Photovoltaic System Reconfiguration strategy for mismatch condition

Dafang Zhao
Nara Institute of Science and Technology, Dependable System Lab

July 2018

1 Introduction

As the world of fossil energy constantly exhausted and the increasingly serious environmental pollution, the research and utilization of renewable energy and green energy have become maintain necessary means of survival and development of the human. Photovoltaic (PV) energy received significant attention since it has unlimited energy and easy to be scaled up. Thanks to extensive technology and research on photovoltaic energy generation, large scale photovoltaic energy generation system have been deployed into many practical application.

2 Related work

2.1 Lung adenocarcinoma papers

Here, we will mainly discuss 3 previous works about LUAD and some work on Maximum Likelihood clustering techniques. Starting with the work of Hayes et al.[**hayes**], whom put in evidence from DNA microarrays data, a total cohort of 231 LUAD patients and 2,553 reliable genes, the presence of 3 clusters. These clusters were found using consensus clustering[**cons**], the genes used came from centroids comparison and gene enrichment analysis. Although this study focused on survival rates, which is not our case, it managed to find strong evidences concerning the existence of three distinct LUAD subtypes named Bronchoid, Magnoid and Squamoid having different gene expression profiles and clinical relevance. [**hayes**] also managed to identify sets of genes strongly related to each found subtypes, these genes will be taken in consideration in our study. The second important study is Wilkerson et al.[**wilk**], aimed to have a better understanding of the LUAD molecular pathogenesis in order to find better targets for chemo-treatments. The study focused on DNA alteration, DNA methylation and mutation occuring in LUAD. They also used cohort wise centroids and ConsensusClusterPlus[**consensusPlus**] for a total of 462 patients (GSE data set). They confirmed the clinical differences between the molecular subtypes, showing that the survival rate and therapy response was differing according to each subtype. The last study we will discuss is the one from TCGA[**TCGA**]. This study mainly focused on the relation between the morphological classification and the gene expression profiles, renaming the 3 known subtypes as Proximal-Inflammatory (PI, former Squamoid), Proximal-proliferative (PP, former Magnoid)and Terminal Respiratory Unit (TRU, forner Bronchoid). They used RNA-seq data as well, data that we will use in our study (TCGA data set), 230 patients and 22k+ genes. They used RNA-seq data to study mutations in LUAD and they compared their results of subtype detection with [**wilk**], finding high correlation between respective clusters.

2.2 Hierarchical Maximum Likelihood algorithms

We want to present here some existing methods, implementing a maximum likelihood approach. Note that our desire is to implement a fully Bayesian approach, which differs slightly, furthermore we have to deal with different categories of data. Castro et al.[**castro**], described two different algorithms for clustering purpose using maximum likelihood estimation and MCMC chains. The first algorithm, called Agglomerative Likelihood Tree (ALT), looks for the Maximum Likelihood Tree (MLT) after computing a measure of similarity based on maximum likelihood estimates paiwise. The MLT obtained by this algorithm will be binary structured. The second algorithm, called MLT estimator, takes a more global approach. Searching directly for the MLT in the tree's forest, it allows non-binary association in trees and smartly uses the ALT algorithm to reduce the space of research. Both models aren't adapted to mixed type data, which means a generalisation should be implemented. Sharma et al.[**sharma**], implemented a new algorithm for clustering purpose (called HML), their algorithm is able to deal with the Small Sample Size problem, occurring when the covariance matrix in small sample size and high dimension (which is the case with gene expression profiles), using a Singular-Value Decomposition (SVD) method. Although HML gave promising results, the SVD approach should be trade out in order to deal with categorical data, applying Multiple Factor Analysis (MFA)[**pages**] decomposition method for instance. We also can cite an interesting paper concerning hierarchical clustering algorithm by D.Müllner[**mullner**], which served as inspiration for our own algorithm creation.

2.3 Bayesian inference related papers

We will see later that our method considers the Bayes factor to cluster our data or appreciate a given clustering solution, our reference in that regard is the paper of E.Raftery[**raftery**], giving a comprehensive approach to it. The book *Pattern Recognition And Machine Learning*[**bishop**], provided us useful insights to Bayesian inference and related techniques. Finally, an analytically tractable solution in our Gaussian marginal likelihood with conjugate prior theoritical model, we used an unpublished paper from Kevin P.Murphy[**murphy**].

3 Methods

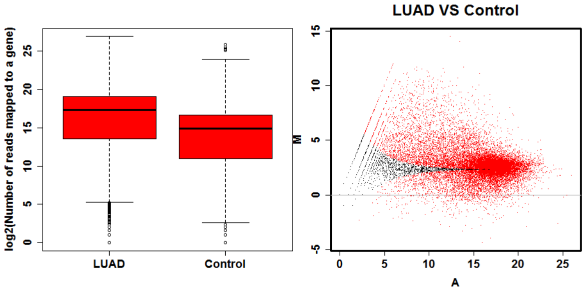
3.1 Data pre-processing

Because we are working with 2 different data sets, it is important to take care of the gene annotation. Indeed, the data set GSE[**shedden**] is annotated with probe names where the TCGA[**TCGA**] is annotated with gene symbols (HGNC). The first step is to use different annotation tables and databases available on R (AnnotationDbi)[**annot**], and from the sources of the data sets in order to obtain comparable data sets. After manipulation of different gene annotations, a gene set in TCGA's data were discarded due to the absence of correspondence in the databases (additional file 1). We also only kept the genes that are in common between GSE and TCGA data sets, discarding the others (Additional file 2) for comparison purpose. Concerning clinical data, The TCGA data set gives us access to a lot of clinical information, but in order to keep a relevant comparing ground with GSE and also to simplify the study we chose to keep only 4 clinical variables (Sex, Age, Race, Smoking history). Clinical data were arranged by convenience, the ”Age” variable was divided in 8 categories according to the distribution frequencies, and ”smoking history” was changed to match its description from the GDC data portal concerning clinical data:

1. : Lifelong non-smoker
2. : Current smoker
3. : Current reformed smoker > 15 years
4. : Current reformed smoker ≤ 15 years
5. : Current reformed smoker duration unknown
6. : Smoker at diagnosis
7. : Unknown

Because the available clinical data for GSE does not make any distinction between categories 3 and 4, we considered them as category 5, the 7th category was considered as missing value (NA under R software).

Among the genes between our two data sets, we decided to explore different gene sets, and study their ability to differentiate molecular sub-types. Indeed, if we can show satisfying results using only a relevant subset of genes it could simplify analyses and diagnosis of new patients. Hence we considered two gene sets, one containing the genes described in [**hayes**] but also the centroids of [**wilk**] and the 100 most mutated genes in LUAD (GDC data portal) which we called ”predictors” gene set. The second one is based on a deferentially expressed genes analysis (DEG) using the R package *DEGseq*. As we had access to the raw count of transcripts from RNA-seq for TCGA, we were able to perform a DEG analysis. We also compared each sub-type to our controls, taking the sub-type classification given by the TCGA study[**TCGA**], see additional file 3. Each sub-type showed more than 17000 DEG, but in order to stay on the same scale as the predictors gene set and also to keep the ability to differentiate between at least 3 sub-types, we kept only the DEG unique to each sub-type, obtaining a 2000+ genes gene set with only one gene in common with the predictors gene set, which was discarded. Finally, and for the following analyses, we split our two data sets into four distinct ones: In table 1, only the ”predictors” gene set is kept for the GSE data set because the DEG analysis was performed on the raw counts of the TCGA data set. Because of some differences in the annotations, the presence of alias in HGNC annotation and absence of some of them in the databases used, the ”predictors” gene set gave two possibilities for the TCGA data set, ”AsGSE” has exactly the same genes as in GSE, but ”diffGSE” gets about a hundred genes different with the GSE data set.



(a) Box plots of diseased patients and healthy ones

(b) MA-plot of diseased patients and healthy ones (controls)
red dots represent DEG

Figure 1: DEG analysis using DEGseq package

Data set	Observations	Variables (num/cat)	Missing values y/n
GSE	442	2249(2245/4)	n
TCGA DEG	230	2858(2854/4)	y
TCGA asGSE	230	1250(1246/4)	y
TCGA diffGSE	230	1160(1156/4)	y

Table 1: Lung Adenocarcinoma data sets

3.2 Hierarchical clustering

Clustering analysis has been and still remains an important tool in machine learning oriented research. Clustering methods are not statistical analyses, indeed, the simple fact of clustering does not grant a final and unique solution. Actually, clustering data randomly will just regroup these data into clusters, but it does not mean that these clusters will have a real meaning, hence our desire to provide here a method assessing this problem and providing statistical significance to clustering results. Finding the number of clusters (K) that reflects the true classes in the data is an important challenge and can determine the quality of the results. Concerning health care purposes, the quality and accuracy of the result represents a great deal, a life saving one. Indeed, the ability to rightly find cancer sub-types allow researchers and physicians to better adapt to each patient particular case, and find new efficient drugs. Thus, we developed a Bayesian model comparison method for this purpose. Our method will be detailed later on, for now lets discuss quickly a certain type of clustering algorithm, and the kind of data we worked on. The data sets we wish to cluster are comprised of different types of data, indeed, the gene expression profiles (GEP) are of numerical continuous type, where clinical knowledge is of categorical type. By categorical type, we mean variables taking a certain range of values (categories) but lacking a mathematical meaning, having or not an order. Numerical data can be discrete or continuous, here no problem to understand the continuous

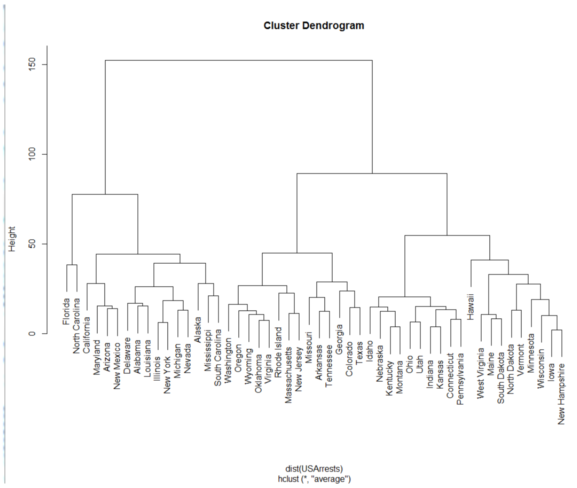


Figure 2: Example of Hierarchical Clustering with R.

form of GEP, on the other hand, categorical data types need some explanations. There are several types of categorical data in our definition of it:

- Binary: A variable which can take only two possible values, i.e: Gender (M/F)
- Nominal: Generalisation of the binary case for K categories, i.e: Race (Australoid/Capoid/Caucasoid/Mongoloid/Negroid)
- Ordinal: As nominal but with an order, i.e: Age
- Interval: similar to ordinal but considering intervals between the values, i.e: also Age

It is important to differentiate these data types because the mathematical tools we apply to them are different, their treatment will also be different. Numerical variables, in our case, will be normalised (log2, $\frac{x-\mu}{\sigma}$...), and categorical ones will be conveniently transformed from characters (MALE/FEMALE), to numbers (1/2), even though it is not absolutely necessary. When it will come to the Bayesian treatment, numerical variables will be assumed to follow Gaussian distributions (close to a Gaussian distribution when we look at the histogram), and categorical ones multinomial distributions, which change completely their handling. Our focus in this study is the hierarchical approach of clustering, thus, we give here further information about this popular clustering method with R. Hierarchical clustering algorithms are a well documented type of algorithm and rely on similarity/dissimilarity matrix produced by distance metrics in order to pair in an agglomerative manner each samples of a given data set. Furthermore, the hierarchical scheme get along nicely with the Bayesian model comparison method. A very simple distance metric example would be the Euclidean distance, defined as:

$$d(i,j) \;=\; \sqrt{\sum_{d=1}^{D}(j_d - i_d)^2}$$

where i and j are two samples of a data set of dimension D. The choice of the metric is primordial and has to reflect the data types and needs of the user, in our case, and because we have both numerical and categorical types, we decided to use the Gower distance:
$d(i,j) \;=\; \frac{1}{N}\sum_{n=1}^{N} d_{ij}^{(d)}$

Where i and j two samples, N the total number of samples in the data set.
$d_{ij}^{(d)}$ is given by:

$$d_{ij}^{(d)} \;=\; \frac{|x_{id} - x_{jd}|}{R_d}$$

Where x_{id} and x_{jd} are the value of the samples i and j for the variable d. R_d is the range for the variable d, dividing by the range avoid to give more weight to variables with greater range. The Gower distance is able to compute the distance between pairs of samples even with variables of different types inside the data set, the results obtained using this metric will act like a benchmark for our own metric. Under R environment, the package "Cluster"[**maechler**] is the perfect tool in association with the base package "stat", for hierarchical clustering. An important part of a hierarchical clustering algorithm is the linkage method, here we will quickly introduce the Ward and Complete linkage methods. The two samples having the shortest distance separating them are linked together, the definition of this distance is what differentiate the linkage methods.
Complete linkage:
If we consider A and B, two different cluster sets, then:

$$D(A,B) \;=\; \max_{a\in A, b\in B} d(a,b) \tag{1}$$

This linkage simply takes the two samples in different clusters which share the maximum distance and bind the two respective clusters, so it uses a dissimilarity matrix.
Ward linkage:
The Ward's method looks for minimising the within-cluster variance, so at each step, it merge two clusters which minimise the overall increase in within-cluster variance:

$$D(A,B) \;=\; \sqrt{\frac{2|A||B|}{|A|+|B|} \cdot \|\overrightarrow{c_A} \;-\; \overrightarrow{c_B}\|^2} \tag{2}$$

With $\overrightarrow{c_K}$ is the centroid vector (euclidean space) of the cluster K. As shown in figure 2, the result of a Hierarchical clustering algorithm is a dendogram (tree), which can be considered as an ensemble of possible clustering solutions. Indeed, if we choose to "cut" the "tree" at a certain height, then we will get the corresponding list of cluster attribution for each sample. Cutting at a high height will give fewer clusters which leads to reduce the sensitivity threshold, maybe regrouping quite different samples together.On the contrary, cutting at low height will give more clusters, threatening the result with over-fitting and considerably complicating the interpretation. So, how to cut that tree in order to have a meaningful number of clusters, combining good sample proximity and simple interpretation? Some tools like AIC or BIC[**burnham**] exists for this purpose, but they use maximum likelihood estimation and not likelihood marginalisation over given parameters. Furthermore, we wish to apply it to the biological field and deal with both numerical and categorical types of data as well as the common case of missing values.
In the further sections, we will first have a short glance at Bayesian inference, then we will give the general framework of it applied to model comparison and finally show how it fits into our clustering method.

3.3 Bayesian Inference

The high popularity of Bayesian inference can be explained by its flexibility and performances in statistical computing and model description. its ability to account for an a priori knowledge, and for uncertainty over statistical parameters, makes the Bayesian approach widely appreciated and implemented into constantly improving tools. Furthermore, the overall accessibility of Bayesian tools, regarding OS or software platforms (R, Python, Matlab...), is a strong argument in favour of its usage. This paragraph is heavily inspired by the work of Christopher M.Bishop[**bishop**]. Bayesian inference is often opposed to the frequentist approach of statistics, indeed, the main difference resides in the statistical reasoning. The Bayesian approach considers that even if a true parameter value exists i.e: a mean, such parameter can be given taking into account the uncertainty on its true value, so as a distribution. The frequentist framework stipulates that, knowing that a true single value of a given parameter exists for the population, it is wrong to assign a probability distribution to it, and relies on parameter estimates from collected population samples. Thus, Bayesian statistics "play" with probability distributions where the frequentist approach only consider the estimated single value given by the available data. From this, we can easily understand why Bayesian inference is favoured, because in real world problems, the cases with few available data and/or high amount of noise are common, hence increasing the uncertainty on the parameters, giving the advantage to the Bayesian vision. So lets discuss further the Bayesian approach on statistics.

First of all, Bayesian inference relies on the famous Bayes' theorem, which is given by:

$$P(A|B) \; = \; \frac{P(B|A)P(A)}{P(B)} \tag{3}$$

Where:

- $P(A|B)$ is the conditional probability, probability of the event A occurring knowing B to be true.
- $P(B|A)$ is the other way around.
- $P(A)$ and $P(B)$ respectively the probabilities of observing the event A and B, also known as the marginal probabilities.

In Bayesian interpretation, the first conditional probability is known as *posterior*, so the updated probability, the second conditional is known as the *Likelihood function* and finally $P(A)$ is, here,the a priori knowledge (prior) on the model. $P(B)$ would be the normalising factor of the equation, insuring that the probability lies between 0 and 1. This theorem comes from the product and symmetry rules of probability, stipulated as following: Product rule:

$$P(A,B) \; = \; P(B|A)P(A)$$

Symmetry rule:

$$P(B,A) \; = \; P(A|B)P(B)$$

Furthermore, from the sum rule, the denominator (normalising constant) can be expressed:

$$P(B) \; = \; \sum_{A} P(A|B)P(B)$$

Detailing the likelihood function, if we consider a Gaussian distributed random variable X (*Independent and identically distributed*) of N samples with parameters μ, σ^2, respectively the mean and the variance, it gives:

$$P(X,\mu,\sigma^2) \; = \; \prod_{n=1}^{N} \mathcal{N}(x_n|\mu,\sigma^2) \tag{4}$$

Which correspond, in this case, to the chained application of the joint probability, equal to the product of the marginal probabilities of each events. In order to have a fully Bayesian approach, we also need to attribute a probability distribution to the parameters, giving them hyperparameters for their respective distributions.

Now that we have established the basics of Bayesian inference, we need to also give some details on a nice application of this kind of statistics, which is the model comparison.

3.4 Model comparison using the Bayesian framework

We saw in the previous paragraph how one can determine the posterior distribution using a Bayesian treatment, given data, parameters and priors. Now we are interested in the comparison between two different models (clustering solutions in our case), on the same data and using the same priors (the parameters varying according to each model), using the Bayesian method, the goal being determining which one of the two considered models fits the data the better. In order to obtain such comparison, the Bayes factor (BF) provides a really straightforward tool[**raftery**], involving easy interpretation, and procuring all the flexibility of the Bayesian inference framework. The idea is simple, compare two cluster solutions and use the Bayes factor in order to favour a model over another, hence, assisting the decision making for clustering methods where the choice of the number of clusters to keep is always subject to debates. So if we consider a data set Dt, the posterior distribution of a model M_K is given by the Bayes' theorem as following:

$$P(M_K|Dt) \; \propto \; P(Dt|M_K)P(M_K) \tag{5}$$

Where:

- $P(M_K|Dt)$ is the posterior probability of the model M_K
- $P(Dt|M_K)$ is the likelihood of the model M_K
- $P(M_K)$ the prior probability
- For convenience the normalising factor is not written here.

Then, if we wish to compare two models M_1 and M_2, we simply look at the ratio of posteriors:

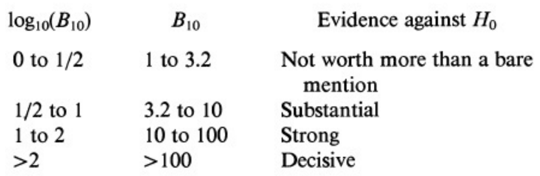
$$\frac{P(M_1|Dt)}{P(M_2|Dt)} \; = \; \frac{P(Dt|M_1)P(M_1)}{P(Dt|M_2)P(M_2)} \tag{6}$$

Thus, the Bayes factor between the model 1 and 2 is defined as:

$$B_{12} \; = \; \frac{P(Dt|M_1)}{P(Dt|M_2)} \tag{7}$$

Which basically can be described as a likelihood ratio when all parameters are known. Or, as it is nicely put in [**raftery**]:

$$Posterior\ odds \; = \; Bayes\ Factor \; \cdot \; Prior\ odds$$



$\log_{10}(B_{10})$	B_{10}	Evidence against H_0
0 to 1/2	1 to 3.2	Not worth more than a bare mention
1/2 to 1	3.2 to 10	Substantial
1 to 2	10 to 100	Strong
>2	>100	Decisive

Figure 3: Model comparison decision base on BF

Now that we defined how to calculate the Bayes factor, we need to precise the calculation of the likelihood functions which require marginalisation over the model parameters. Following the Bayesian treatment, the likelihood densities are obtained by integrating over the parameter space, also known as the likelihood marginalisation, accounting for unknown parameters. So for a given model H_K, with distribution's parameter θ_K (can be a vector) we can write:

$$P(Dt|M_K) \;=\; \int P(Dt|\theta_K, M_K)P(\theta_K|M_K)d\theta_K \tag{8}$$

where:

- $P(Dt|M_K)$ the likelihood density of the model K, how well the model fit the data.
- $P(Dt|\theta_K, M_K)$ the likelihood density of the data given the parameter θ_K depending on the model K, how well the model parameter fit the data.
- $P(\theta_K|M_K)$ the parameter's prior distribution

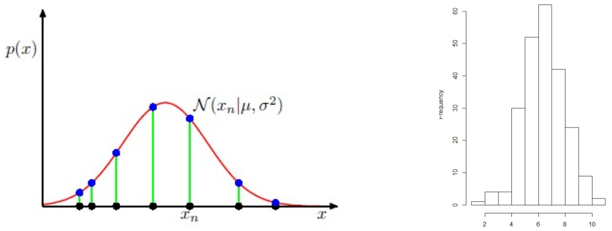
The marginalisation step allow us to obtain the likelihood function only in function of the model, it is said that we marginalise out the parameters. Depending on the data and the choice of prior, this marginalisation integral may very well be intractable, leading to different numerical methods like asymptotic approximation or sampling methods[**raftery**]. In some cases, however, the choice of conjugate priors and/or uninformative ones leads to analytically tractable solutions, which is our case. Once we know this ratio's value, we can draw a conclusion concerning the two models, as given in figure 3. When a small value is obtained, we can't conclude about which one of the models is better, but passing a certain value (greater than 2 in log_{10}), then one model is decisively better than the other.

3.5 Application of the Bayesian scheme to our clustering problem

3.5.1 Likelihood function and priors

In this section we simply describe and define the expression of the likelihood function for our application. We also give information on the choice of priors and their respective hyper-parameters. These expressions will be further detailed when we will express the marginal likelihood. We will start by discussing the assumed probability distributions of the data in hand. Our data is described as following:

- Dt the global notation for our data set, hence representing all data.
- $d = \{1 \dots D\}$ represents the dimension of numerical conditions (gene expression profiles).
- $l = \{1 \dots L\}$ represents the dimension of categorical conditions (clinical variables).



(a) Gaussian distribution, from [bishop]

(b) Histogram of a gene expression profile

Figure 4: Gaussian distribution assumption

- $n = \{1 \dots N\}$ the number of samples (patients).
- $k = \{1 \dots K\}$ the number of clusters.
- X represent all numerical variables.
- $\Sigma \;=\; \sigma_1^2 \cdots \sigma_D^2$ the vector representing the noise of each numerical variable.
- $\Psi = \{\mu_1 \cdots \mu_D\}$ is a vector of gene average values.
- Y represent all categorical variables.
- $\Phi = \{P(Y_1^S) \dots P(Y_L^S)\}$ vector of probabilities associated with each possible value for each categorical variable.

Numerical data:
The gene expression profiles are numerical continuous variables, with appropriate normalisation, the probability distribution can be assumed to be normal. Thanks to that assumption we can write the total likelihood function as in equation 4. Now, if we account for the high dimensional space (several genes), with the joint probability rule it becomes:

$$\mathcal{L} \;=\; P(X|\Psi, \Sigma) \;=\; \prod_{n=1}^{N}\prod_{d=1}^{D}\mathcal{N}(X_{nd}|\mu_d, \sigma_d^2) \tag{9}$$

Choosing a conjugate prior, leads to an analytical expression for the marginal likelihood. Here, we only consider a prior on the mean parameter:

$$P(\Psi|m, \sigma_0^2) \sim \mathcal{N}(\mu_d|m, \sigma_0^2)$$

In our case, the hyper-parameters are $m = 0$ and σ_0^2 set as the maximum of Σ.

Categorical variables:
Y follow multinomial distributions, if we take the case of a binomial distribution, particular case of the multinomial distribution, $Y_l \sim \mathrm{B}\,(N, p)$ taking the values A or B, then the probability of observing the value A is given by:

$$P(Y_l = A) = \binom{N}{k} p^k (1-p)^{N-k}$$

In our case, we can disregard the possible combinations $\binom{N}{k}$ thank to the independence of the samples, hence the likelihood can be written:

$$\mathcal{L} = P(Y|\Phi) = \prod_{n=1}^{N}\prod_{l=1}^{L}\prod_{S} P_S^{\delta(Y_{nl}, S)} \tag{10}$$

Where:

- S represents the set of possible value (states) for a given Y_l
- δ is the Dirac delta function defined as followed:

$$\begin{cases} 1 \text{ if } Y_{nl} = S \\ \quad 0 \text{ else} \end{cases}$$

Lets give an easy example, for a dichotomous categorical variable Y_l taking 2 possible values A and B. Let $P(Y_l^A)$ and $(1 - P(Y_l^A))$ be the probability of the event A and B respectively. N_A, N_B are respectively the number of instances for each value of the variable Y_1 Then the likelihood as described in equation (2) will give:

$$\mathcal{L} = \prod_{1}^{N_A+N_B} \prod_{A,B} P(Y_l^A) \cdot (1 - P(Y_l^A))$$

Which gives:

$$\mathcal{L} = P(Y_l^A)^{N_A} \cdot (1 - P(Y_l^A))^{N_B} \tag{11}$$

Concerning the prior for the categorical variables, a conjugate prior would be a prior following a Dirichlet distribution, but in our case we simply use a uniform prior equating 1.

Overall likelihood function:
When we regroup the two previous likelihood function under a single expression, and take in consideration the existence of clusters, we obtain the following:

$$\mathcal{L} = \prod_{k=1}^{K} \prod_{n \in N_k} \prod_{d=1}^{D} P(X_{knd}|\mu_{kd}) \cdot \prod_{l=1}^{L} \prod_{S} P_S^{\delta(Y_{knl}, S)} \tag{12}$$

Where N_k is the set of index of sample belonging to the cluster k.
We now move to the heart of the method, the step of marginalisation in order to acquire the likelihood of a given model, leading to the Bayes factor determination.

3.5.2 Likelihood marginalisation and Bayes factor

Our method suppose the equiprobability of each clustering model we wish to compare, hence the priors in equation 6 are simplified and the posterior odds directly equate the marginal likelihood ratio between the two models. We still have to express the marginal likelihood, because we used a Gaussian conjugate prior for the numerical variables, we can use the analytic expression in [**murphy**]:

$$\begin{aligned} \mathcal{L}_{Marg} &= P(Dt|m, \sigma^2, \sigma_0^2) \\ &= \int [\prod_{n=1}^{N} \mathcal{N}(x_n|\mu, \sigma^2)] \mathcal{N}(\mu|m, \sigma_0^2) d\mu \end{aligned} \tag{13}$$

So, in the mixed type case, we add the categorical data with ϕ the parameter on it:

$$\mathcal{L}_{Marg} = \int \prod_{n=1}^{N} [\mathcal{N}(x_n|\mu, \sigma^2)] \mathcal{N}(\mu|m, \sigma_0^2) \prod_{S} P_S^{\delta(Y_n, S)}] d\mu d\phi \tag{14}$$

Because the numerical and categorical variables don't have any parameter in common we can write equation 15 as:

$$\begin{aligned} \mathcal{L}_{Marg} = &\int \prod_{n=1}^{N} [\mathcal{N}(x_n|\mu, \sigma^2)] \mathcal{N}(\mu|m, \sigma_0^2)] d\mu \cdots \\ &\cdots \int \prod_{n=1}^{N} [\prod_{S} P_S^{\delta(Y_n, S)}] P(\phi) d\phi \end{aligned} \tag{15}$$

Thus, dissociating the two integral, according a separated treatment for each part. We will give now the becoming of each integration separately, and reassemble them to obtain the final expression.

Marginal likelihood for numerical variables:
Because we have a high dimension case, then the likelihood correspond to the constant application of the joint probability rule over all conditions. The marginal likelihood on numerical variables becomes:

$$\mathcal{L}_{Marg}^{num} = \prod_{d=1}^{D} \int \prod_{n=1}^{N} [\mathcal{N}(x_{nd}|\mu_d, \sigma_d^2)] \mathcal{N}(\mu_d|m, \sigma_0^2)] d\mu_d \tag{16}$$

And by expending the expression for all the parameter values we have:

$$\begin{aligned} \mathcal{L}_{Marg}^{num} = &\int \prod_{n=1}^{N} [\mathcal{N}(x_n 1|\mu_1, \sigma_1^2)] \mathcal{N}(\mu_1|m, \sigma_0^2)] d\mu_1 \cdots \\ &\int \prod_{n=1}^{N} [\mathcal{N}(x_{nD}|\mu_D, \sigma_D^2)] \mathcal{N}(\mu_D|m, \sigma_0^2)] \mu_D \end{aligned} \tag{17}$$

It has been shown in [**murphy**] that the expression in equation 13 can be analytically written:

$$\begin{aligned} \mathcal{L}_{Marg} = &\frac{\sigma}{(\sqrt{2\pi}\sigma)^N (\sqrt{N\sigma_0^2 + \sigma^2})} \exp(\frac{-\sum_{n=1}^{N} x_n^2}{2\sigma^2} \\ &- \frac{m^2}{2\sigma_0^2}) \exp(\frac{\frac{\sigma_0^2 N^2 \overline{x}^2}{\sigma^2} + \frac{\sigma^2 m^2}{\sigma_0^2} + 2N\overline{x}m}{2(N\sigma_0^2 + \sigma^2)}) \end{aligned} \tag{18}$$

In our case we choose $m = 0$, and accounting for the multivariate Gaussian distribution it becomes:

$$\begin{aligned} \mathcal{L}_{Marg}^{num} = &\frac{1}{(\sqrt{2\pi}^{DN} \prod_{d=1}^{D} \sigma_d^{N-1} (\sqrt{N\sigma_0^2 + \sigma_d^2})} \\ &\exp(-\frac{1}{2} \sum_{d=1}^{D} \sum_{n=1}^{N} \frac{x_{dn}^2}{\sigma_d^2}) \exp(\frac{\sigma_0^2 N^2}{2} \sum_{d=1}^{D} \frac{\overline{x}_d^2}{(N\sigma_0^2 + \sigma_d^2)\sigma_d^2}) \end{aligned} \tag{19}$$

Finally, applying the *natural logarithm* to equation 19, it simplifies as:

$$\begin{aligned} ln(\mathcal{L}_{Marg}^{num}) = &ln(\frac{1}{(\sqrt{2\pi}^{DN} \prod_{d=1}^{D} \sigma_d^{N-1} (\sqrt{N\sigma_0^2 + \sigma_d^2})}) \\ &- \frac{1}{2} \sum_{d=1}^{D} \sum_{n=1}^{N} \frac{x_{dn}^2}{\sigma_d^2} + \frac{\sigma_0^2 N^2}{2} \sum_{d=1}^{D} \frac{\overline{x}_d^2}{(N\sigma_0^2 + \sigma_d^2)\sigma_d^2} \\ = &-\frac{DN}{2} ln(2\pi) + \sum_{d=1}^{D} [\frac{(N-1)}{2} ln(\sigma_d^2) + \frac{1}{2} ln(N\sigma_0^2 + \sigma_d^2)] \\ &- \frac{1}{2} \sum_{d=1}^{D} \sum_{n=1}^{N} \frac{x_{dn}^2}{\sigma_d^2} + \frac{\sigma_0^2 N^2}{2} \sum_{d=1}^{D} \frac{\overline{x}_d^2}{(N\sigma_0^2 + \sigma_d^2)\sigma_d^2} \end{aligned} \tag{20}$$

Using the *natural logarithm* lower the computational time needed and also simplify conveniently the *exponential* in the expression.

Marginal likelihood for categorical variables:
The integral on one categorical variable, seen in equation 8 can be expended, once again thanks to the joint probability rule, in the multivariate case as:

$$\mathcal{L}^{cat}_{Marg} = \prod_{l=1}^{L} \int_0^1 \prod_{n=1}^{N} \prod_{S} P_S^{\delta(Y_{nl},S)} P(\phi_l) d\phi_l$$
$$= \int_0^1 \prod_{n=1}^{N} \prod_{S} P_S^{\delta(Y_{n1},S)} P(\phi_1) d\phi_1 \cdots \int_0^1 \prod_{n=1}^{N} \prod_{S} P_S^{\delta(Y_{nL},S)} P(\phi_L) d\phi_L \tag{21}$$

Here we can integrate between 0 and 1 because we are working with probabilities, so if we ignore the prior which always equates 1, we see that the marginal likelihood of each categorical variable is equivalent to a Beta function. Beta function:

$$\mathbf{B}(x,y) \ = \ \int_0^1 t^{x-1}(1-t)^{y-1} dt$$

We see here that the Beta function fits the expression of the likelihood in the example of equation 11. We can easily make the analogy between t and $1-t$ with $P(Y_l^A)$ and $(1 - P(Y_l^A))$. The difference resides in the parameters of the function, indeed the Beta function deals with x and y, which gives $x-1$ and $y-1$ as power of t and $t-1$. In the example it is N_A and N_B which take this role, so we can easily express them together:

$$x = N_A \ + \ 1 \tag{22}$$
$$y = N_B \ + \ 1 \tag{23}$$

So the integral, if expressed with N_A and N_B becomes:

$$\mathbf{B}(N_A, N_B) \ = \ \int_0^1 t^{N_A}(1-t)^{N_B} dt$$

So it perfectly matches with our example. Even in the case of variables with more than two possible events, the Beta function can be generalised in its calculation, using the other definition of the Beta function, closely related to the Gamma function, given by:

$$\mathbf{B}(x,y) \ = \ \frac{\Gamma(x)\Gamma(y)}{\Gamma(x \ + \ y)}$$

Gamma function:

$$\Gamma(n) \ = \ (n-1)!$$

And with our example:

$$\mathbf{B}(N_A, N_B) \ = \ \frac{\Gamma(N_A+1)\Gamma(N_B+1)}{\Gamma(N_A+N_B+2)} \ = \ \frac{N_A!N_B!}{(N_A+N_B+2)!}$$

When we generalise the expression, we have:

$$\mathbf{B}(Y_l) \ = \ \frac{\prod_{j=1}^{J}\Gamma(N_{Y_l=j}+1)}{\Gamma(N+sup(S_l))} \ = \ \frac{\prod_{j=1}^{J} N_{Y_l=j}!}{(N+sup(S_l))!} \tag{24}$$

With S_l, the set of possible events of the variable Y_l, N, the number of samples, j iterating over the set S_l. Thus, from equation 21, we can write the likelihood:

$$\mathcal{L}^{cat}_{Marg} = \frac{\prod_{j_1=1}^{J_1}\Gamma(N_{Y_1=j_1}+1)}{\Gamma(N+sup(S_1))} \cdots \frac{\prod_{j_L=1}^{J_L}\Gamma(N_{Y_L=j_L}+1)}{\Gamma(N+sup(S_L))} \tag{25}$$

Then, for convenience and for saving computational resources, we apply the *natural logarithm* to equation 25:

$$\mathcal{L}^{cat}_{Marg} \ = \ \sum_{j_1=1}^{J_1} ln(\Gamma(N_{Y_1=j_1}+1)) - ln(\Gamma(N+sup(S_1))) + \cdots$$
$$\cdots + \sum_{j_L=1}^{J_L} ln(\Gamma(N_{Y_L=j_L}+1)) - ln(\Gamma(N+sup(S_L))) \tag{26}$$

Total marginal likelihood:
So if we bind the two parts together:

$$ln(\mathcal{L}_{Marg}) \ = \ ln(\mathcal{L}^{num}_{Marg}) \ + \ ln(\mathcal{L}^{cat}_{Marg}) \tag{27}$$

Notice that the marginal likelihood expressions are given considering only one cluster, but as shown in equation 12, accounting for more than one cluster would simply require to add a sum over all clusters and change the total number of sample into the number of sample in each respective cluster.

Bayes Factor:
Now that we have the expression of the marginal likelihood over the respective parameters, which, once marginalised gives the likelihood of a given model, we can simply get the BF by applying the ratio (bocoming a difference with ln) between two models 1 and 2:

$$ln(BF_{12}) \ = \ ln(\mathcal{L}^{M1}_{Marg}) \ - \ ln(\mathcal{L}^{M2}_{Marg}) \tag{28}$$

And from the results and the information given in figure 3, we can see which one of the two models fits our data the better. From this value, we can directly compare clustering results from any algorithm, and we can also create a clustering method based on the behaviour of the BF, hence deciding to merge of samples together or not based on the value of the BF.

3.6 Proposed Methods

We want here to present different implementations, based on the BF scheme. Concerning the algorithms, only one was successfully implemented but the reasoning behind the other ones will be explained as well.

3.6.1 Bayes Factor calculation from clustering result

The first, and simplest, implementation reside in the simple calculation of the BF from a list of already clustered and labeled data. The source code can be found in supplementary file. Here we simply feed the function with two clustering results, one with 3 clusters and the other with 4 for instance, we also give it the data set it is from and it return the BF after calculation of the numerical and categorical marginal likelihood separately. This implementation is able to deal with missing values, it basically replace them by zeros when calculation, hence missing values don't participate in the marginal likelihood calculations.

The source code of all implementation is given in supplementary files.

3.6.2 Hierarchical Bottom-Up clustering Algorithm

The first hierarchical algorithm takes a bottom-up approach, which means it starts by assuming all the samples as distinct clusters and merge them at each step until all the samples are gathered in a single cluster. In this algorithm, we consider all possible combinations, meaning that the tree will not be reduced as a binary one, more than two branches association is possible. At each step of the algorithm, the marginal likelihood ($\mathcal{L}_{Marg}$) between each pair of samples is computed, then the samples maximising $\mathcal{L}_{Marg}$ are taken and merged together into one cluster. That decision comes from the fact that maximising the $\mathcal{L}_{Marg}$ means maximising the fitness to the observed data. The matrix is then updated to reflect the new assumed model, accounting for the freshly created cluster. The calculation of such matrix requires the marginal likelihood of the previous assumed model, we show that a mathematical simplification on the likelihood expression can be used for the first algorithm's iteration. Indeed, a lot of calculation is unnecessary for our purpose, which allows us to simplify the expression, hence simplifying the calculation, reducing the required computational resources. Lets detail the mathematical simplifications, which will be used in the other implemented methods as well. First of all, we will use the following notation by convenience:

- $a = -\frac{DN_k}{2}ln(2\pi) + \sum_{d=1}^{D}[\frac{(N_k-1)}{2}ln(\sigma_d^2) + \frac{1}{2}ln(N_k\sigma_0^2 + \sigma_d^2)]$
- $b = -\frac{1}{2}\sum_{d=1}^{D}\frac{x_d^2}{\sigma_d^2}$
- $c = \frac{\sigma_0^2 N_k^2}{2}\sum_{d=1}^{D}\frac{\overline{x}_d^2}{(N_k\sigma_0^2+\sigma_d^2)\sigma_d^2}$
- lets call d the categorical part of the marginal likelihood
- $d \ = \ \sum_{s=1}^{S}[ln(\Gamma(N_s))] \ - \ ln(\Gamma(N + sup(S)))$
- taking even more liberties, we can note $a(N_k = 1)$ corresponding to a but with N_k, number of samples in the cluster k, equals 1.
- In the same way, $b(x_n)$, (x_n) and $d(x_n)$ respectively the part b, and d taking the value of the sample x_n for their calculation.

a,b and c come from equation 20, just without the sum over all samples. The first step is to calculate the marginal likelihood of the model 0, the model assuming each sample as its own cluster. In such scheme, N_k would equates 1, and the sample mean in c directly equates the sample value by gene. Furthermore, d also gives great simplification because the numerators in equation 25 becomes 0 with *ln*. Hence we get:

$$\mathcal{L}_{Marg}^{M0} \ = \ N\cdot a(N_k = 1) \ + \ \sum_{n=1}^{N} b(x_n) \ + \ \sum_{n=1}^{N} c(x_n) \ + \ \sum_{n=1}^{N} d(x_n) \tag{29}$$

Now if we desire to have the marginal likelihood of any pair of samples i and j:

$$\begin{aligned}\mathcal{L}_{Marg}^{ij} \ &= \ (N_k - 2)a(N_k = 1) \ + \ a(N_k = 2) \ + \ \sum_{n=1}^{N} b(x_n) \\ &+ \ \sum_{N\backslash i,j} c(x_n) + \ c(\overline{x_i x_j}) \ + \ \sum_{N\backslash i,j} d(x_n) \ + \ d(x_i, x_j)\end{aligned} \tag{30}$$

We then can express it using $\mathcal{L}_{Marg}^{M0}$:

$$\begin{aligned}\mathcal{L}_{Marg}^{ij} \ &= \ \mathcal{L}_{Marg}^{M0} \ - \ 2a(N_k = 1) \ + \ a(N_k = 2) \ + \ \sum_{n=1}^{N} b(x_n) \\ &+ c(\overline{x_i x_j}) - \ c(x_i) \ - \ c(x_j) \ + \ d(x_i, x_j) \ - \ d(x_i) \ - \ d(x_j)\end{aligned} \tag{31}$$

So for each sample pairs, the difference resides only in the parts depending on x_i and x_j, the rest can be considered constants and ignored, which gives finally:

$$\begin{aligned}\mathcal{L}_{Marg}^{ij} \ \propto \ c(\overline{x_i x_j}) \ - \ c(x_i) \ - \ c(x_j) \ + \ d(x_i, x_j) \\ - \ d(x_i) \ - \ d(x_j)\end{aligned} \tag{32}$$

The simplification is possible only for the first iteration, after the first merging, it falls due to the unpredictability of possible sample associations. The lack of such generalised simplification makes the algorithm time complexity too high (exponential), the marginal likelihood matrix having to be updated after the creation of each cluster. Another version of this algorithm was under development, with main difference a binary tree forcing approach in order to keep the simplification applicable at each step. Unfortunately, as soon as the number of clusters becomes odd, the simplification falls and complications arise. boxruled [!] UpdateMatrixupdateMatrix InputinputOutputoutput $Dt = (N, D + L)$, σ^2, prior hyper-parameters (HP) A list of cluster solutions associated with respective BF $L_M \ \leftarrow L_{Marg}^{M0}$
$Cl \leftarrow$ list of clusters (each sample at first)
$N_k \leftarrow$ vector of number of samples by cluster (only 1s at first)
Create the first marginal likelihood matrix $L_{Mat} \ \leftarrow$ call $Cl, \ N_k, \ Dt, \ \sigma^2$, HP
$K \ \leftarrow$ N
we have more than one cluster Get the samples names of the maximum in L_{Mat}
Update Cl by merging the two samples
Update N_k as well
Compute the marginal likelihood of the new model
Calculate the Bayes Factor
Replace L_M by the new model L_M
$Cl, \ N_k, \ Dt, \ \sigma^2$, hyper-parameters
K = K - 1 Result $\leftarrow$ Append(Cl,K,BF)
Return Result boxruled [!] InputinputOutputoutput Cl,N_k,$Dt = (N, D)$, σ^2,HP List of clusters, each sample value vector Marginal likelihood matrix Create a matrix filled with NAs $L_{Mat} \ \leftarrow matrix(NA)$
i in 1 to the number of clusters-1 j in i+1 to the number of cluster $L_{Marg}^{ij} \ \leftarrow$ Compute the the marginal likelihood between clusters i and j
Put the value into the corresponding index of the new matrix $L_{Mat}[i, j] \ \leftarrow \ L_{Marg}^{ij}$ Return L_{Mat} The function updating the matrix return a NxN matrix with the lower triangle filled with NAs. This algorithm is able to deal with missing values.

3.6.3 Hierarchical Top-Down clustering Algorithm

The second implementation is a hierarchical Top-down algorithm, which means, we start with all samples gathered in a single cluster and at each step we divide it in smaller clusters based on some conditions. The work-flow of this algorithm is pretty straightforward and uses the Gower distance, from the "Cluster" package under R, in addition to marginal likelihood. This approach allow us to avoid having to re-calculate a huge matrix, hence saving a lot of computation time, but it also presents an imperfection, the choice of the dividing condition. Indeed, each time we want to split a given cluster in two different ones, we face some troubles like, what will be the rule of division or which cluster to divide first. Unfortunately it is of great difficulty to predict the behaviour of the marginal likelihood, hence finding a relevant way of splitting a cluster is challenging. For lack of anything better, the splitting condition was set to simply divide a cluster into two equally, or almost equally (case of odd number of samples), sized clusters. This separation relies on two steps:

1. Calculate the Gower distance matrix for each cluster and get one of the two samples having the maximum distance, we call that sample the *reference sample*

2. Compute the marginal likelihood between the *reference sample* and all the remaining samples

Through that method, we get, for each singular cluster, the maximum Gower distance and one of the sample associated with it. We can then sort by decreasing order these distances, which will give us the order of cluster splitting. This assumption is simple but logical, indeed, the cluster containing the two most distant samples is the strongest candidate for being split in first. Once we have that list, we just have to iterate on it and calculate for each respective cluster the marginal likelihood combinations. We then, once again use a decreasing order sorting to basically obtain on the first position the closest samples and at the latest positions the farthest ones. Finally, we split in two, taking the $floor(\frac{N_k}{2})$ first samples (so if odd number, take the inferior integer), and associate them with the *reference sample* to create a new cluster, the remaining samples are associated into a second new cluster. At each new iteration over the ordered list, we compute the marginal likelihood of the changed model (one more cluster to take into consideration), and compute the BF with the former model. Here the first assumed model is as following (all sample in one cluster):

$$\mathcal{L}^{M0}_{Marg} = a(N_k = N) + \sum_{n=1}^{N} b(x_n) + c(\overline{x_1 \cdots x_N})d(x_1, \cdots, x_N) \qquad (33)$$

And when calculating the marginal likelihood with a *reference sample*, the mathematical simplification can go even further than in equation 32, indeed here x_i will be the same for all, x_{ref}, so we can consider it a constant and ignore it, which leads to:

$$\mathcal{L}^{ref,j}_{Marg} \propto c(\overline{x_{ref}x_j}) - c(x_j) + d(x_{ref}, x_j) - d(x_j) \qquad (34)$$

boxruled [!] InputinputOutputoutput $Dt = (N, D + L)$, σ^2, prior hyper-parameters (HP), number of maximum desired clusters A list of cluster solutions associated with respective BF
$old.L_M \leftarrow L^{M0}_{Marg}$
Cl ← list of clusters (a single cluster with all samples)
K accounts for the number of clusters, 1 at first $K \leftarrow 1$
We haven't reached the desired number of clusters gower ← Compute the Gower distance matrix of each cluster
gower.max ← fetch the reference sample and the distance value for each cluster
sort the values by decreasing order gower.max.ord ← sort(gower.max)
indices ← clusters indices in the order given by gower.max.ord
i in indices Take the cluster to split in Cl
Compute the marginal likelihood with the reference sample
cl.split ← newly formed clusters
cl.remaining ← clusters not split yet
K ← K+1
Marginal likelihood of model k $new.L_M \leftarrow$ compute model marginal likelihood
$old.L_M$ equates $L^{M_{K-1}}_{Marg}$ BF ← $old.L_M - new.L_M$
Update Cl
Result ← append Cl and BF Return Result This algorithm is also able to deal with missing values in the same manner as the first one.

3.6.4 Simplified hierarchical Bottom-Up clustering Algorithm

The last implemented algorithm is also a bottom-up one, but in order to minimise the overall time complexity, only the first marginal likelihood matrix is calculated and all clusters are formed from it. This algorithm is of really simple design, the idea is to feed it with a desired number of cluster and it will stop once this number is reached. At first, the algorithm calculate the marginal likelihood matrix as in the first step of the first algorithm, using the mathematical simplification shown earlier. Then, from this matrix it will fetch at each step the maximum and the two samples corresponding to put them together in a single cluster. Once it is done, it puts NA into the index of this maximum, which allows to avoid taking the same samples twice. Because of the possibility of getting samples already in a cluster formed at a previous step, it is necessary to add a checking task, which behaves differently according to the case we are in:

- if the two samples are already in the general cluster solution
  1. Get the respective cluster index already comprising these samples.
  2. Merge these two clusters into one.
- if only one of the sample is already in a cluster
  1. Identify which of the two is already in a cluster.
  2. Get the index of the corresponding cluster
  3. Add the second sample into that cluster
- if none of the samples are already in an existing cluster, we simply put them together in a new cluster

By following this work-flow, and looping it until all samples are put into a unique cluster, we obtain the final cluster solution. Unfortunately, a flaw in the design made it difficult to get the desired result, being able to both cluster all samples and to put them into a given number of clusters was not as easy as anticipated. By lack of time some issues were not fully addressed, leading to poor results, the algorithm having tendencies to stop before clustering all samples and creating really unbalanced clusters with generally a big one regrouping most of the samples and the remaining ones having a handful number of samples. boxruled [!] UpdateMatrixupdateMatrix MergeBothmergeBoth MergeOnemergeOne InputinputOutputoutput $Dt = (N, D + L)$, σ^2, prior hyper-parameters (HP), number of clusters desired A cluster solution associated with its BF $L_M \leftarrow L^{M0}_{Marg}$
Cl ← list of clusters (each sample as cluster)
N_k ← vector of number of samples by cluster (only 1s)
$L_{mat} \leftarrow$ Cl,N_k,Dt,σ^2,HP
List containing the general cluster solution, will be update at each iteration $Cl.sol \leftarrow$ void list
We haven't reached the desired number of clusters samp.max ← samples associated with max(L_{mat})
if we note i and j indices of the current maximum $L_{mat}[i, j] \leftarrow$ NA
both samples already in Cl.sol Cl.sol ← Cl.sol,samples
Only one sample already in Cl.sol CL.sol ← Cl.sol,samples
None of the samples are already in Cl.sol Cl.sol ← Append in a new index the two samples
$L_{marg} \leftarrow$ Compute marginal likelihood of the model
BF ← $L_M - L_{marg}$
Cl.sol ← Append Cl.sol with BF
Return CL.sol boxruled [!] InputinputOutputoutput Cluster list, two samples to add A new list containing all clusters loop on each element of the clusters list one of the samples is in the current cluster indices ← Get the index of this cluster
Do nothing
Merge the two clusters corresponding to indices into one
return the new cluster solution The function *MergeOne()* acts like in algorithm 3.6.4, but directly puts the sample not already clustered in the cluster containing the other sample. These functions avoid getting duplicated samples and clusters.

4 Results

4.1 Method testing

In order to assess the performance of the methods, we applied them to different data sets found on [**source1**] and [**source2**]. We used them to investigate the computational time required by our methods, and because we know the true classes for some of the data sets, we were also able to appreciate their accuracy. The list and description for each data set can be seen in table 2: We chose these data sets in order to consider different possibilities, hence we have data sets with both categorical and numerical variables, but also the Wine data set with only numerical, opposed to the Zoo one having only categorical variables. They also offer different dimensions, which allow us to explore the time required by the algorithms to perform.

4.1.1 Bayes Factor induced clustering

For comparison purpose, we decided to apply another already existing method to the same data sets. As stated before, the Gower distance can be used on mixed type of data, then applying a simple hierarchical clustering algorithm, available under R with the *hclust()* function, with either Ward or Complete method of linkage (equations 1,2). We applied the same linkage method but instead of the Gower distance matrix, we calculated the Marginal likelihood matrix between each possible sample pairs. We are interested in finding the right number of classes, the accuracy, and the computational time required. In table 3, *Candidates* repre-

Data set	Observations	Variables (num/cat)	Class identifier y/n	Missing values y/n	# of classes
Abalone	4177	9(8/1)	n	n	29
Arrhythmia	452	279(272/7)	n	n	16
Cortex	1080	81(78/3)	n	y	8
Credit	690	15(6/9)	y	y	2
Dermatology	366	34(1/33)	y	y	6
Wine	178	13(13/0)	y	n	3
Zoo	100	16(0/16)	y	n	7

Table 2: Data sets used for testing our methods

Data set	Metric	Method	Candidates	Best candidate	# of classes
Abalone	Gower	Ward	30	30	29
		Complete	24,28,32	24	
	Marg	Ward	22,27,30	27,30	
		Complete	26	26	
		HC top-down	18	18	
Arrhythmia	Gower	Ward	16,18	18	16
		Complete	18,20	18	
	Marg	Ward	17	17	
		Complete	not conclusive		
		HC top-down	11,16,18	18	
Cortex	Gower	Ward	8,12,14	12	8
		Complete	8,10,12	12	
	Marg	Ward	13	13	
		Complete	9,13	13	
		HC top-down	6,8,11	6	
Dermatology	Gower	Ward	not conclusive		6
		Complete	6,9	6	
	Marg	Ward	5	5	
		Complete	5,8	5	
		HC top-down	5,8	5	
Wine	Gower	Ward	2	2	3
		Complete	3	3	
	Marg	Ward	3	3	
		Complete	2	2	
		HC top-down	2	2	
Zoo	Gower	Ward	7,9	7	7
		Complete	6	6	
	Marg	Ward	not conclusive		
		Complete	5,7	7	
		HC top-down	4,7,9	7	

Table 3: Table of results for Gower and Marginal likelihood metrics

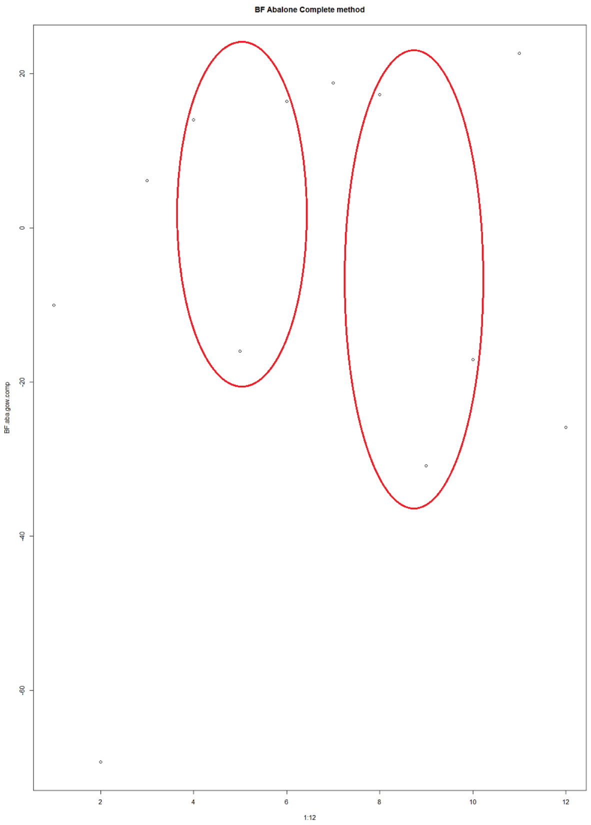


Figure 5: Conclusive example of BF plot interpretation

sent the cluster models which represent good candidates for the best cluster model and *# of classes* is the true number of classes in the data set. The results obtain in table 3, were found by plotting the BFs for each model, determining candidates model, and in the case were several candidates were found, comparing the BFs between candidates to find the best one. Only the algorithm top-down was used, because the bottom-up ones lacked optimisation or yielded really poor clusters. As this table shows, all techniques manage to get pretty close or even equalise the true number of classes in the data sets, so we consider that the ability of our implemented methods to discover the right number of clusters is satisfying. Figure 5 shows how we choose the candidate models. From equation 28 and because each dot represent the log BF between the previous and the current model, i.e: 2 clusters vs 3 clusters, we concluded that when the dot is below zero, it means that the current model is better than the previous one. Furthermore, when after going below zero the next dot goes in higher values (the current model becomes the former one for this dot), it confirm that the "new" former model is better. We also noticed that even when not going below zero, but showing the same pattern can also predict a good candidate, as shown in figure 6, there are two candidates not giving a log BF below zero, but are actually better than the one going below zero. Most of the BF were strong or decisive according to figure 3. Only for the data set Abalone we got a substantial result between the 22 cluster model and the 27 one, and a not worth even a bare mention case between 27 clusters and 30 clusters, which is why we considered them both as best candidates.

4.1.2 Accuracy results

Some data sets have well defined class identifiers, allowing us to compute the accuracy of the different methods. In order to determine the accuracy, we simply used the Jaccard's index between the true classes and the clustering results. For two sets A

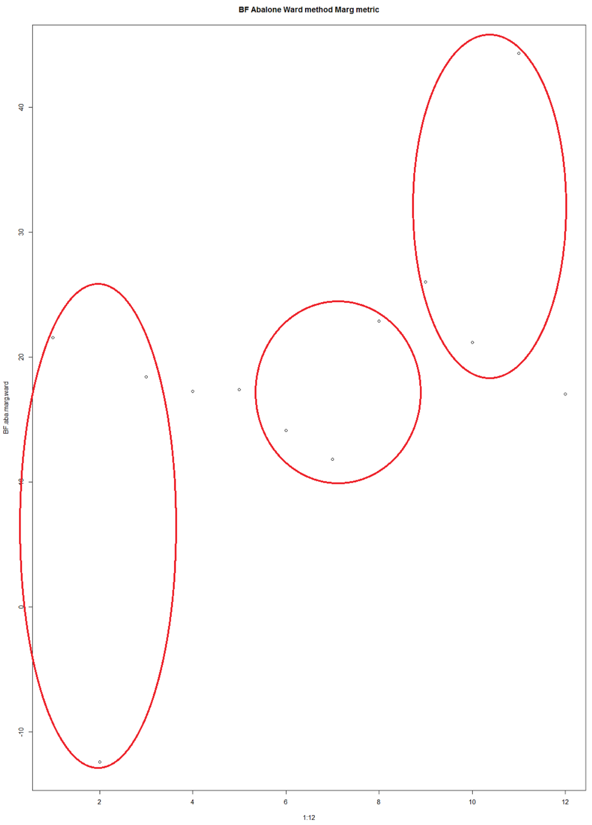


Figure 6: Conclusive example with less obvious candidates

Data set	True class	Gower Ward	L Marg Ward	HC Desc
Credit	1	0.44	**0.48**	0.37
	2	0.44	**0.59**	0.42
Dermatology	1	0.66	**0.98**	0.45
	2	0.55	**0.74**	0.29
	3	**1**	**1**	0.58
	4	0.44	**0.8**	0.13
	5	**1**	0.8	0.11
	6	**1**	**1**	0.19
Wine	1	**0.95**	0.32	0.59
	2	**0.92**	0.4	0.59
	3	**0.96**	0.27	0.29
Zoo	1	**0.87**	0.53	0.61
	2	**1**	0.95	0.45
	3	**0.44**	**0.44**	0.125
	4	0.92	**1**	0.79
	5	**0.33**	**0.33**	0.13
	6	**0.8**	**0.8**	0.33
	7	**0.8**	0.7	0.46

Table 4: Table of maximum Jaccard index by method

and B, the Jaccard's index can be calculated with:

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{35}$$

So, this index is merely a similarity measure between two sets. In table 4, we put a summary of the results, putting only the maximum Jaccard index found for the respective method used and for each class. Our results show that the accuracy using the Gower distance matrix and the log marginal likelihood matrix share the lead, depending on the data set considered. However, our hierarchical top-down clustering method perform poorly and rarely goes over an index value of 0.5. Although our methods are able to find the right number of clusters, we showed here that the accuracy is not really on point, especially for the hierarchical top-down algorithm we implemented.

data set	Method	Candidates	Best one	Second best
GSE	HC Ward	3,7	7	3
	HC Comp	4,8	8	4
	HC TD	2,4,6,8	2	4
TCGA DEG	HC Ward	2,6	2	6
	HC Comp	2,5,7	2	5
	HC TD	2,5,8	2	5
TCGA AsGSE	HC Ward	6	6	
	HC Comp	4,6	4	6
	HC TD	2,4,7,9	2	4
TCGA diffGSE	HC Ward	6,9	6	9
	HC Comp	2,9	2	9
	HC TD	2,4,7,9	2	4

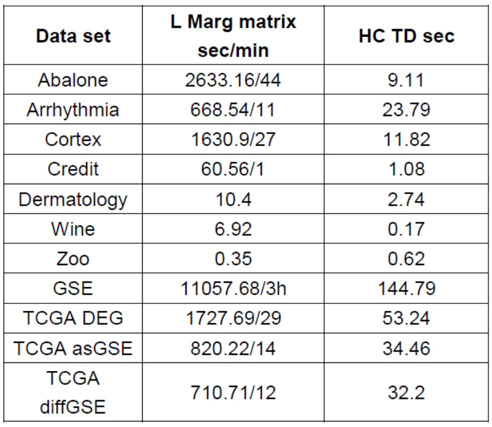
Table 5: Clustering results with the lung Adenocarcinoma data sets

Data set	Subtype	HC TD	L Marg Ward	L Marg Comp	Best subtype predictor
TCGA DEG	TRU	0.28	0.28	**0.32**	PI
	PI	**0.39**	0.3	0.3	
	PP	0.21	0.21	**0.24**	
TCGA AsGSE	TRU	**0.4**	Not conclusive	0.33	TRU
	PI	0.23	Not conclusive	**0.29**	
	PP	0.21	Not conclusive	**0.27**	
TCGA DiffGSE	TRU	**0.38**	Not conclusive	0.26	TRU
	PI	**0.24**	Not conclusive	**0.24**	
	PP	**0.22**	Not conclusive	0.21	

Table 6: Methods Accuracy on Adenocarcinoma TCGA data sets

4.2 Results on the lung adenocarcinoma data sets

Even thought the test were not conclusive, and accurate enough, we applied our methods to different data sets to see if we could still find interesting results. We used the 4 data sets described earlier in section 3.1, and obtain the results shown in table 5. We can see that in most cases, our methods find a best cluster solution of two clusters, but as it has been previously determined in [**hayes**], [**wilk**] and [**TCGA**], the number of sub-types is at least three. From there, we can assume that, unfortunately, our scheme is not sensitive enough, and pushing the investigation further would not really lead to meaningful outcomes. In the case of the TCGA data sets, because we have access to the classification made in [**TCGA**], we just can have a look at the accuracy yielded by the different gene sets taken into consideration, which would, at least, give us which gene set would seem to be a better predictor to lung Adecarcinoma sub-types exploration.
Our analysis about accuracy, shown in table 6, was produced by taking a three clusters solution from our different methods. Although it yielded poor results, especially the Ward method being often inconclusive due to an aggregation of most of the samples into one cluster, we still can see that the differentially expressed



Data set	L Marg matrix sec/min	HC TD sec
Abalone	2633.16/44	9.11
Arrhythmia	668.54/11	23.79
Cortex	1630.9/27	11.82
Credit	60.56/1	1.08
Dermatology	10.4	2.74
Wine	6.92	0.17
Zoo	0.35	0.62
GSE	11057.68/3h	144.79
TCGA DEG	1727.69/29	53.24
TCGA asGSE	820.22/14	34.46
TCGA diffGSE	710.71/12	32.2

Table 7: Computational time by data set for each method

gene set is a decent predictor for the PI sub-type, where the gene set in "AsGSE" is a decent predictor for the TRU sub-type. Unfortunately, none of the considered gene sets were able to be good enough predictor for the PP sub-type. Further analyses have not been conducted, deemed to be unnecessary given such inconclusive results. This is why, we consider our methods as a framework, laying the foundations for future research using the same scheme but bringing some tuning to our model.

4.3 Time Complexity

Optimisation is a very important feature in computer science, being able to implement an algorithm computing quickly is a great challenge. We have shown how we could simplify the mathematical model to avoid some unnecessary calculation, and because we used R, we can also talk about some optimisation problem we had. R software is far from being an optimal environment for algorithmic implementations, especially under Windows. Even if some tools like R open for Windows or the package "Parallel", for Mac OS or Linux distributions, allow parallel computing, not all functions in R profit from it. Furthermore, the handling of objects is not really practical, often reduced to use lists for simplification purpose. So, overall, coding an algorithm with R asks additional reflections and adaptations in order to minimise the computational time consumption. In table 7, the time required for each case is given, and as we can see, the Marginal likelihood matrix is a quite time consuming process. For huge data sets, we quickly reach several hours of computational time, which is not desirable. These results were obtained with an Intel Core i7-3970X 3.50GHz processor, 32GB of RAM under Windows 7 Ultimate 64-bits. At first glance, we see that the number of samples play a more important role in the time consumption than variables. Indeed, if we look at the time required for the GSE data set (442 observations, 2249 variables) and for the TCGA DEG one (230, 2858), we notice a huge difference even though the TCGA data set has 600+ variables more than GSE for only 200+ less observations. We confirmed that idea by varying the number of observations with a constant number of variables and doing the opposite for the marginal likelihood matrix calculation, which is shown in figure 7 and 8. Indeed, from these figures, wee observe that the time consumption increase linearly when varying the variable number only, and takes the form of a quadratic equation when varying the observations only. So, the overall time consumption when varying the number of observations and variables together give the figure 9. The time complexity given in function of N (number of samples) for our implementations is:

- Marginal likelihood matrix: $O(N^2)$
- Top-Down clustering algorithm: $O(N)$

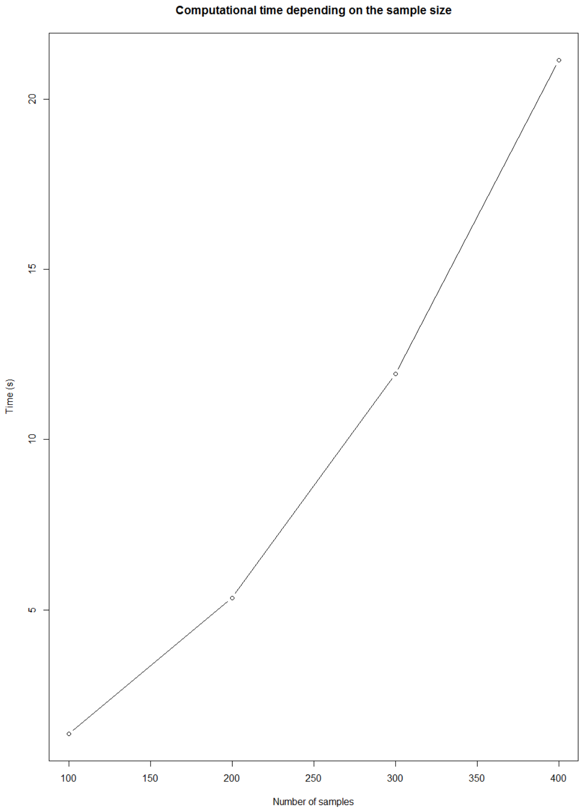


Figure 7: Time consumption evolution in function of number of observations

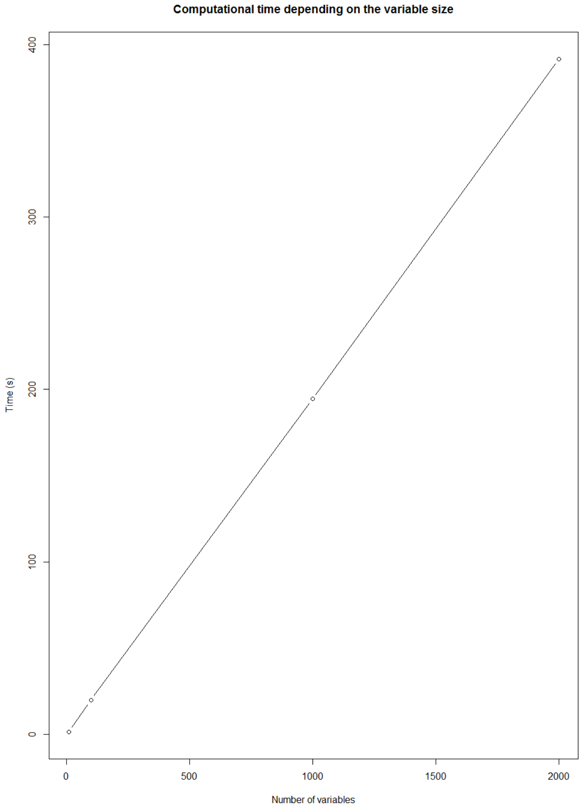


Figure 8: Time consumption evolution in function of number of variables

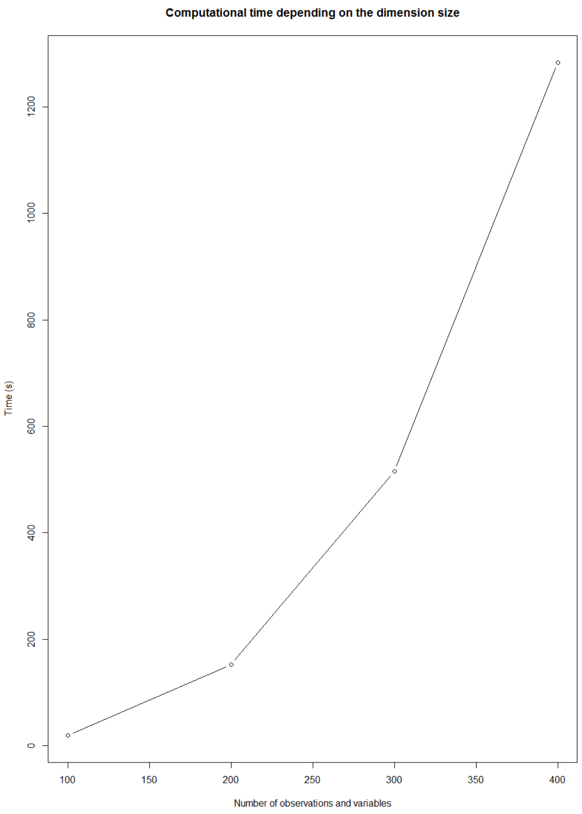


Figure 9: Time consumption evolution in function of number of observations and variables

5 Discussion

As our results show, the pairwise $\mathcal{L}_{Marg}$ matrix could represent an interesting alternative to distance matrix calculation for mixed type of data. Indeed, this method is able to mostly find the right number of classes, or at least a close one, for most of the test data sets. The Top-down clustering algorithm is also able to find a satisfying number of clusters, close to the true one, but not as good as the other method. Unfortunately, we also showed that although we find a relevant number of clusters, the accuracy is not on point, the Jaccard index struggling to go over 0.5. Once again an association between the marginal likelihood matrix and a Ward or complete linkage hierarchical clustering outperform the top-down algorithm and even in some cases the Gower distance usage or equating it, i.e: Credit, Dermatology, Zoo data sets. The lack of accuracy of the top-down clustering algorithm can be explained by the splitting condition, too simplistic, and maybe the theoretical model, lacking sensitivity. Hence, we believe that by refining the mathematical expressions, adding for instance a prior on the variance for gene expression profiles and adding an informative prior following a Dirichlet distribution for the clinical knowledge, would improve the accuracy. Some other techniques could be used to obtain a numerical approximation rather than an analytical expression of the likelihood, like MCMC sampling (Gibbs, Metropolis-Hasting). Furthermore, finding a relevant splitting scheme at each step of this algorithm, relying on the behaviour of the marginal likelihood, would also lead to better results. Concerning the marginal likelihood matrix, updating such matrix at each new cluster association should also grants to this method better outcomes, but then, as we highlighted it in section 4.3, the time complexity would go through the roof. We strongly believe that a Python or Java implementation rather than R, could provide a significant time consumption improvement, and such languages should also allow a more practical object/data manipulation. Thus, tuning the mathematical model and working on the optimisation of the algorithms could potentially yield to interesting and attractive new methods for cluster investigation.

6 Conclusion

Although further refinements are needed to reach applicable methods, we managed to propose an interesting framework for clustering investigation using Bayesian inference. The lack of accuracy from our implementations prevented us from moving our analyses forward with adenocarcinoma data sets. Without the ability to cluster accurately our data, we preferred to restrict ourselves concerning any interpretation of the clusters. Nonetheless, this paper can provide a good basis for future work, we also gave some hints about possible improvement to such model. The ability of such model to be applied to any practical use makes it attractive and subject to future enhancement, indeed, the probability distribution can be easily tuned to match the needs and efficiency and readability of hierarchical clustering has already been proven over the years. Several algorithmic approaches were adopted, hence, we proposed three different hierarchical algorithms, written under R, and a simple model comparison tool, all the source code is available. On a more personal note, working on Bayesian inference was a great learning opportunity, given the popularity and increasing interest for such statistical scheme, and also represented a challenge given the strong mathematical requirements to understand and implement it. The occasion to freely develop algorithms was also a first, and gave its fair share of programming challenges as well as a personal opening to unexplored fields of computer sciences. Concerning Japan, my experience here has been a real pleasure, socially as well as culturally, producing unforgettable memories. It truly represents a chance and a great life experience, the once in a lifetime kind, and I hope to come back in this beautiful and welcoming country as soon as possible.