 cg01866162 cg19324627
  2.549509   2.549509   2.549509   2.549509   2.549509   2.549509   2.549509
cg04301614 cg17080180 cg11059483 cg15166089 cg04713352 cg27553637
  2.549509   2.549509   2.549509   2.549510   2.549510   2.549510
```

Figure A.3: The values of the smallest and largest Bayes factors computed with $\sigma_\delta = 0.5$).

```
> source('bayesian_project_1.R')
[1] "smallest Bayes factors:"
  cg21120063   cg22199080   cg22980079   cg06242000   cg24619694   cg23620639
3.119779e-05 6.053066e-04 7.155351e-04 8.678542e-04 9.609306e-04 1.143179e-03
  cg24735937   cg15679095   cg08827391   cg06290096   cg27433088   cg04726200
2.031806e-03 2.897513e-03 3.200810e-03 3.396693e-03 4.396816e-03 4.743238e-03
  cg14142521   cg26884581   cg03662459   cg18003231   cg09705062   cg06771126
5.044640e-03 5.484536e-03 5.635123e-03 7.230150e-03 7.548459e-03 7.878210e-03
  cg11653466   cg11081833
8.368485e-03 8.393294e-03
[1] "largest Bayes factors:"
cg16790239 cg21182322 cg27383744 cg25023994 cg18042079 cg06765514 cg23267110
  4.795828   4.795828   4.795828   4.795829   4.795829   4.795829   4.795829
cg10348863 cg02273078 cg16358826 cg20252016 cg21351102 cg01866162 cg19324627
  4.795830   4.795830   4.795830   4.795830   4.795830   4.795831   4.795831
cg04301614 cg17080180 cg11059483 cg15166089 cg04713352 cg27553637
  4.795831   4.795831   4.795831   4.795831   4.795831   4.795831
```

Figure A.4: The values of the smallest and largest Bayes factors computed with $\sigma_\delta = 1$).

Appendix B

Bayesian hierarchical clustering

B.1 Algorithm

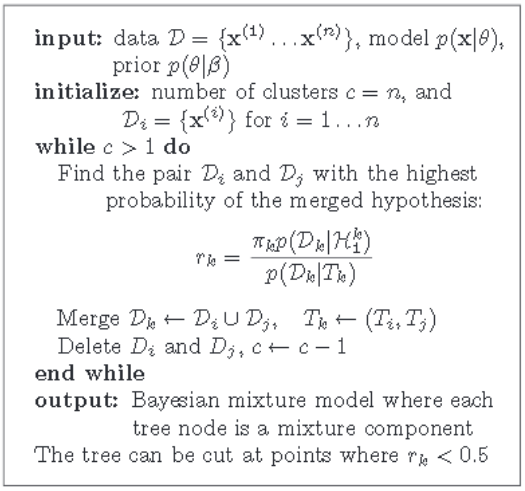

Figure B.1: Algorithm of the Bayesian hierarchical clustering, from [4].

B.2 'bclust()' function

The function 'bclust()' assume a Bayesian linear model for clustering [5], so that:

$$y_{vct} = \mu + \delta_v \gamma_{vc} \theta_{vc} + \eta_{vct}$$

Without variable selection (which is performed in the first section), the model becomes:

$$y_{vct} = \mu + \gamma_{vc} \theta_{vc} + \eta_{vct}$$

where :

- the indexes v, c and t corresponds to 'variable', 'cluster' and 'individual', respectively
- μ is a constant for all variable
- η_{vct} represents the noise (independent of θ_{vct})
- γ_{vc} is a binary variables (of value 0 or 1)
- θ_{vc} is a continuous variable modeled by a Gaussian distribution with mean zero

The figure B.2 is an attempt to represent, via an (invented) example, these parameters for 3 different variable (in grey) where the (simplified) 'boxplot' of these variables for all the samples. The horizontal black line represent the media (i.e. all variables are 'median-centered') and the red horizontal line represent the first parameter μ, which is the same for all variable. In green, the effects of θ_{vc} are represented, without this effect for the first variable because θ_{vc} is equal to zero (no difference can be observed between the two clusters for this variable). Finally, in blue, the effects of the noise are represented.

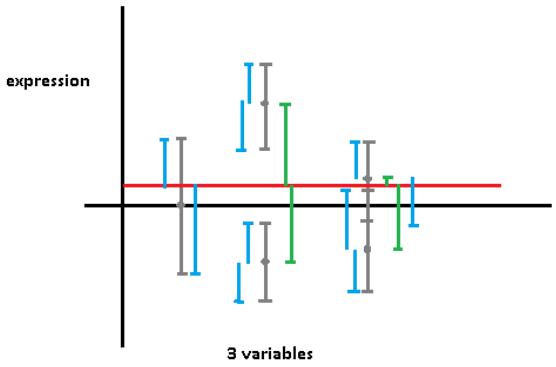

Figure B.2: Representation of the variables of the linear model used by the function 'bclust()'.

The binary variable γ_{vc} has simple Bernoulli model with probabilities p, which represents the probability for a variable to find a different mean for the two clusters.

The arguments taken by the function, without variable selection, are : (log(var(e)),log(var(n)),log(var(theta
with

- var(e) and var(n) are the variance for replicated data and for the noise[1].
- var(theta) represents the variance for the variable θ ans is set to 0.2

[1]Actually I didn't really understood the explanations for these parameters since var(n) is set to zero for unreplicated data. However, after some tests, it seems to have little influence on the results.

- μ is set to zero (mean = median)
- p is set to 1, so that for each variable a different mean could be observed for the two clusters (since this is the argument to select the variables), and so logit(p) is set to 100

So that the final parameters give are (log(0.2), -50, log(0.2), 0, 100).
More details for this function can be found in [5].

Appendix C

Bayesian probit regression

C.1 Description of the program

The hyper-parameters of the model are the prior probability of selection of each gene 'prob_ind' (which is π_j), fixed to be the same for all genes, and c. The Gibbs sampling is composed of three parts: the simulation of Ind, the simulation of α and the simulation of β.
The simulations for Ind is the most difficult part. Basically, at each iteration, it computes, for each gene (second for loop), $S(Ind^0)$ and $S(Ind^1)$ and then compute the probability $p(Ind_j = 1)$. To do so, the data matrix X_{Ind} which contains only the column associated to non-zero Ind_j has to be adapted. Hence, is the gene j was selected at the past iteration ($Ind[j, i-1 = 1)$, then $S(Ind^1)$ can be computed without changing X_{Ind} but the column corresponding to the gene j must be removed for computing $S(Ind^0)$. The opposite is true when the gene was not selected at the past iteration (i.e. X_ind has to be adapted to compute $S(Ind^1)$).
Once $p(Ind_j = 1)$ is computed, a value is drawn for Ind_j, and the matrix X_{Ind} is adapted for the next gene if the value drawn is different from the past iteration.
The simulations for α and β are quite evident, since it's only a draw of a (truncated for α) normal distribution. The 'if()' statements are just there to deal with the case where Ind only contain zero or just one non-null element.

Bibliography

[1] Baldi P., Long A. D. (2001) *A Bayesian Framework for the Analysis of Microarray Expression Data: Regularized t-Test and Statistical Inferences of Gene Changes.* Bioinformatics 17 (6): 509-519.

[2] Gonen M., Johnsony W. O., Lu Y. and Westfall P. H. (2005) *The Bayesian two-sample t-test.* Memorial Sloan-Kettering Cancer Center, Dept of Epidemiology & Biostatistics.

[3] Fox R. J., Dimmic M. W. (2006) *A two-sample Bayesian t-test for microarray data.* BMC Bioinformatics, 7:126

[4] Heller K. A., Ghahramani Z. (2005) *Bayesian Hierarchical Clustering.* Proceedings of the 22nd International Conference on Machine Learning

[5] Davison A. C., Nia V. P. (2012) *High-Dimensional Bayesian Clustering with Variable Selection: The R Package bclust.* Journal of Statistical Software, Volume 47, Issue 5

[6] Villani M. (2013) *Bayesian Learning - Lecture 8: 'Random number generation, Monte Carlo simulation and Gibbs sampling'.* LiU course: 732A46, Bayesian Learning

[7] Villani M. (2013) *Bayesian Learning - Lecture 12: 'Bayesian variable selection and Model checking'.* LiU course: 732A46, Bayesian Learning

[8] Lee K. E. (2004) *Bayesian models for DNA microarray data analysis.*

[9] Smith M. and Kohn R. (1997) *Nonparametric Regression Using Bayesian Variable Selection.* Journal of Econometrics 75, 317–344.

[10] Villani M. (2013) *Bayesian Learning - Lecture 5: 'Normal model with conjugate prior, The linear regression model, Regression with binary response'.* LiU course: 732A46, Bayesian Learning