Master Project

Presentation and study of robustness for several methods to classify individuals based on their gene expressions

Author: Julien Damond
EPFL

Professor: Prof. Stephen Morgenthaler
EPFL

Supervisor: PhD Sahar Hosseinian
Diagnoplex

june 2011

Contents

Abstract

Motivation: Several studies have shown that it is possible to detect cancer tissues based on gene expressions using methods of machine learning. The main problem with classifying gene expression data is to obtain accurate rules that are easy to interpret and provide indications for follow up studies. Indeed high accuracy is hard to achieve due to the small number of observations and the large amount of genes in the human genome. Some methods of machine learning are based on an important quantity of genes, which lead to decision rules that are usually difficult to interpret.

These methods were tested on different samples and their results were compared. Most of them provided good results with a high accuracy (see [1] and [2]). Among these methods for gene classification one distanced itself from the others by producing transparents results which were readily interpretable and were very useful for follow up studies. It highlighted pair of genes that were the most efficient to classify individuals with respect to their gene expressions. This is the so called Top Scoring Pair (TSP) classifier.

This method achieves prediction rates that are as high as those of the other methods. In contrast to other classifiers which use considerably more genes and more complicated procedures, the TSP has an easy and quick implementation and involves very few genes, namely only two. This provides very easy rules that are accurate and transparent. Finally, the TSP is paramter-free, which avoids overfitting and inflation of the estimation of the prediction rate.

Results: In this paper we will present the TSP classifier, give its definition, explain how it is constructed and we will also present the procedure used to classifiy new observations. We will study the robustness of this method, how the results (in this case, the provided pair of genes) are affected by modifying the training set. Firstly we use bootstrap methods to simulate datasets in order to analyse the stability of the method. Secondly we study the robustness in a mathematical way through the definition of the TSP.

We will also present an extension of the TSP, namely the k-TSP. This method is based on the same idea as the TSP but involves more pairs of genes. We will compare the robustness of the two methods. Then we will briefly introduce another extension, the WTSP which adds weights to the k-TSP. Finally we will present a widely used method, the penalized logistic regression. The goal is to compare these methods on specific datasets and draw advantages and disadvantages of these methods.

1 Introduction

In biological studies it is common to work with microarray data. With the recent improvement in biological technologies, this technique has become easier to use and is relative cheaper. For this reason it has been widely used to examine the possible discrimination of cancer samples from normal ones. It allows the user to obtain a large amount of measured genes, more precisely it is common to have thousands of genes. Through analysis of gene expressions, some genes are found to be highly correlated with the sample tissues we analyse; we call these genes biomarkers. They can be used in a lot of biomedical applications, the main example is the prediction of cancer based on gene expressions. The real challenge in such situation is to find decision rules which have a high accuracy on the classification and a meaningfull interpretation. Unfortunately such classifiers are often very sensitive to changes on the data set, since in different studies of the same cancer, different biomarkers can be selected. This lack of stability (in this case stability refers to the sensitivity of the selection procedure to perturbation of the training set) is mainly explained by the lack of data. Indeed it is often the case in such studies that the number of samples compared to the number of genes remains quite small and can be around one hundred. This leads to estimators with high variance. This problem is known as the *small N, big P problem*. One of the undesirable properties of these kind of problems is the computational difficulties. Indeed, as the size of the dataset increases, the required time to compute the estimation increases as well and can be very long even using computers. One solution is to consider a dimension reduction, whose goal is to remove the variables (genes in this case) that are useless for the study. A solution to reduce the variance of the estimator is to increase the sample size. This can be made through joining datasets from different studies, this would increase the number of observation and thus reduce the variance of the estimators. Joining datasets is presented by Geman et al in [3]. However, one must be carefull using this method. Indeed microarray results can strongly depend on the technologies used, such as spotted cDNA and Affymatrix arrays which cannot be directly compared. A lot of factors can also affect the results, for example the generation of microarray, alternative experimental protocols, experiment parameters, sampling of different patient populations, etc.

A lot of methods are already available for analysing such kind of datasets, especially in the so-called learning machine field. The most well known method, called Support Vector Machine (SVM) uses all the genes to perfom the analysis. This method provides good results but is very hard to interpret. In this paper we will present a method called Top Scoring Pairs (TSP) classifiers, the original method seeks pairs of genes whose intensities are ordered in a different way with respect to the group from which the measures come from. This method provides good results based on only two genes. One of its advantages is that the results are easily interpreted and transparent and they also provide follow-up for other studies by indicating which gene could be important in a specific disease.

We will study the stability of the TSP, namely how changes in the datasets can affect the genes pair provided by the method. We will proceed in two ways. In the first way we use boostraping to reproduce datasets that "look like" the original data but slightly different and apply the TSP on it. We will perform 500 such bootstraps and compute the frequency of appearence of the TSP computed on the original dataset among the TSP computed on the bootstraped dataset. The second way is more formal, we will use the mathematical definition of the TSP to study the importance of one single observation on the stability of the TSP. The influence of a single observation will also be studied through the notion of sensitivity curve, which will be applied on the special case of the TSP. We will also present the notion of breakdown point which measures the percentage of data that can be modified such that the method still provides the same genes pairs (this is an adaptation of the traditional notion of breakdown point).

We will present a first extension of the TSP, namely the k-TSP. It has the same strategy as the TSP but works with more than only one pair of genes. The method researchs for the k best pairs of genes based on the same principle as for the TSP. The decision rule of an observation is based on a voting system funded on the prediction of the k pairs of genes selected by the method. In this paper we proposed two solutions to determine k. The first one is based on the crossvalidation whose goal is to minimize the error prediction rate. The second one is based on a score used to derive the methods, its goal is to make the estimation of the value of k less sensitive to perturbation in the dataset. Atought the numer of genes has been increased by a factor k, it remains small enough to keep the nice properties of the TSP on the easiness of interpretation of the results and the usefulness of the selected genes pairs for follow up studies.

We will study the robustness of the k-TSP in the same way as for the TSP. We will use bootstrap resample and apply the k-TSP on each step in order to study the sensitivity of the method. We will also investigate the mathematic properties of this method. They will be quite similar to the one of the TSP, except we will have to deal with several pairs instead of a unique one.

Then we will briefly present another extension of the TSP, the WTSP. It is similar to the k-TSP but is based on ratio of genes expression instead of differences. We mention this method as it is another interesting way to study the relative ordering expression of profiles within different groups. We will not make any investigation on this extension, neither implement it.

After that we will present a last method, whose goal is to explain the response variable (the group of the observation) based on a linear combination of the variables (here the genes). This relation can be defined through a function, chosen to increase the performance of the method. As mentioned before, the biological datasets may contain a lot of variables, to deal with this problem a penalization is added on the number of parameters in order to deacrease the number of non zero parameters. This will reduce the quantity of genes used and make the interpretation easier. This method is knows as the penalized logistic regression.

Finally we will use two datasets to compare the methods presented along this paper, and discuss their advantages and disadvantages based on the results obtained on these two datasets.

2 The Top Scoring Pair Classifier

In this section we define the Top Scoring Pair (TSP) classifier. We begin by introducing the notation of the data set. Then we will present a first score, which will be used to decide which pair will be considered to be the best one. In order to break ties (several pairs that achieve the maximum score) we will define a second score which will allow to select only one pair of genes. Finally we will present the decision rule to classify a new sample.

In the second part of this section we will study the robustness of the TSP using bootstrap resamples. We will use it to analyse the sensibility of the methods to perturbations produced on the dataset. We will also perform a mathematical analysis of the TSP based on its defintion to determine the influence of one observation, and compute if it makes the TSP to select another pair. We will also derive a function that allows the user to compute the lowest number of observations one can add without producing changes on the results of the TSP. We will discuss robust notion of the TSP, as the breakdown point and the sensitivity curve.

2.1 Presentation of the Top Scoring Pair Classifier

The first point of this section is to define the notation used for the TSP. We will keep this notation as we will introduce other methods and discuss the results.

2.1.1 Notation

Consider a gene expression profile consisting of P genes labeled as $\{g_1, g_2, \dots, g_P\}$ and assume there are N observations $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, where $\mathbf{x}_n = (x_{1,n}, \dots, x_{P,n})$ represents the expression values of the P genes for the observation n. These data can be represented as a matrix of dimension $P \times N$ in which the expression of the i-th gene, $i \in \{1, \dots, P\}$, from the j-th observation, $j \in \{1, \dots, N\}$, is denoted by $x_{i,j}$. In this setting the columns represent the gene expressions of the samples for an observation.

We define the set of possible class labels as $C = \{C_1, \dots, C_M\}$ and the class for the observation j is denoted by y_j, where $y_j \in C$. We assume for the moment that $M = 2$. For example C_1 is the class for observations from people with cancer and C_2 from healthy people. We will extend the problem to multi class classification in a later chapter. The whole training set is expressed as $S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$. We assume that the expression's profile and its class label are random variables. We denote by $\mathbf{X}$ the gene expressions and by $\mathbf{Y}$ the class label. We assume that the elements of S are independent and identically distributed observations. Here we labeled the genes as $\{g_1, g_2, \dots, g_P\}$. The labels are not always available (especially for the last datasets we will present), for this reason we also use the number of the line in the matrix X which stands for the gene g_i as its names, in other words, the gene g_i will be labeled as i.

The TSP classifier is a rank-based classification method, more exactly the decision rules depend only on the relative ordering of the expression values within each profile. This should not be confused with rank based methods for determining differentially regulated genes. In such methods we are interested in genes that have different expression values between two population. It is possible to use ranked expression value for a fixed gene among all observations, this is not the purpose of this paper. Here, the expression values of the P genes are ordered within each profile.

The first step of the TSP is to transform the data matrix into a matrix which contains the rank of the expression values within each profile. They are ranked with respect to their expression value, the most expressed value will be ranked top (obtain the highest rank N) and the least expressed ranked last (obtain the lowest rank 1). We define the matrix of the rank as R, where $R(i, n)$ is the rank of the i-th gene for the n-th observation. The reason why we use the rank matrix will become clear as we introduce the second score for the gene pair which it is based on the ranked values of the genes within each profile. We remark that the part based exclusively on the first score could have been done using the matrix of values as well as the rank matrix.

2.1.2 The score Δ

The goal of the TSP is to find a pair of genes that best discriminates between the two groups with respect to the relative rank of the expression of these two genes. To state this more mathematically we want to find a pair (g_i, g_j) such that we have $R(i,n) < R(j,n)$ with a high probability for individuals of group C_1 (resp. C_2) and a low probability for individuals of group C_2 (resp. C_1). We define these probabilities by

$$p_{ij}(c_m) = \mathbb{P}\big(R(i,n) < R(j,n) | y_n = c_m\big), \; m \in \{1,2\}. \tag{1}$$

We note that the event $\{R(i,n) < R(j,n)\}$ and $\{x_{in} < x_{jn}\}$ are exactly the same. So the probability of each of these events will be the same, thus, at this stage, it is equivalent to work with the original matrix or with the matrix composed of ranks.

The probabilities $p_{ij}(\cdot)$ are estimated by the relative frequencies of occurrence of the event of interest. Mathematically this is expressed as

$$p_{ij}(C_i) = \frac{|\{n \in C_i : R(i,n) < R(j,n\}|}{|C_i|}.$$

If the value of $p_{ij}(C_1)$ is big (close to 1), then the occurrence of the event $\{R(i,n) < R(j,n)\}$ is high and thus people from group C_1 will often tend to have values of the pairs (i,j) such that $R(i,n) < R(j,n)$. On the other hand if this probability is small (close to 0), the event $R(i,n) < R(j,n)$ will occur less often.

We can restate the goal of the TSP as finding a pair of genes such that $p_{ij}(\cdot)$ is big for the group C_1 and small for the group C_2 (or inversely small for the group C_1 and big for the group C_2). These values will highlight the capacity of the pair of genes to classify correctly the individual to one of the groups. Indeed if $p_{ij}(C_1)$ is big, this means that on a big proportion of people of group C_1 we observe $R(i,n) < R(j,n)$. Inversely this events will occur rarely for people from group C_2, a big value of $p_{ij}(C_1)$ and a small value of $p_{ij}(C_2)$ will result in a good classification of the individuals with the pair of genes (i,j). This idea is the main concept on which the TSP relies. We use the absolute difference of this probabilities as the power of the pair of genes and call it the "score" of the pair (i,j). We thus define the score as

$$\Delta_{ij} = |p_{ij}(C_1) - p_{ij}(C_2)|. \tag{2}$$

The value of Δ_{ij} gives an indication on the accuracy of the pair (i,j).

Let us suppose that the selected gene pair is (i, j) and that $p_{ij}(C_1) > p_{ij}(C_2)$, then $\Delta_{ij} = p_{ij}(C_1) - p_{ij}(C_2)$. We wish that Δ_{ij} is big, so $p_{ij}(C_1)$ need to be high (close to 1) and $p_{ij}(C_2)$ low (close to 0). Having $p_{ij}(C_1)$ close to 1 means that for a big proportion of individual from group C_1 the event $R(i,n) < R(j,n)$ occurs often and $p_{ij}(C_2)$ close to 0 means that for a low proportion of the population from group C_2 the event $R(i,n) < R(j,n)$ occurs rarely. As the classification rules are based on $R(i,n) < R(j,n)$ or $R(i,n) > R(j,n)$ (see section on the classification), a value of Δ_{ij} close to 1 will provide high accuracy (assuming the dataset represents well the true expression of the genes).

The TSP will compute the score for every pair of genes and choose the pair that achieve the highest one. It is not enough to compute only the score, it is possible that several pairs of genes achieve the maximal score. In order to break ties and pick only one pair, a new quantity is introduced.

2.1.3 The average ranking difference Γ

In order to break ties for pairs achieving the maximum score we introduce the “average ranking difference”. We define it, for each pair (i, j), as

$$\gamma_{ij}(C_m) = \frac{\sum_{y_n = C_m} \big(R(i,n) - R(j,n)\big)}{|C_m|}, \ m \in \{1, 2\},$$

where $|C_m|$ is the number of observations that belong to the class C_m. The average ranking difference is specific to each class, it measures how "far" (in term of rank) are the expressions of the two genes within each group. The TSP will be a good method if it classifies well the patients. The score already introduces this notion, but we can add a new notion that makes the TSP classifiers even better. We wish the groups are well separated. For the TSP, it means that the difference between the two genes expression within each group is big (in absolute value) and we also want it to be very different from the other group (of opposite signe). The difference being a real number, an intuitive solution would be to ask that the average ranking difference is high and of opposite sign from one group to another, so the genes would have a very different relative ordering in each group. We express this notion mathematically as follow

$$\Gamma_{ij} = |\gamma_{ij}(C_1) - \gamma_{ij}(C_2)|, \tag{3}$$

this quantity is called the “rank score”.

If several pairs achieve the maximum score over all scores, the rank score is used to break ties and allows to pick only one pair.

The decision to use the rank matrix instead of the orignal data matrix had no influence until the introduction of the rank score Γ. Indead the probabilities $p_{ij}(\cdot)$ were not influenced by this modification because the events would contain the same individuals. But this modification has a big impact on the rank score Γ, in fact it robustifies this second score. Without this modification, an outlier with a big (positive or negative) value will make the rank score explode and thus making us to choose the corresponding pair of genes among all the pairs achieving the same score Δ_{ij}.

2.1.4 Classification of a new observation

So far we have defined the TSP. Now we will explain how to classify a new observation with this method. Given that the provided pair of genes is (i, j), we need only to compare these two genes for the new patients denoted by $\mathbf{x}_{n+1}$. We observed that it was equivalent to work with the original data or with the ranked values. For the derivation of the method it was more comfortable to work with the ranked matrix. However, for the prediction of a new observation it is more direct to work with the expression values. For the resulting pair of genes (i, j), we only need to compare the expression of the genes i and j for the new patient. Without loss of generality we can suppose that $p_{ij}(C_1) > p_{ij}(C_2)$. In the case $x_{i,n+1} < x_{j,n+1}$ we assign the new individual to the group C_1 and to the group C_2 otherwise. We can also write more formally

$$h_{\text{TSP}}(\mathbf{x}_{n+1}) = \begin{cases} C_1, & \text{if } x_{i,n+1} < x_{j,n+1} \\ C_2, & \text{otherwise} \end{cases} \tag{4}$$

The decision rule should be inversed if $p_{ij}(C_1) < p_{ij}(C_2)$.

To summarize, the TSP is a classifier based on the expression of genes; it searches gene pairs whose expressions are inversely ordered between the two groups. It has two parameters to score a pair, the score Δ_{ij} and in case of tie break, the rank score Γ_{ij}. It classifies a new observation to the class to which the ordered values match with the highest probability $p_{ij}(C_1)$ or $p_{ij}(C_2)$.

2.2 Multi-class classification

For the moment we only dealt with binary problem classification. It is common to have to handle with more than two classes. The TSP family can easily be extended to such problems. The strategy is to decompose the procedure into several steps in which the usual TSP method can be applied. We briefly present three common procedure.

We suppose we have to deal with M classes and we denote the set of multiple classes by $C = \{C_1, C_2, \ldots, C_M\}$.

2.2.1 One-vs-Other

The One-vs-Other (1-vs-r) approach decomposes the original problem into a set of M binary problems. For each class $m = 1, 2, \ldots, M$ we need to construct a TSP to distinguish between the class C_m and the class composed of all the other classes $C \backslash C_m$. To predict the class of a new observation, we need to evaluate each of these M classifiers, which results in a set of M predictions each of these choosing either a single class or a class composed by $M-1$ classes. We are interested exclusively in the predictions of single classes and ignore the one for composite classes. If only one observation to a single classe is available, we classifie the observation to this class. If the M predictions contain several single classes we choose the one that contains the more observations. If no single classes were pointed out we assign the observation to the biggest class.

2.2.2 One-vs-One

The second approach we present is called the One-vs-One (1-vs-1) scheme. Much more calculations are needed for this method. In fact we aim to make comparisons between all possible pairs of classes. For every pair of classes (C_l, C_m) with $m \neq l$ a binary classifier h_{lm} is constructed based only on the training set composed of these two classes. Consequently $M(M-1)/2$ binary classifiers are generated, each predicting exactly one of the classes. We combine the predictions by counting the number of prediction for each class and assign the observation to the class that gets the highest number of predictions.

2.2.3 Hierarchical classification

The last procedure we present is the hierarchical classification (HC). It is a sequential procedure based on a tree. On the first node of the tree a binary classifier h_1 needs to be constructed and it has to distinguish between the largest class and the class composed of all remaining classes. On the second node a new classifier h_2 is constructed and it distinguishes between the second largest class and the remaining classes (the single classes from the previous steps are removed). And so on until all classes are represented on the tree. Each single class represents a leaf of the tree, these beeing seen as the labels. To classify a new observation we need to make it go through the whole tree until it reaches a leaf whose label will be the prediction of the new observation. At each step we need to evaluate the observation with the current classifier. If it chooses a single class, the classification ends and outputs the label of the leaf as prediction, otherwise we have to go down in the tree and evaluate the next classifier, and so on until a single class is chosen.

It is totally arbitrary to choose one of this methods. They may produce different results and the goodness may be highly correlated to the kind of dataset one analyses.

Along this paper we will deal with two class problems. But in the last chapter we will present a dataset, which will contain three classes. We will choose one class as a reference and generate three TSPs, two against each of the two remaining classes and one against the class composed of the merged remaining classes. We will discuss this procedure with more details in the related chapter.

2.3 Robustness of the TSP

The main goal of this paper is to study the robustness of the TSP. More exactly we want to see how an estimator is affected by small departures from the model's assumptions. In our case we are interested in the robustness of a method, namely the TSP. We saw that this method is based on the computation of two quantities, the score Δ and the rank score Γ. The first point in studying the robustness of the TSP is to analyse if these two quantities are robust or not. It is obvious that if they aren't, the chances that the method is robust are small. The second point is the analysis of the robustness for the method.

There are several ways to proceed, one way is to see the influcence of only one observation on the results provided by the method. It is common to add an outlier x_0 to the data and to investigate its influence on our estimator. We can also study the impact of the outlier x_0 on the estimator but in this case as a fonction of x_0. This notion is often caracterised by the influence function $\mathrm{IF}(x_0, F)$, where $F(\cdot)$ represents the underlying distribution of the data. Another important quantitiy is the breakdown point (BP), which measures the largest proportion of atypical point that the data set can contain such that the estimator still gives information about a paramter. We will give more details about these notions in a later section and adapt them to the TSP.

In order to study the robustness of the TSP we will proceed in two steps. In the first step, we perform the TSP on a dataset where tissue samples were analysed from healthy people and from people affected of breast cancer. The goal of the original study was to classify patients based exclusively on their gene expressions. Then we will use bootstrap resample of this dataset to obtain several new datasets which are similar to the original one and then we will compute the TSP on each of the bootstraped samples. Finally, after repeating the experiment a large number of times, we computed the appearance frequency of the original TSP among all TSPs generated from each bootstraped sample and reported this frequency as well as the pairs of genes.

In the second step, we examine the robustness of the estimator of the score Δ and the rank score Γ. We will present in details the notion of influence function and breakdown point in a general point of view. We will compute these functions for the two quantities, which will allow us, in particulary, to determine the influence of a new observation on the computation of these quantities. Then we will examine the robustness of the TSP through these two quantities and propose an adaptation of the influence function and the breakdown point especially for the TSP. We will compute the adaptatation of these two notions and apply them on the specific case of the leukemia cancer dataset.

2.3.1 Study of robustness through simulation

In this section we study the behaviour of the TSP as the dataset is modified. We based our computations on a public available dataset named **Leukemia cancer dataset**, this dataset has been widely used in many papers and can be found at *http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi.* This dataset was used to infer on prediction of leukemia cancer based on gene expression profiles. It is clearly presented and analysed in the paper from *Golub et al.* in [4]. The results of this paper highlighted the feasibility of cancer classification based uniquely on gene expresssion measures. The data consists of gene expression profiles measured in leukemia tumor samples. The training dataset contains 38 bone marrow samples from two classes, labeled as *AML* for acute myeloid leukemia and *ALL* for acute lymphoblastic leukemia, the number of observations in each class is 27 for ALL and 11 for AML. A test set was also available, it consists of 30 samples with 20 ALL and 14 AML. To be consistant with the analyse made on dataset which we will present later, we merged the training and the test set to have 78 observations with 47 patients in group ALL and 25 the group AML. The number of genes was 7129.

Filtering Due to the extremely high dimension of the data set, the mathematical computations were extremely long and sometimes ran out of space for specific calculations. We reduced the number of genes and thus worked only on a subset of genes instead of on the whole dataset. This reduced considerably the computational time. A lot of methods to reduce the datasets exist, they are part of data mining and a lot of litterature about them is available. Among these methods we can cite sophisticated statistal methods like clustering, principal component analysis, etc. There is no best method, one can be in certain case better and in other case worse,they all depend on the analysed dataset. The goal of this paper is not to reduce the dataset the most efficiently but to study the robustness of TSP. The method we used is totaly arbitrary and other methods could be used as well and could be better. Here, the goal of the reduction is mainly to decrease the size of the dataset in order to make the computations easier and faster.

Given the fact that the TSP uses only genes that have a significantly different relative ordering expression whether it comes from an infected patient or an healthy one, we decided to keep only genes that have significant different means between the two groups. We conducted this by first using a Shapiro test to test the assumption that the genes expression come from a normal distribution or not. In the first case we used a Welch's test (an unpaired t-test without the assumption of equal variance) to test the difference in means of the two populations. In the second case we used a Wilcoxon t-test. This lead to an important reduction of the dataset. Indeed the number of genes went down from 7129 to 2691, which makes the dataset much more comfortable to deal with and the computations much faster.

Simulation The aim was to simulate new datasets from the original dataset which were similar, to compute TSPs on the resulting datasets and to study the sensitivity of the method to changes in the dataset. We used bootstrap resample from the original dataset to create new datasets. Bootstraping is a computer-based method that measures the accuracy of a sample estimate as well as properties of an estimate like the variance, confidence intervals, etc. A number of steps need to be defined, which stand for the number of dataset we want to generate. At each step we create a new dataset from the original one, where each observation has the same probability of being chosen. It is posssible that an observation doesn't appear in the new sample as well as it can appear several times. The size of the bootstraped dataset was chosen to be the same as the original dataset.

This method is very simple and its computation is very quick. It is also very good and was shown to provide good results for high number of resampling. In our simulation we choose n to be equal to 500, which is large enough for accurate results.

Results For each of the 500 bootstraps we computed the TSPs and stored the selected gene pairs. We are also intersted in the appearance of the single genes, because this can be useful to determine the importance of a single gene in the detection of the specific cancer tissues. It can be further used if we are interested in classification based on single genes or if we look for a set of genes correlatd to the disease. Figure 1 shows the frequency of appearance for each genes among all gene pairs. We keept only genes that had a significant frequency of appearance relatively to the other pair ($> 8\%$).

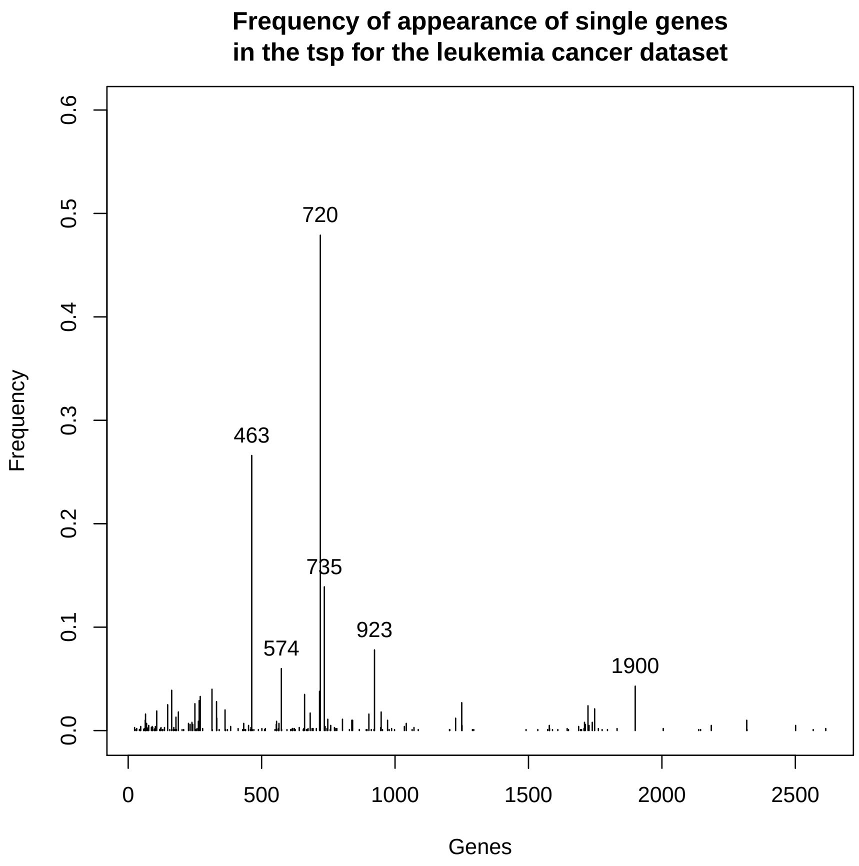

Figure 1: Frequency of appearance of the single gene in the pairs selected by the TSP over 500 bootstrap resamples from the dataset **Leukemia cancer**, only the genes whose appearance where above 8% where hold.

The genes 463, 574, 712, 735, 923 and 1900 are the genes that appear the most often in the pair of genes selected by the TSP over the bootstrap. If one would be interested on a subset of genes correlated with the disease, the TSP would provide such a set. This set could be improved by using other methods since the

TSP used in this way would only provide a clue about which genes should the set contains.

The most interesting part of the simulation is the frequency of appearance of the pairs selected by the TSP as we focus our study on the robustness of the method. We are interested in the freqeuncy of appearance of the original gene pair of the TSP in the bootstraped datasets. Figure 2 presents the resulted TSP over the bootstraped data as well as the number of times the gene appeared. We keept only pairs that had a significant frequency of appearance relatively to the other pair ($> 5\%$).

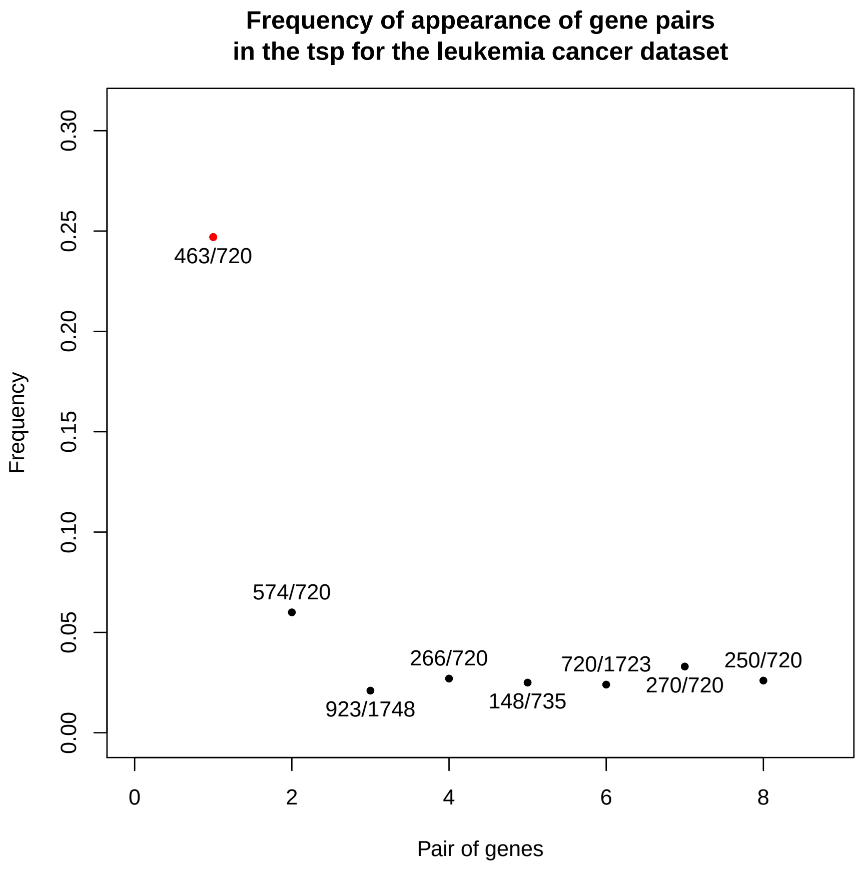

Figure 2: Frequency of appearance of the pairs of gene selected by the TSP over 500 bootstrap resamples from the dataset **Leukemia cancer**, we ploted only pairs whose frequency was at least 5%. The pair with the red dot represents the pair selected by the TSP on the original data set.

Figure 2 shows the frequency of appearance of the pair of genes that appeared the most often in the bootstrap of the Leukemia cancer dataset. We see that the pair $(463, 720)$ which was detected by the TSP on the original dataset appears often among the TSPs computed on the bootstrap, indeed this frequency is about 25% of the time and is much more bigger than the second best frequency of appearance. Which is attaigned by the pair $(574, 720)$ and is about 7%. The other pairs of genes computed by the TSP on the bootstrap have a frequency about 3%. Finally we observe that the genes 720 appears in three quarter of the pairs, which makes it very interesting for the study of this dataset because of its high correlation with the disease. On the other hand Figure 2 showed also that the gene 1900 was of interest, but it doesn't appear in the pair of genes for the selected threshold of appearance.

Figure 3 shows the ROC curve for the TSP applied on the leukemia cancer dataset.

These results showed good properties of the TSP. We want to make the reader sensible to the fact that they hardly depends on the dataset we used. We briefly introduce another dataset and analyse the sensibility of the TSP to changes in the same way as for the leukemia cancer dataset.

This dataset is based again on tumor sample, but this time from **breast cancer**. It contains 78 observations, which consists of patients who developped a distance metastase within 5 years (they were labeled as "diseased") and patients who remained healthy for at least 5 years after their original diagnosis (they were labeled as "healthy"). We have 34 "diseased" patients and 44 "healthy". The number of genes was 24481. We did the filtering and the bootstrap in the same way as for the leukemia cancer dataset.

The TSP computed on the original dataset of breast cancer selected the pair of genes (541/2343). Actually the maximum score Δ was achievied by two pairs. The tie was separated by the second score Γ in order to pick one pair. Figure 4 shows the frequency of appearance of the pair of genes that appeared the most often. We see that the pair (498/2333), which was ranked second on the original dataset, appears more often than the pair that was ranked first. Inversely the top ranked pair from the original dataset is ranked second with the bootstrap resampling. The top pair on the boostrap sample appears slighty more than 8% of the time, whereas the frequency of the top pair based on the original dataset appears is approximately of 7%.

There are other pairs of gene highlighted by the bootstrap which were not detected by the TSP. There were 5 pairs which appear around $2 - 3\%$ of the time. In total this represents about 10% of the total frequency of the bootstrap.

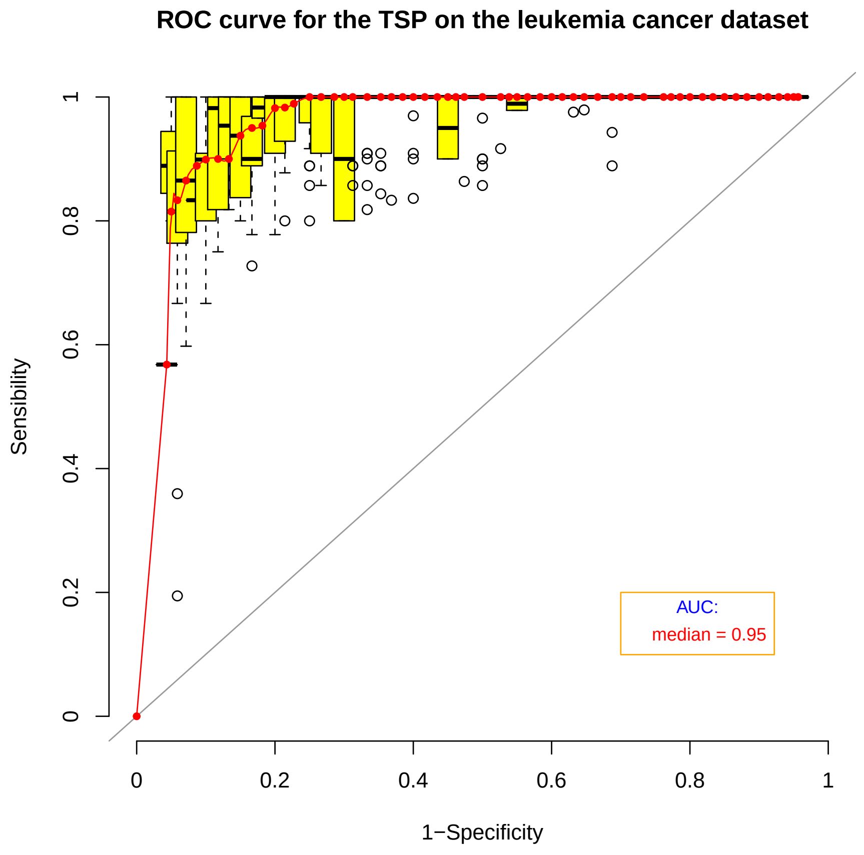

Figure 3: ROC curve for the TSP on the leukemia cancer dataset. The red line represents the median of the sensibility. The grey line stands for the line with interecept 0 and slope 1.

If we add the frequencies of the 7 gene pairs selected (whose frequencies are above 0.02%) we obtain only 25% of the total frequency, which is a very low percentage. Indeed, this means that 3 times over 4 a perturbation of the dataset leads to pairs that doesn't belong to the selected pair of genes (with the threshold) over the boostrap.

We see on the second dataset that the TSP is much more sensible to perturbations than on the first dataset. The pair provided by the TSP on the original dataset appears much more often among the bootstrap made on the first set than among the bootstrap made on the second set. This sensitivity highly depends on the setup of the dataset. We saw that, on the second dataset, two pairs of genes reached the maximum score Δ, which makes this two pairs very competitive to each other.

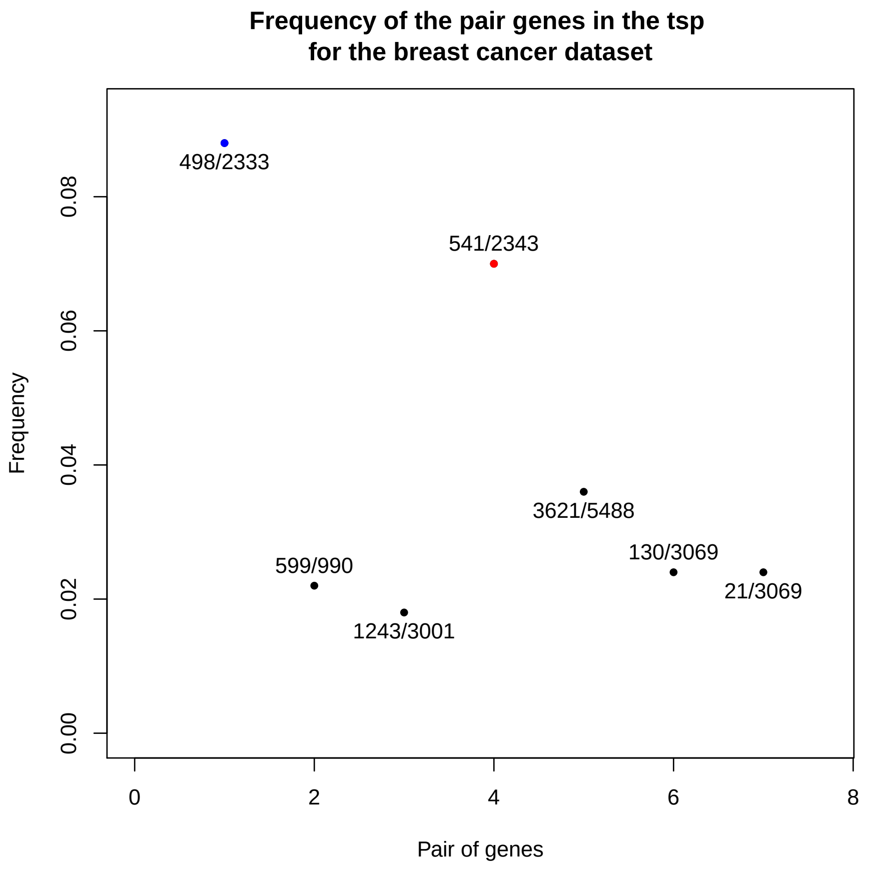

Figure 4: Frequency of appearance of the pairs of gene selected by the TSP over 500 bootstrap resamples from the dataset **Breast cancer**, we ploted only pairs whose frequency was at least 2%. The pair with the red dot represents the pair selected by the TSP on the original data set. The pair with the blue dot achieved the same score Δ as the top pair but was ranked second with respect to the score Γ.

This problem will be analysed with more precision in a later section.

The simulation provided in this section highlights the power of the bootstrap, it allowed us to obtain a large amount of datasets based on an original dataset and showed us how the TSP reacted to changes in the dataset. We note that the changes were totally arbitrary, every observation could have been removed or observed several times. We actually observed lack of robustness of the TSP for low perturbations as well as for important perturbations in the dataset.

In a later section we will analyse the robustness of the TSP through its mathematical definition. But before we will present some basic notions of robustness in a general meaning. Then we will adapt them to the special case of the TSP in order to see why and how changes in the data can modify the pair selected by the TSP. Finally, we will derive a function to calculate the number of observations that can be added (and removed) from the dataset leaving the TSP unchanged. This function will be a function of the setting of the dataset (the number of observations, the number of observations in each group, the size of the groups,etc).

2.3.2 Basic notion of robustness

The main goal of statistic is to analyse different kinds of datasets and to be able to draw useful estimations from them in order to learn from what we observed and to allow predictions with a certain amount of accuracy (often also estimated from the data). The main weakness of most estimators is their reaction to perturbations on the dataset, i.e., they stay in a neighborhood of the original estimation or if they hardly differ when adding an atypical observation to the dataset. A field of statistics deals with such problems, it studies how an estimator reacts to changes in the dataset and provides new estimators which are supposed to be less sensitive to perturbations. This branch of statistics is relatively new to the other fields but it gained a lot of importance in the last 30 years. This field is the so called *Robust statistics* and seeks to provide methods that are competitive with popular statistic methods but which are less affected by outliers or small derivations from the model's assumptions.

In this section we present two basic notions of robust statistics. The first notion is based on the sensitivity of an estimator to a single observation, which represents the influence of adding a new observation to the dataset. It measures the variation of the estimator as a function of the single observation added to the dataset. The second notion we present computes the largest proportion of atypic points that the dataset can contain such that the estimator still yields accurate information about the parameter of interest.

Sensitivity curve and influence function The first notion we present is called the *Sensitivity Curve*, and as its name shows, it deals with sensitivity. An important point in robustness is to compute how a new observation added to the dataset can modify our estimations. The sensitivity curve is based on this point of interest. Let us denote by $\hat{\theta}_n(x_1, \ldots, x_n)$ the estimation of the paramter θ based on a sample of size n composed of the observations $x_1, \ldots, x_n$. We write x_0 as the new observation.

The sensitivity curve (SC) for the sample of size n composed of $x_1, \dots, x_n$ is defined as follows

$$\begin{aligned} \mathrm{SC}_n(x_0) &= \frac{\hat{\theta}_{n+1}(x_1, \dots, x_n, x_0) - \hat{\theta}_n(x_1, \dots, x_n)}{1/(n+1)} \\ &= (n+1)\big(\hat{\theta}_{n+1}(x_1, \dots, x_n, x_0) - \hat{\theta}_n(x_1, \dots, x_n)\big). \end{aligned} \tag{5}$$

The sensitivity curve computes the difference of the estimation of θ based on the original dataset plus the new observation x_0 and the estimator based exclusively on the original dataset. This quantity is then scaled by the number of observation of the largest set $(n + 1)$. It represents the influence of the observation x_0 on the estimator of θ.

The influence function is calculated for a finite n, we note that there exists an extension to this quantity as n tends to infinity. This is the so called *Influence Function*, which measures the influence of an observation x_0 as n tends to infinity. In this paper we will concentrate us only on the sensitivity curve, for more details about the influence function see [5].

The next point that we will present computes the largest number of observations we can add to the dataset such that the pair provided by the TSP remains unchanged.

Breakdown point We have seen how to compute the influence of a new observation to the estimate. We will now introduce another quantity which allow us to compute the largest amount of atypical points (contamination) that the data may contain such that the estimate $\hat{\theta}$ will still give an accurate estimation of the parameter θ. We are interested in the proportion of incorrect observations (arbitrarily large observation) that can be contained in the dataset such that the estimator still gives accurate estimation of the parameter of interest. By accurate we mean that the estimator cannot take arbitrarily large values. As for the sensitivity curve there exist a distinction in the case of finite and unfinite sample size. Here we are interested only in the finite version as we deal with dataset for which the size n is known and finite. For more details on the unfinite case see [5].

The *finite-sample breakdown point* (FBP) of an estimate $\hat{\theta}$ for the sample $\mathbf{S} = \{x_1, \dots, x_n\}$ is the largest proportion $\epsilon_n^*(\hat{\theta}_n, S)$ of data points that can be arbitrarily replaced by outliers without $\hat{\theta}_n$ leaving a set which is bounded and also bounded away from the boundary Θ, where Θ stands for the set on which θ ranges.

We can express this more formally. For this we define $\mathcal{X}_m$ as the set of all datasets S' of size n having $n-m$ elements in common with the set S. Mathematically this is expressed by

$$\mathcal{X}_m = \{S' : \ |S'| = n, \ |S \cap S'| = n-m\}.$$

Then

$$\epsilon_n^*(\hat{\theta}_n, S) = \frac{m^*}{n},$$

where

$$m^* = \max\{m \geq 0 : \ \hat{\theta}(S') \text{ bounded and also bounded away from } \partial\Theta \ \forall S' \in \mathcal{X}_m\}.$$

We will now biefly discuss this definition. First, the set $\mathcal{X}_m$ stands for all possible perturbations of the original dataset with m observations being modified. Then it searches the largest amount of data (largest m) such that the estimation remains bounded and bounded away from the boundary in the worst case. The worst case is finding the lowest number of observations m we need to modify such that the estimation doesn't provide anymore a good information for the parameter of interest, i.e., the lowest amount of data we need to modifiy such that the estimator "breaks".

We give a quick example to enhance the intuition on this notion. Let us suppose we have a set of independant observations following a Normal distribution, i.e., $(X_1, \ldots, X_n) \overset{\text{iid}}{\sim} \mathcal{N}(\mu, \sigma^2)$ and we are intested in estimating the mean μ of this sample. An usual way is to use the maximum likelihood estimator, which is defined as

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

This estimator has a breakdown point of 0, indeed by changing any of the x_i we can make $\hat{\mu}_n$ arbitrarily large. Robust statistics suggest to estimate the mean through the median of the sample $(X_1, \ldots, X_n)$. In this case, if we change only one of the x_i's it won't be enough to make this estimator arbitrarily large. In fact the median has a breakdown point of 0.5 as we need to change at least the half of the observations to make it arbitrarily large.

Moreover we note that for any estimator it is not possible to have a breakdown point higher than 0.5. This is due to the fact that, if more than a half of the observations are contamined, then we can't distinguish between the data from the true model and the perturbated data.

In the next subsection we analyse the robustness of the quantities used for the derivation of the TSP, i.e., the score Δ and the rank score Γ.

2.3.3 Analytical study of the robustness for Δ and Γ

The TSP's procedure in selecting the pair of genes is entirely based on the computation of the quantities Δ and Γ. In order to study the property of robustness for the TSP it is necessary to first investigate this property for the score and the rank score.

Let us suppose the dataset is defined as $S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i$ stands for the gene expression of the observation i and y_i as an indicator for the group to which the sample i belongs, we write $y_i = 1$ if it belongs to the group C_1 and $y_i = 2$ if it comes from the group C_2. Let us write $\Delta = \Delta_n(S)$ and $\Gamma = \Gamma_n(S)$ to emphasis that these quantities depends on the current dataset S. We begin with the score Δ. We compute the sensitive curve and discuss the breakdown point for this parameter and then we will do the same for the rank score Γ.

Robustness of Δ We analyse the robustness of Δ for a given pair of genes. Without loss of generality we write this pair as (i, j) and we suppose that $p_{ij}(C_1) > p_{ij}(C_2)$, which defines the classification rule expressed in (4). We define the following quantities $n_1 = |C_1|$, $n_2 = |C_2|$, $k_1 = |\{k \in C_1 : x_{i,k} > x_{j,k}\}|$ and $k_2 = |\{k \in C_2 : x_{i,k} > x_{j,k}\}|$. The probabilities $p_{ij}(C_i)$ are estimated by $\hat{p}_{ij}(C_i) = \frac{k_i}{n_i}$ for $i = 1, 2$. The score becomes in this case

$$\Delta_{ij}(S) = \left| \frac{k_1}{n_1} - \frac{k_2}{n_2} \right|.$$

A new observation is characterised by the vector $(\mathbf{x}_0, y_0)$ and by definition the sensitive curve is defined as

$$\mathrm{SC}_n(x_0) = \frac{1}{n+1} \left(\left| \frac{k_1 + I\big((x_0, y_0) \in M_1\big)}{n_1 + I(y_0 = 1)} - \frac{k_2 + I\big((x_0, y_0) \in M_2\big)}{n_2 + I(y_0 = 2)} \right| - \left| \frac{k_1}{n_1} - \frac{k_2}{n_2} \right| \right),$$

where $M_1 = \{(x, y) \in S : y = 1,\ x_i < x_j\}$ and $M_2 = \{(x, y) \in S : y = 2,\ x_i < x_j\}$. We can suppose suppose that $y_0 = 1$ (the procedure is similar with $y_0 = 2$). With the assumption that $p_{ij}(C_1) > p_{ij}(C_2)$, the sensitivity curve becomes

$$\frac{1}{n+1} \left(\left| \frac{k_1 + I\big((x_0, y_0) \in M_1\big)}{n_1 + 1} - \frac{k_2}{n_2} \right| - \frac{k_1}{n_1} + \frac{k_2}{n_2} \right).$$

Moreover we suppose that the part in the absolute value is positive, so we can omit the absolute value (need to put a -1 if it is not the case), and finally the sensitivity curve is given by

$$\mathrm{SC}_n(x_0) = \frac{1}{n+1}\left(\frac{n_1 I\big((x_0, y_0) \in M_1\big) - k_1}{n_1(n_1+1)}\right).$$

We see that the sensitivity curve is, in this case, a line with a jump when $(x_0, y_0) \notin M_1$. On each side of the jump the curve takes two different values

$$\frac{-k_1}{n_1(n_1+1)(n+1)} \quad \text{and} \quad \frac{n_1 - k_1}{n_1(n_1+1)(n+1)}.$$

The graph of the sensitivity curve for the parameter Δ is shown in Figure 5. We observe that these values remain bounded (the absolute value never exceed 1), so the influence of one observation is bounded and cannot make the sensitivity curve explode.

For example on the breast cancer dataset with the assumption that $y_0 = 1$ and $p_{ij}(C_1) > p_{ij}(C_2)$, we obtain the two values $1.44 \cdot 10^{-5}$ and $-1.44 \cdot 10^{-4}$ for the influence function. On the other hand, if we suppose that $y_0 = 2$ but $p_{ij}(C_1) > p_{ij}(C_2)$ still holds, we obtain the values $3.19 \cdot 10^{-5}$ and $-3.29 \cdot 10^{-4}$. Here the division by $n_1(n_1+1)$ (or $n_2(n_2+1)$ in the case where $y_0 = 2$) reduces significantly the influence of one observation.

The computation of the breakdown point for the case of the parameter Δ is not obvious. Indeed, the parameter Δ ranges in $[0, 1]$, so the boundary of this parameter is $\{0, 1\}$. Given a dataset $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ defined as previously. If all observations from the group C_1 except one satisfies the rule $R(i, n) > R(j, n)$ and no observation from the group C_2 satisfies it, then the score would be $\frac{n_1-1}{n_1}$. If we construct a second dataset S' equal to S but where the observation from the group C_1 which didn't satisfied the rule is replaced by an observation which satisfies the rule, then the score Δ will be 1. From the definition we presented about the breakdown point of an estimator, the breakdown point of Δ would be 0 as the estimator reaches the boundary. But the estimators computed on the set S' still give accurate results about the pair (it provide perfect classification on the current dataset) and differ from the previous estimator only by $\frac{1}{n_1}$. For this reason setting the breakdown point of Δ to 0 is a severe decision. Another definition of the breakdown point should be used here in order to allow more flexibility when reaching the boundary.

During the computation of the sensitivity curve we came accross an important property of Δ which has to be highlighted. As we introduced a new observation x_0, we computed the difference of the scores $\Delta_{ij}(S) - \Delta_{ij}(S')$. This gives an indication about the influence of one observation on the score .

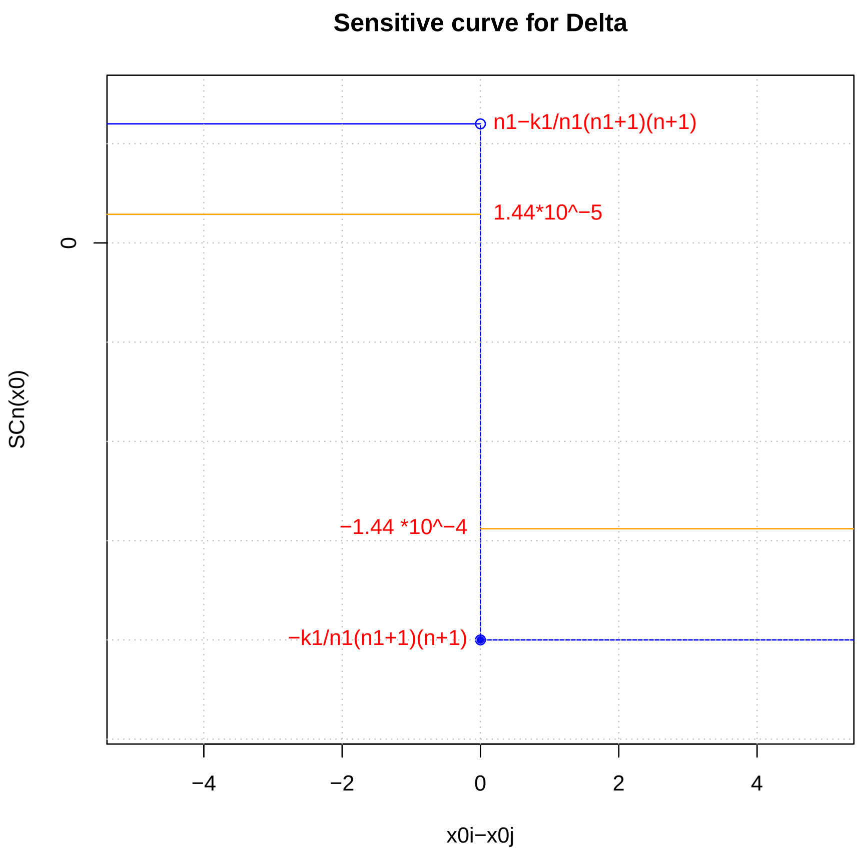

Figure 5: The red line represents the sensitive curve for the parameter Delta in an arbitrary dataset. The orange line represents this curve applied on the leukemia cancer dataset, where we supposed that$p_{ij}(C_1) > p_{ij}(C_2)$ and $y_0 = 1$

If we suppose that $\frac{k_1}{n_1} > \frac{k_2}{n_2}$ the difference is given by

$$\Delta_{ij}(S) - \Delta_{ij}(S') = \begin{cases} \frac{n_1 - k_1}{n_1(n_1+1)}, & \text{if } x_{i,0} < x_{j,0} \\ \\ \frac{-k_1}{n_1(n_1+1)}, & \text{otherwise.} \end{cases} \tag{6}$$

Here we supposed that $x_0 \in C_1$, the result is similar with $x_0 \in C_2$. In our study we worked with bootstrap to investigate the robustness of the method. In bootstrap, the notion of replacing is of interest since the size of the resample dataset is the same as the size of the original dataset. In this case where only one observation is replaced, the difference in the score is given by

$$\Delta_{ij}(S) - \Delta_{ij}(S') = \begin{cases} \frac{-1}{n_1}, & \text{if } x_{i,0} < x_{j,0} \\ \frac{1}{n_1}, & \text{if } \frac{k_1-1}{n_1} > \frac{k_2}{n_2} \\ \frac{2k_1}{n_1} - \frac{2k_2}{n_2} - \frac{1}{n_1}, & \text{if } \frac{k_1-1}{n_1} < \frac{k_2}{n_2}. \end{cases} \quad (7)$$

The difference $\frac{2k_1}{n_1} - \frac{2k_2}{n_2}$ is, because of the assumption made, quite small. Indeed we supposed that $\frac{k_1-1}{n_1} < \frac{k_2}{n_2}$ which is equivalent to $\frac{2k_1}{n_1} - \frac{2k_2}{n_2} < \frac{1}{n_1}$. Finally we can conclude that $\Delta_{ij}(S) - \Delta_{ij}(S') \leq \frac{1}{n_1}$. The case when the inequality is strict occurs when there is a change in the sign of $\frac{k_1-1}{n_1} - \frac{k_2}{n_2}$, i.e. when $p_{ij}(C_1)$ becomes less than $p_{ij}(C_2)$. This shows that the influence of one observation of the group C_1 (resp. C_1) on the score is bounded by $\frac{1}{n_1}$ (resp. $\frac{1}{n_2}$). And thus the influence of one observation is bounded by $\max\left(\frac{1}{n_1}, \frac{1}{n_2}\right)$.

Robustness of Γ Let us analyse the rank score. As a new observation was introduced to the dataset, the rank score will take into account a new element and will be modified. The sensitivity curve of this parameter is

$$\begin{aligned} \mathrm{SC}_n(x_0) \;&= \frac{1}{n+1}\Bigg(\Bigg|\frac{\sum_{y_n=C_m}\Big(R(i,n)-R(j,n)\Big)+\Big(R(i,x_0)-R(j,x_0)\Big)I(y_0=1)}{|C_m|+I(y_0=1)} \\ &\quad - \frac{\sum_{y_n=C_m}\Big(R(i,n)-R(j,n)\Big)+\Big(R(i,x_0)-R(j,x_0)\Big)I(y_0=2)}{|C_m|+I(y_0=2)}\Bigg| \\ &\quad - \left|\frac{\sum_{y_n=C_m}\Big(R(i,n)-R(j,n)\Big)}{|C_m|} - \frac{\sum_{y_n=C_m}\Big(R(i,n)-R(j,n)\Big)}{|C_m|}\right|\Bigg). \end{aligned}$$

It is obvious that this function remains bounded. Indeed the values which allow to compute the rank score are the rank of the gene expressions within the profiles. The ranks within each profile will stay bounded and can't be larger than the total number of genes, so the difference between the rank of two genes within one profile will be less than the total number of genes (so it can't become arbitrarily lage). Furthemore the number of indices on which we sum is finite. Finally we use the fact that a finite sum of bounded number is also bounded. We can use $2(n+1)P$ as a bound for the sensitivity curve of Γ, where P stands for the number of genes.

In this case again, the notion of breakdown point we presented won't yield any interesting results. Indeed, the values of Γ ranges in $[0, \infty[$. It is possible for Γ to reach 0 in the case where all the patient's genes expressions would have the same ordering in the two groups. A pair of genes that would satifies this would be unintersting for the classification procedure as the indivual's gene expression have the same order in both groups. Moreover, the value of Γ can't be made arbitrarily large as it uses a finite sum of rank values (also finite).

2.3.4 Adaptation of the basic notion of robustness for the method TSP

We defined two important notions to analyse the robustness of parameters. In this paper we are mostly interested in the robustness of the TSP, i.e., we need a notion to define robustness for a method instead of parameters. In the next section we will present an adaptation of the sensitivity curve and the breakdown point for the method TSP.

Sensitivity curve The definition for the sensitive curve that we presented is for the case of a continuous estimator. In our case we want to analyse the influence of an observation on the TSP. The TSP yields a pair of genes, which is a discrete element in $\mathbb{N}\times\mathbb{N}$. Adding an observation to the dataset may have different impacts, it can reduce or increase the accuracy rate of the TSP (if the gene pair matches or not with the classification rule determined by the TSP), or much worse, it can produce changes in the results of the TSP by using a different pair for the classification rule. In this report we are interested in the second case (altough it is highly related to the loss of accuracy, as we will see later).

We thus adapt the notion of sensitivity curve to that case. As we are intested only in changes in the resulting gene pair, it is natural to consider two events, either the pair provided by TSP_{x_0} is the same as for the TSP or the pair is a different one. We mean by TSP_{x_0} the TSP based on the augmented dataset, i.e., the original dataset including the observation x_0. The sensitive curve will be defined in the following way. We set it to 1 if TSP_{x_0} yields the same result as the original TSP and to 0 otherwise. We denote the adaptation of the sensitive curve to the TSP by SC_{TSP} and we define it as

$$\text{SC}_{\text{TSP}} = I\big(\mathit{pair}(\text{TSP}_{x_0}) = \mathit{pair}(\text{TSP})\big),$$

where the function $\mathit{pair}(\cdot)$ returns the pair of the TSP and $I(\cdot)$ represents the indicator function.

This adaptation will be very useful to study the sensitivity of the TSP to a new observation, as it will compute if the TSP will remain unchanged ($\text{SC}_{\text{TSP}} = 1$) or if another pair of gene will be selected by the method ($\text{SC}_{\text{TSP}} = 0$). We will have to be careful to the fact that SC_{TSP} will be highly correlated to the current dataset (number of observations, number of observations in each group, etc). This relationship will be analysed and precisely determined as a function of the setting in a later section.

We note that $\mathrm{SC}_{\mathrm{TSP}}$ may be easily adapted to the case where we add a set of observation to the dataset instead of a single observation.

Breakdown point Here we are interested in the breakdown point for the TSP, and as before we have to adapt this notion to this special case. We define the breakdown point for the TSP, denoted by $\mathrm{BP}_{\mathrm{TSP}}$, as the largest proportion of data that can be replaced by contaminated observations (will be defined more precisely in the next section) such that the pairs of genes provided by the TSP computed on the perturbated dataset are the same as the TSP computed on the original dataset. Mathematically this can be written as

$$\mathrm{BP}_{\mathrm{TSP}} = \frac{1}{n} \max \underset{n' \in \{0,1,\ldots,n\}}{\arg\max} \min_{S_{n'} \subset \{1,\ldots,n\}} I\big(pair(\mathrm{TSP}_{S_{n'}}) = pair(\mathrm{TSP})\big),$$

where $\mathrm{TSP}_{S_{n'}}$ stands for the TSP computed on the dataset where the elements x_i, with $i \in S_{n'}$, have been replaced by arbitrary observations.
Some words to the definition. We took the indicator function on the pairs of both TSP, the one computed on the original dataset and the one computed on the contamined dataset in order to see when the pairs of genes provided by both TSP will be different.
Second we took the minimum over all possible subsets of indices from $\{0, 1, \ldots, n\}$ because we want to see if the gene pair will change in any of the case where the number of contamination is n'. If for one subset from $\{0, 1, \ldots, n\}$ the TSP computed on the contamined dataset differs from the original gene pairs, the indicator function will be equal to 0. Thus we want that the number of perturbations allowed such that the gene pair doesn't change to be lower than the current value of n'. The value of n' won't be considered, because for n' the indicator function is 0. We note that for $n' = 0$ the indicators function is 1 and thus argmax will contain 0 and not n', so $0 \leq \mathrm{BP}_{\mathrm{TSP}} < n'$ (for $n'' > n'$ the minimum of the indicators function will be 0 because we can choose the same subset of index as in the case where the indicator function was 0 for n' and add a pertubation which let the indicator function to be equal to 0).
Afterwards we take the max over all possible amounts of contamination such that the genes pairs don't change (they are given by the arg max function) in order to have the largest number of contamined data that can be introduced to the dataset such that the gene pair stays the same as for the original dataset.
Finally, we divide this number by the total number of observations n to obtain a proportion.

2.3.5 Analytical study of the robustness of the TSP

In this part we are intersted in computing analytically the robustness of the TSP. Firstly we will study the influence of a new observation on the pair of genes selected by the TSP on the original dataset and show that this influence depends highly on the setup of the dataset and express it explicitly as a function of the setup. Secondly we will give the value of the breakdown point of the TSP using a sample where this value can be computed easily and we will show that it is equal to 0.

Influence of a new observation We suppose we performed the TSP on a dataset and that the gene pair resulting is given by (i, j). Without loss of generality we suppose that $p_{ij}(C_1) > p_{ij}(C_2)$, so for a new observation x_{n+1}, we assign it to the group C_1 if $x_{i,n+1} < x_{j,n+1}$ and to the group C_2 otherwise.

We set the following notation, $n_1 = |C_1|$, $n_2 = |C_2|$, $k_1 = |\{k \in C_1 : x_{i,k} > x_{j,k}\}|$ and $k_2 = |\{k \in C_2 : x_{i,k} > x_{j,k}\}|$.

The score of the pair (i, j) is defined by $\Delta_{ij} = \left|\frac{k_1}{n_1} - \frac{k_2}{n_2}\right|$.

Suppose the second best TSP is represented by (k, l) (here we suppose that there are no ties, i.e., $\Delta_{kl} < \Delta_{ij}$). We want to see the influence on a new observation on the gene pair yield by the TSP. More precisely we are interested to see when the pair (i, j) will become ranked second instead of first. Suppose we add a new observation x_0 to the dataset and without loss of generality we can suppose that x_0 comes from the group C_1. We distinguish two cases

1. $x_{i,0} > x_{j,0}$:
   The new observation matches with the decision rules of the current TSP. Then the TSP will remain unchanged and its score will increase and be equal to
   $$\Delta'_{ij} = \left|\frac{k_1 + 1}{n_1 + 1} - \frac{k_2}{n_2}\right| > \Delta_{ij}, \Delta'_{kl}.$$
   The TSP will select again the pair (i, j).

2. $x_{i,0} < x_{j,0}$:
   This case is more interesting since the classification rule of the TSP doesn't match with the relative ordering expression of the new observation. Thus the computation of the score will be affected. Indeed the score for the pair (i, j) will become lower as we add an individual to the group C_1 whose relative ordering gene expression doesn't match with the classification rule. The score

of the pair (i,j) would be given by

$$\Delta'_{ij} = \left|\frac{k_1}{n_1+1} - \frac{k_2}{n_2}\right| < \Delta_{ij}.$$

Now the point is to see if this new individual will make the score of the second pair bigger. We suppose it is the case (otherwise we would use the pair that has the higher score and whose classification rule matches with the relative ordering of the new observation for this pair), i.e., $x_{k,0} > x_{l,0}$. Since the new observation matches with the classification rule of the TSP based on the pair (k,l) the score Δ_{kl} will increase and be

$$\Delta'_{kl} = \left|\frac{k'_1+1}{n_1+1} - \frac{k'_2}{n_2}\right|,$$

where k'_1 and k'_2 are defined as for the first pair but with indices (k,l) instead of (i,j). The results of the TSP on the original dataset will change if the score of the second pair (the one for wich the classification rule matches with the relative ordering of the new individual) becomes higher than the score of Δ_{ij}, i.e., if

$$\left|\frac{k'_1+1}{n_1+1} - \frac{k'_2}{n_2}\right| > \left|\frac{k_1}{n_1+1} - \frac{k_2}{n_2}\right|,$$

which is equivalent to

$$\left|\frac{k'_1+1}{n_1+1} - \frac{k'_2}{n_2}\right| - \left|\frac{k_1}{n_1+1} - \frac{k_2}{n_2}\right| > 0.$$

This is a necessary and sufficient condition such that the pair (i,j) is not selected anymore by the TSP.

Let us define the fonction

$$f(k_1,k_2,k'_1,k'_2,n_1,n_2) = \left|\frac{k'_1+1}{n_1+1} - \frac{k'_2}{n_2}\right| - \left|\frac{k_1}{n_1+1} - \frac{k_2}{n_2}\right|. \tag{8}$$

This function depends highly on the setup of the dataset as it depends explicitely on the variables $(k_1,k_2,k'_1,k'_2,n_1,n_2)$, which are quantities specific to the analysed dataset.

Here we supposed that the new observation belonged to the group C_1. If the observation came from the group C_2 the derivation of the function f would have worked in the same way, but n_2 would have been increased to $n_2 + 1$ instead of n_1. Supposing this observation would be unfavorable to the classification rule of the pair (i, j) (the pair (i, j) would misslcassifie this observation) but not for the pair (k, l), then k'_2 would be increased to $k'_2 + 1$. We note that adding an observation in the group C_1 or in the group C_2 has not the same influence on the score when $|C_1| \neq |C_2|$.

We also have to be careful on the fact that for the derivation of f we supposed that $p_{ij}(C_1) > p_{ij}(C_2)$. If this inequality is inversed, the interpretation of favorable observation to a pair of genes will have a different meaning, in the sense that the relative order of the gene expressions will be inversed.

We saw that if the new observation's relative ordering gene expressions matches with the classification rule of the best pair, the TSP selects the same pair. But in the case where the relative ordering gene expressions of this new individual doesn't match with the classification rule, then changes in the selected pair of genes may occur. The pair that could become ranked first is one for which the relative ordering gene expression of this new observation matches with the classification rule of this pair, otherwise the score of this pair won't be increased and can't become higher than the best score. In order to have this pair ranked first, its score must become higher than the updated best score (taking into account that a new observation is added to the dataset). The function $f(\cdot)$ gives information if the selected gene pair will be subject to changes with the simple condition $f(\cdot) > 0$ (this means a different gene pair will be selected).

We can make an extension to this function to find how many observations we can add to the current dataset such that the selected gene pair will remain unchanged. We consider the worst case, i.e., add observations whose relative ordering genes expressions don't match with the classification rule but match with the classification rule of the second best pair (which will become the top ranked one in this case). We have to split the derivation of this function in two cases as for the derivation of the function f. First we suppose that the new observations come from the group C_1 and define F_1 is defined as

$$F_1(k_1, k_2, k'_1, k'_2, n_1, n_2, x) = \left| \frac{k'_1 + x}{n_1 + x} - \frac{k'_2}{n_2} \right| - \left| \frac{k_1}{n_1 + x} - \frac{k_2}{n_2} \right|.$$

Then we seek for the largest positive integer x such that the function $F_1(k_1, k_2, k'_1, k'_2, n_1, n_2, x)$ stays negative.

In the same way, for new observations which come from the group C_2 we define the function F_2 with the same parameters as for F_1 as

$$F_2(k_1, k_2, k_1', k_2', n_1, n_2, x) = \left| \frac{k_1'}{n_1} - \frac{k_2' + x}{n_2 + x} \right| - \left| \frac{k_1}{n_1} - \frac{k_2}{n_2 + x} \right|.$$

Here we supposed that $p_{ij}(C_1) > p_{ij}(C_2)$. The case where $p_{ij}(C_1) < p_{ij}(C_2)$ will be similar but we need to change the indices, i.e., k_1, k_2, n_1 and n_2 will become k_2, k_1, n_2 and n_1 respectively.

These functions seek how many observations from group C_1 or C_2 wich are unfavorable to the best pair but favorable to the second one, one can add such that the orignal pair remains ranked first, i.e., the largest integer x such that $F_1(k_1, k_2, k_1', k_2', n_1, n_2, y) < 0, \forall\ y \leq x$ and equivalently for F_2.

One can combine these two functions in order to obtain the largest amount of data one can add such that the pair of genes remains unchanged in the worst case, i.e., if the new observations came from the group C_1 or C_2. We note that if the sizes of the two groups are equal, then it has no influence if the observations come from C_1 or C_2. If the groups' sizes are not equal then the worst case will be when unfavorable indviduals come from the group with the lowest size as its influence will be of size $\frac{1}{C_i}$, $i = 1, 2$. We can combine the functions F_1 and F_2 to have a general form for the largest number of observations one can add such that the gene pair stays the same. We define the function F, which stands for this quantity, as

$$F(k_1, k_2, k_1', k_2', n_1, n_2, x) = \max\{x :\ F_1(y),\ F_2(y) < 0,\ \forall\ y \leq x\}. \tag{9}$$

We saw a function that is able to compute if a new observation may produce changes in the gene pair selected and we also saw an extension to compute how many observations, which are unfavorable to the top pair but favorable to the second pair (seen as the worst case), can be added to the dataset without producing changes to the selected gene pair. We will apply these two functions on the dataset of leukemia cancer and breast cancer and compare their values with the result we found when we used the boostrap resampling. We will first apply them on the breast cancer dataset as it will give a direct calculation to compute the breakdown point fo the TSP and will highlight the effect of ties on the choice of the best pair.

When applying the TSP on this dataset a tie will occur. Indeed the two pair of genes $(498, 2333)$ and $(541, 2343)$ will be selected with the same score $\Delta = 0.82$, the second score Γ enables to break this tie and choose the pair $(541, 2343)$. In this case, if we add only one observation which is favorable to one of the pair but unfavorable for the second pair, it is obvious that the first pair will be selected and be the only one to achieve the best score. Here we don't consider the second pair $(498, 2333)$ and see when the score of the third best pair will become ranked first. The third best pair of this dataset is $(3621, 5488)$ with a score $\Delta = 0.79$. For these two pairs of genes we have the setting $k_1 = 40$, $k_2 = 3$, $k'_1 = 40$, $k'_2 = 4$. In this case the group C_2 contains less observations than the group C_1, so an unfavorable observation for the top TSP will have a higher influence if this observation belongs to the group C_2. Here we are interested in the worst case, so we suppose that the observation comes from the group C_2, the size of the first group n_1 as well as the number of people in the group satisfying the rule of the first TSP k_1 and of second TSP k'_1 won't change, in addition the number of people in the second group satisfying the rule of the first tsp k_2 won't change too. However the number of people in the second group that satisfies the rule of the second TSP will be incremented by 1 (as the size of the second group). After some computation we obtain that $F = 0$. Which means that the second considered TSP will become as good as the first one and a tie will occur. It may happen that this TSP will have an average ranking difference higher that for the first TSP and finally be picked as the best pair by the TSP. We note that this analyse could have been done in the same way when we remove one observation from the dataset. This example illustrates well how adding a single observation might change the pair of genes selected by the TSP. This highlights the sensibility of the TSP and gives an accurate function of its sensibility depending on the setting of the analyzed dataset.

Through the computation of the function F on the dataset of breast cancer we observed that adding an observation that doesn't match with the top TSP but which matches with the second best TSP might produce changes on the gene pair that the TSP selects. This means that, in the worst case, the largest proportion of data we can perturbate such that the TSP produces the same result is 0%, for this reason the breakdown point of the TSP in, a general case, is 0. It is clear that it highly depends on the dataset we used, and for this reason we gave an adaptation of the breakdown point to a specific dataset. Moreover the maximum score Δ was reached by two pairs, it was intuitively obvious that adding an observation that matches only with one of the classification rule would have produced changes on the score Δ and thus modify the selected pair of genes.

We will apply these functions on the dataset leukemia cancer. The best score is $\Delta = 0.98$ and is reached by the pair $(463, 720)$, the second highest score is $\Delta = 0.96$ for the pair $(1900, 2234)$. The values of the parameters for the function of interest are $n_1 = 47$, $n_2 = 25$, $k_1 = 46$, $k_2 = 0$, $k'_1 = 0$ and $k'_2 = 24$. By the same way as before we find that adding only one observation in any of the groups will modify the selected pair of genes. This can be also quickly derivated from

$$\max\left(\frac{1}{n_1}, \frac{1}{n_2}\right) = 0.04 > 0.02 = 0.98 - 0.96 = \Delta_{463,720} - \Delta_{1900,2334}.$$

This shows that introducting only a single observation (x_0, y_0) that doesn't satisfy the classification rule of the top pair can change the pair selected by the TSP. This remark allows us to compute the adaption of the sensiitivity curve for the TSP. Acutally it will depend on the values of x_0 for the genes selected. If (x_0, y_0) will be correctly classified, then the gene pair selected by the TSP won't change and we will have $\mathrm{SC}_{\mathrm{TSP}} = 1$. By contrast, if (x_0, y_0) will be missclassified, then the resulting gene pair will change and then $\mathrm{SC}_{\mathrm{TSP}} = 0$.

The mathematical analysis showed that the estimators of the TSP Δ and Γ were slightly sensitive to pertubations on the dataset and that the influence of a new observation was bounded by $\frac{1}{n_1}$ or $\frac{1}{n_2}$, depending to which group the observation belonged to. A solution to deacrease the influence of a new observation is to increase the number of observations in each group, this is a well known problem in statistics. Indeed with a too low number of observation the parameters are estimated with low accuracy and small changes in the data can strongly affect the estimations. In the case of the TSP we know that the highest influence that an observation can have on the score Δ is bounded by $\max\left(\frac{1}{n_1}, \frac{1}{n_2}\right)$, and that on the score Γ the influence is bounded by $\frac{P}{N}$.

Finally we observed, by applying the method on bootstraped resamples of the original dataset, that the TSP may be highly affected by pertubations on the dataset. This sensibility could be measured mathematically and expressed through the largest amount of "bad" perturbations the TSP could bear without producing changes in the yielded pair of genes. When we computed the influence of one observation we supposed that there we no ties in the score of the pairs of genes. If ties were present the sensitivity would have been even bigger. Indeed, the presence of ties mean that several pairs achieved the maximum score Δ and the choice of a the top pair was made through comparing the score Γ. If we add an observation that satisfies the classification rule only for one of the pair whose score achieved the maximum score, the TSP will select this pair. Thus, in case of ties, it is enough to add only one well chosen observation to produce changes on the results of the TSP.

The bootstrap could indirectly define a robust way to select a pair of genes. Indeed one can choose the pair that appears the most often over the 500 bootstraps and use it at the top pair. The bootstrap had shown that this pair is the one that appears the most often as the dataset is slightly modified and would claim that this pair is the most robust one over the pairs present in the dataset.

One important point of this section is that all the results presented here were computed on the leukemia and breast cancer datasets. If we would have used another dataset, the results would have been different. This will be illustrated in the last section when we will present another dataset to compare the methods. Indeed the results of precision, explained through the function f, F and the ROC curve will have a very different form. This is often the case in statistics, the results of the methods used depend strongly on the dataset we analysed. This make the analyses difficult to perform because there are not a best way to proceed.

In the next section we will present a method that is based on the same idea as the TSP, but is extended in the sense that it can consider several pairs of genes. This method is supposed to yield a better accuracy and be less sensitive to perturbations.

3 The first extension of the TSP: the k-TSP

In this section we present an extensions of the TSP, namely the k-TSP. As we saw in the last section, the TSP may change when the training data is perturbed by adding or deleting few observations. The k-TSP classifier extends the TSP and is constructed to deal with this problem as well as providing more accuracy than the TSP and to yield a more stable classifier. The k-TSP was introduced by Tan et al. [2]. It builds a classifier using k disjoint TSPs that provide the best scores Δ. We can see the k-TSP as an ensemble method, where the goal is to use the power of many "weaker" rules to construct a "stronger" classifier. This technique of combining several predictors is also called boosting. The value of k is chosen such that the accuracy is maximal. For computational problem we have to set k to be less than 10. Moreover the classification will be based on a vote, i.e., counting the number of predictions for each classes for each classifiers and choose the group that obtained the most votes. In order to select only one class we have to require k to be an odd number. There are different ways to choose k, in this paper we used cross-validations on the dataset and selected the k that produced the highest accuracy. If $k = 1$, then it is obvious that the k-TSP is equivalent to the TSP as it selects only one pair of genes, namely the one with the highest score Δ.

The flexibility provided by the k-TSP might be of great interest for follow up studies as it considers more than two genes that are of particular interest and might give a better indication to which genes should be considered.

3.1 k-TSP

In this section we will explain how the k-TSP works as a direct extension of the TSP. We will show that it is also based on the two same scores Δ and Γ, and then we explain how it classifies a new observation. Afterwards we will study its robustness, which is quite straightforward from the TSP, we will also study its sensibility to changes in the dataset in the same way as for the TSP. We will use bootstrap to obtain different samples, perform the k-TSP on it and analyse the changes in the results. We will repeat this procedure for several datasets and compare the results with those obtained for the TSP.

3.1.1 Definition of k-TSP

The k-TSP is a direct and easy extension of the TSP. The main feature that differs for these two methods is the number of TSP included in the final prediction. It is intuitive to select the k best pairs to construct the k-TSP as we want the k-TSP to reach a high accuracy. To avoid high correlation between the selected pairs, we require the genes in the dataset to appear at most in one pair in the k-TSP. This imposes the k-TSP to be combined with more different genes and thus, in case where measurement errors occur in the dataset, avoid the k-TSP to be quickly corrupted.

The first step is to construct an ordered list of all possible gene pairs where the first one stands for the best one, the second one for the second best and so on. We must first compute Δ_{ij} and Γ_{ij} as defined in (2) and (3) for all possible pairs of genes (i, j). We sort them in descending order with respect to the value of the first criteria Δ, and if ties occur we use the criteria Γ to break them. We take the top pair (i, j) as being the first TSP. We remove all pairs that involve either the gene i or the gene j from the list created in the previous step and select the second pair as beeing the second TSP. We remove all the pairs that involve at least one of the gene used in the second TSP. We repeat this procedure until we obtain a list of k TSP. This list is defined as the k-TSP.

The number of pairs k will be determined by cross validation with the restriction that is has to be an odd number in order to break ties in the majority procedure and to be less than 10 for computational easiness.

Moreover we restricted k to be less than 10 for computational speed. The number of TSPs k is determined by cross validation. More precisely, the data set will be separated into a training set and a test set wihout overlapping. We perform a k-TSP on the training set for every possible value of k and compute their accuracy on the test set by computing their prediction error. We proceed in this way for several separation of training and test set (as in usual cross validation procedure) and finally average the error prediction for each of the possible k. We choose the value that minimizes the error prediction and perform a final analyse on the whole data set with the selected value for k.

In our study we used 10 fold cross validation, this means we seperated the dataset into 10 subsets without overlapping, preformed a k-TSP on the set combined of 9 subsets and computed the prediction error on the remaining subset. We repeated this on all possible choice of 9 subsets and finally averaging the prediction error.

Table 1 present the algorithm to perform a k-TSP on a data set for a given k. This algorithm is easily adapted to the cross validation procedure to determine k.

```
Algorithm k-Top scoring pair

  Input:  The expression gene values for each observation,
          the group to which it belongs, denoted by $S$,
          and the number of TSPs $k$.

  Output: $k$ pairs of genes that achieved the best score $\Delta$
          (genes can appear only in one pair)

    1  Define a list called $k$-TSP that will contain the chosen TSP methods
    2  Compute the values of $\Delta_{ij}$ and $\Gamma_{ij}$ for all pairs of genes $(i, j)$,
        where $1 \leq i \neq j \leq P$.
    3  Make a sorted list $\mathcal{M}$ of all pairs of genes such that $(i, j) \prec (i', j')$
        defined as either $\Delta_{ij} > \Delta_{i'j'}$ or $\Delta_{ij} = \Delta_{i'j'}$ and $\Gamma_{ij} > \Gamma_{i'j'}$.
    4  Repeat $k$ times:
        Pick the top pair $(i, j)$ and put it into the list k-TSP.
        Remove every pairs in the list $\mathcal{M}$ which involves either $i$ or $j$.
    5  Return $k$-TSP.
```

Table 1: **Algorithm 1** k-Top scoring pair

We used the package of the TSP in R [6] as a base to create new functions that allow to compute the k-TSP on this software. We extended the code C in a way that makes it able to yield k pairs of genes, where the k is given as a parameter of the function. We extended all the functions available in the package, as the function *plot, summary, prediction,etc* to deal with the new class of created objects, namely the class *k-TSP*. We also made an extension on the overall method to be able to deal with the presence of *NA* in the datasets. The initial package doesn't take it into account, indeed where the comparisons $R(i,n) < R(j,n)$ are not possible because of the presence of *NA* the method goes over it but uses in the computation of the score the division by n_1 or n_2 depending on which group the observations belongs too, instead of dividing by the number of time the comparison was possible. This produces a bias in the computation of Δ, and can give lower score to pairs of genes that contain *NA* which would have been better if the *NA* weren't considered. In our functions we corrected this by dividing through the number of comparisons that were possible (i.e. when no *NA* are present) within each group. We also modified the functions *plot, summary, prediction, etc* to deal with the presence of *NA* and to produce coherent results. The code is available in the Appendice and maybe a package will be available soon on R. To validate our method, we computed it for the dataset of leukemia and prostate cancer and compared them with the results found in [1] and [2]. The results are summarized in Table 2. Our results are quite similar to the one obtained in other papers. This allow us to validate our method. We note that the values of k are computed through crossvalidation and thus they possess a random part (see next paragraph for details about the crossvalidation). This can lead to differences, as for the prostate cancer dataset. By contrast, the correct rate prediction is less sensitive to the partition made from the dataset in the crossvalidation procedure.

	Tan et al.		Yoon & Kim		Our implementation	
	k	CRP	k	CRP	k	CRP
Leukemia	9	0.95	5	0.94	9	0.97
Prostate	1	0.95	5	0.90	1	0.93

Table 2: Comparisons of the results of the k-TSP computed on the datasets leukemian and prostate cancer with the results found in the articles [2] and [1]. The abreviation CRP stands for "correct rate prediction".

To determine the value of k that we will use to compute the k-TSP, we used 10-fold crossvalidation. The 10-fold crossvalidation works as follows, first it seperates the dataset into 10 non overlapping subsets of approximately equal size ($\approx \lfloor \frac{n}{10} \rfloor$, where n is the size of the dataset and $\lfloor m \rfloor$ stands for the largest integer less than m). We choose one of these 10 subsets as the test set and we merge the 9 remaining subsets to compute a k-TSP with $k = 1, 3, \ldots, 9$. For each of these k values we apply the k-TSP on the test set (the chosen subset) and compute the error prediction rate for this value of k. This gives an error prediction rate for each of these k. We choose another subset to be the test set and proceed in the same way as previously. We repeat this operation until we have chosen all the subsets as test set. We have finally 10 error prediction rates for each of the k values. For each k we take the mean over all error prediction rates calculated for this k and set it as the total error prediction rate. We choose the k that provided the lowest error prediction rate and apply the k-TSP on the full dataset with the chosen k.

This method is widely used to estimate parameters. It has the advantage that the user doesn't require a supplementary test set to test the model fitted on the training set. It allows the user to have also an estimation of the error prediction rate for the fitted model. One must be careful to the fact that this estimation is overestimated as the final model will be based on a bigger dataset and thus provide an higher accuracy.

It is arbitrary to use 10-fold or 5-fold crossvalidation. There exists also the special case of the $n-1$-fold crossvalidation, known under the name of the leave-one-out crossvalidation (LOOCV). The more subsets we use to perform the crossvalidation the less biais will have the estimation of the prediction. But the variance will increase with the number of subsets we divide the dataset into. Indeed the test set will become small and the estimation will be based only on few estimations.

In this paper we chose to work with the 10-crossvalidation, because it provides a good trade-off between the unbiasedness and the variance of the error prediction rate.

3.1.2 Classification of a new observation

The k-TSP was fairly quickly defined from the TSP, the second step is to define how to classifie a new observation. This is achieved through the classification rule of each TSP contained in the k-TSP. Then we use the majority procedure to select the group to which a new observation will be assigned. For a new observation, we compute the prediction for each TSP in the k-TSP and count how many times they predicted the class C_1 and C_2. We choose the class that obtained the more votes. In order to express this mathematically we define the following function for a new observation x_{n+1}

$$h_l(\mathbf{x}_{n+1}) = \begin{cases} C_1, & \text{if } x_{i,n+1} < x_{j,n+1} \\ C_2, & \text{otherwise} \end{cases}, \tag{10}$$

where l stand for the index of the pair (i, j) in the list of the the k-TSP, $l = 1, 2, \ldots, k$. We supposed that $p_{ij}(C_1) > p_{ij}(C_2)$, we have the opposite situation if $p_{ij}(C_1) < p_{ij}(C_2)$.

We define the vote of a single pair to a class as

$$I(h_u(x_{n+1}) = C_i) = \begin{cases} 1, & \text{if } h_u(x_{n+1}) = C_i \\ 0, & \text{otherwise} \end{cases}. \tag{11}$$

The final prediction is given as a sum of all single predictions, and we write it as

$$y_{n+1} = h_{k\text{-TSP}}(x_{n+1}) = \underset{i\in\{1,2\}}{\arg\max} \sum_{u=1}^{k} I(h_u(x_{n+1}) = C_i). \tag{12}$$

We observe that if k is an even value, the prediction can be indefinite and predict the new observation to be in both classes. For this reason we restrained k to be an odd number, to get always a clear prediction.

Application of the k-TSP on the leukemia cancer dataset Figure (3) presents the results of the k-TSP applied on the leukemia cancer dataset. We computed k with 10-fold crossvalidation, it results in choosing $k = 9$ and the error prediction rate was around $e = 0.028$.

The results of the k-TSP on the original dataset have little similarities with the results of the TSP on the bootstrap. Indeed only the first pair of genes of the k-TSP appears on the bootstrap of the TSP. The graph of the frequency of appearance of the genes pairs in the bootstrap of the TSP shows that the gene 720 appears in multiple pair.

k-TSP	Indices	Scores
TSP 1	463/720	0.98
TSP 2	923/1900	0.96
TSP 3	2318/2614	0.96
TSP 4	735/2019	0.94
TSP 5	1250/2185	0.94
TSP 6	942/2130	0.94
TSP 7	331/838	0.94
TSP 8	902/1972	0.92
TSP 9	617/2460	0.92

Table 3: Result of the k-TSP computed on the leukemia cancer dataset. The 10-fold crossvalidation provided $k = 9$. The TSP are ordered with respect to their score, in case of tie break the rank average score is used.

The k-TSP allows a gene to appear only in at most one pair of genes in the k-TSP. For this reason, once the gene 720 is used in a pair, it can't be used again to create the other pairs present in the bootstrap of the TSP. The top pair which involved the gene 720 was $(463, 720)$ for both the TSP and the k-TSP.

The goal of allowing a gene to appear only in at most one pair was to reduce the correlation between the genes pairs and to prevent against outliers. Indeed if the values of a gene for an individual are aberrant (or unvailable), this procedure avoids having to use this gene in several pairs. On the other hand, this restriction has the disadvantage to eliminate genes that were important only when compared with a gene present in a previous pair. For example, among all the genes that were highlighted on the bootstrap that involve genes 720, none appears again in the k-TSP.

3.2 Robustess of k-TSP

We saw that the k-TSP was defined through an ordered list where the componants, which are pairs of genes, were ordered through their scores Δ and Γ. As for the TSP it is obvious that one factor that may influe on the robustness of the k-TSP is the robustness of the parameters Δ and Γ. This point was already analysed in a previous section and acts here similarly.

As the list is defined through multiple calculations of TSPs, it is evident that the robustness of the k-TSP will be influenced by the robustness of the TSP. We saw that the TSP was sensible to perturbations of the dataset. As the k-TSP is an ensemble of TSPs, a first intuition is to say that it will also be sensitive to perturbations. But if we analyse the k TSPs we choose to build the k-TSP, we might assume they are the best k one, and that the $k+1$-th best TSP (which is not selected) will have a much lower score that the k-th best one, and thus, even if a large amount of perturbations occurs, the k-TSP won't select the $k+1$-th best pair and still contains the same k pairs. This highlights a first property for robustness of the TSP: the score of the "worst" TSP in the list of the chosen TSP has to be "far away" from the best score over the non selected gene pair. We will come back to this point in a later subsection.

In this section we will analyse the robustness of the k-TSP. We will proceed in the same way as for the TSP. First we apply the k-TSP we implemented on the leukemia cancer dataset. Then we perform 500 bootstrap resamples of this dataset and for each of these resamples we compute the k-TSP on it. We will study how the number of TSPs chosen k as well as the pairs of genes chosen change over the boostraped dataset. In the second part we will analyse mathematically the robustness of the k-TSP. We will briefly discuss the notion of sensitivity curve and of breakdown point for the k-TSP as it will be a direct deduction from the results on the TSP. Then we will investigate the influence of adding a new observation on the results yield by the k-TSP. This will have some similarities with what was made before for the TSP, and the results will be analogue.

3.2.1 Study of robustness through simulation

We performed 500 bootstraps resamples of the leukemia cancer dataset and on each of these resamples we computed a k-TSP, where k was computed through 10-fold crossvalidation. For the analysis of the results we proceed in the same way as for the TSP. First we look for the frequency of appearance of the genes among the 500 computed k-TSPs. Second we compute the frequency of appearance for the pairs of genes. Afterwards we present an histogram of the values of k that were selected over the bootstraps. Finally we analyse which k-TSPs appeared the most often among the bootstraps.

Figure 6 shows the frequency of appearance of the genes in the chosen pairs of genes among the 500 computed k-TSPs. We see that much more genes appear, this is clearly explained by the fact that the k-TPS uses $2k$ pairs and the TSP only 2. There are 2 genes that appear much more often than the others, the genes 720 and 1900. The gene 720 appeared often in the results of the TSP too, but the gene 1900 appeared much more rarely. If we look in Table 3, we can see that the gene 720 is present in the top pair and the gene 1900 in the second pair. When applying the TSP only the first pair will be selected, but, with the k-TSP, more pairs can appear and thus gives more chance to the second, third, etc. pair to appear. Based on the TSP, the gene 1900 would have been considered only as little important as it appeared only 5% of the time. Whereas, based on the k-TSP, this pair would be considered as very important, as it appears the most often with the gene 720. We can distinguish a second cluster composed by the genes 463, 735, 923 and 1250. The first three genes of this cluster appeared also quite often in the TSP, but the fourth one, the gene 1250, never appeared. The k-TSP would make the gene 1250 to be considered as important to detect the disease. However, it wouldn't have ever been considered when using only the TSP. The rest of the genes that had an important frequency of appearance in the k-TSP were not considered by the TSP. Finally, we note that the gene 923 that appeared in the TSP, is not of importance in the k-TSP, even if it appears in the second best pair. The reason is that this gene is only useful when it is compared with the gene 1900, when the gene 1900 is not available (may be already used in another pair) the usefulness of this gene drops.

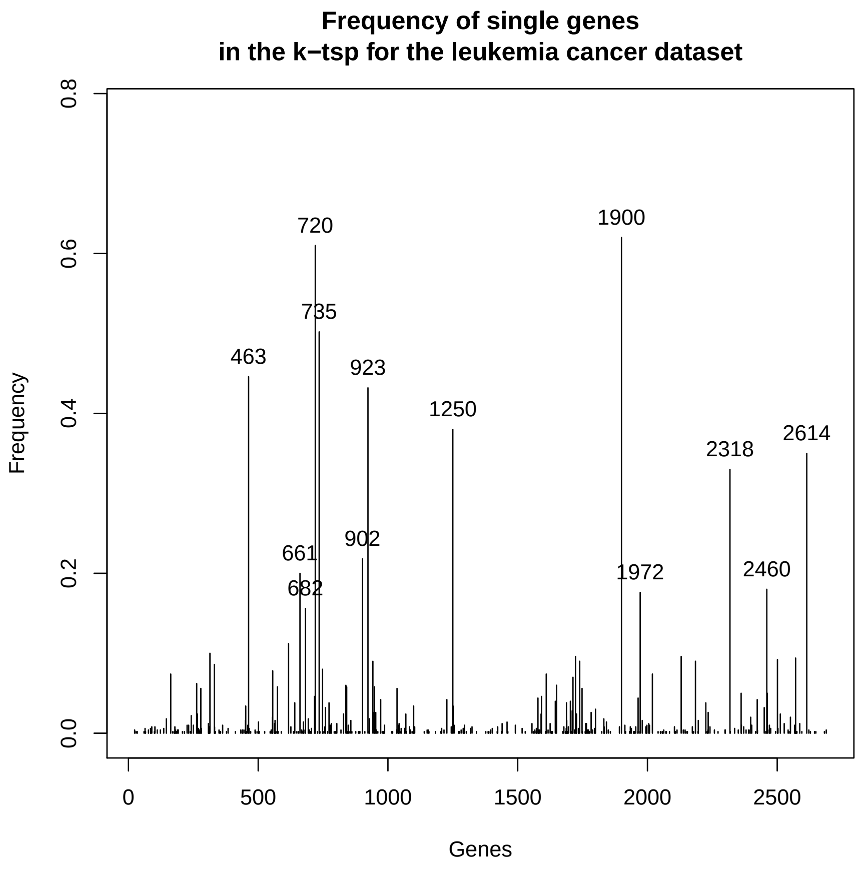

Figure 6: Frequency of appearance of the single gene in the pairs selected by the k-TSP over 500 bootstrap resamples from the dataset **Leukemia cancer**, only the genes whose appearance were above 8% were hold.

We can conclude that the k-TSP provides generaly almost all genes that were present in the TSP, but some exceptions can occur. It makes also genes that weren't important for the TSP much more important. Actually, the idea of the k-TSP is to deal with more than only one pair of genes, and to make other genes also of importance to detect the disease. As we can see in this case, the k-TSP will allow the user to obtain more gene related to the disease that wouldn't have been considered by the TSP, even by applying a bootstrap procedure. This highlight one of the advantage of the k-TSP, to be more flexible and thus can provide more and better genes that would be linked to the disease.

We analyse now the appearance of the pairs of genes over the bootstraps. Figure 7 presents the frequency of appearance for the pairs of genes present in the k-TSPs over the 500 bootstraps resamples. The pair $(463, 720)$ appears as being the top one and appears in about one third of the k-TSPs. The second pair $(923, 1900)$ appears around 17% of the time. Then a cluster of pairs appears, whose appearence are between $7 - 13\%$. Although if for the TSP a lot of pairs including the gene 720 appeared, this is not necessarily the case for the k-TSP as the graph shows.

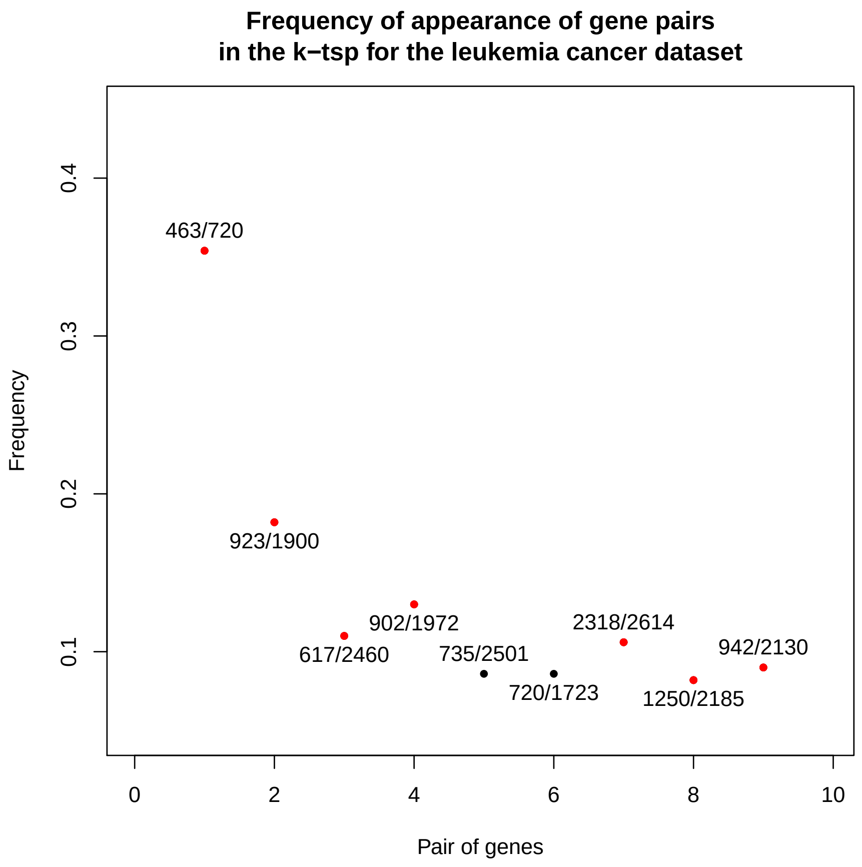

Figure 7: Frequency of appearance of the pairs of genes selected by the TSP over 500 bootstrap resamples from the dataset **Leukemia cancer**, we ploted only pairs whose frequency were at least 5%. The pair with the red dot represents the pair selected by the k-TSP on the original data set.

An important point is to look on the values of k among the bootstrap resamples. We wish that k stays stable, this would be a good point to present the k-TSP as a robust method. Figure 8 shows a histogram of the values of k that were computed through 10-fold crossvalidation over the 500 bootstraped datasets. Unfortunately k varies a lot, and ranges over all possible values, i.e., all odd numbers between 1 and 9. The two most frequent values for k are 3 and 5, however the frequency of having $k = 3$ is higher. So the most probable value for k is 3. Now, the point is which k TSPs one would choose to construct the k-TSP.

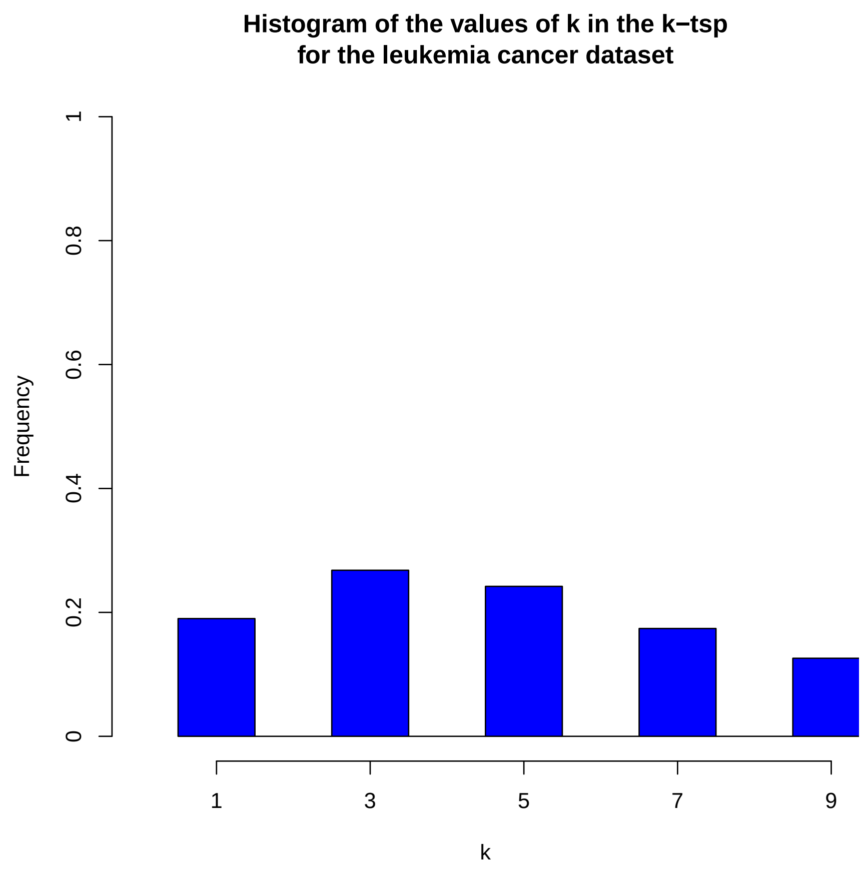

Figure 8: Histogram of the values of k computed by 10-fold crossvalidation over the 500 bootstraps on the leukemia cancer dataset.

Over the repeated computations of the k-TSPs on the bootstraped resamples, the k-TSPs that appeared the most often was formed by only one pair of genes. This resulted to be finally a TSP. This problem is caused by the sensitivity of the k TSPs present in the k-TSP. As we saw in a previous section, small perturbations can strongly affect the pair of genes yield by the TSP. The k-TSP being formed by several TSPs also possesses this property. Again this result of sensibility depends highly on the dataset we analyse. For instance, if we would consider a dataset where only 3 pairs would be "good" for classification and all the other genes would have a score of 0, then it is obvious that the k-TSP will select either one pair or the three pairs. And, in this case, the result could be much more stable, altough it could easily jumps from 1 to 3 pairs when small perturbations occurs. We will study this sensitivity in the next section.

This problem of sensibility causes too much variations in the k-TSPs. And thus the k-TSPs with k big will have less chance to reappear in the bootstrap (we didn't considered the order in which the TSPs were ranked). This causes that the k-TSP that appeared the most often were finally simple TSPs. Table 4 presents the k-TSPs that appeared the most often over the bootstraps. These pairs were selected by the k-TSP computed on the original dataset, but one of them were not ranked as one of the top pairs. Indeed the pairs $(463, 720)$, $(923, 1900)$ were ranked first and second respectively, but the pairs $(902, 1972)$ and $(617, 2460)$ were ranked eighth and nineth. If we look on the score of the pair chosen by the original k-TSP, we see that the third one (the one that was expected to appear the most after the first two) had a score of 0.96 and the eighth one a score of 0.92. Changing one observation in the second group that was favorable to the third pair but unfavorable to the eighth pair into an observation that would be favorable to the eighth pair but unfavorable to the third pair would increase the score of the eighth pair by 0.27 and lower the score of the third pair also by 0.27 ($= \frac{1}{n_2}$). This results in an higher score for the eighth pair than for the third one. Thus, changing one observation can have a big influence on the pair we choose, even for the k-TSP.

k-TSP	Indices
k-TSP 1	463/720
k-TSP 2	923/1900
k-TSP 3	902/1972
k-TSP 4	617/2460

Table 4: The k-TSPs that appeared the most often over the bootstraps.

This example shows that the k-TSP can change a lot when perturbations occur. Indeed, we computed on the original dataset a k-TSP with $k = 9$ and saw that, over the bootstraps, this value changed a lot. Nevertheless, the pairs of genes provided by the original k-TSP were good, in the sense they were highly correlated with the disease. The bootstrap makes this assumption even stronger as almost all the pairs of genes of the orignal k-TSP are present in the computation over the bootstraps. Indirectly, we presented two methods to perform the k-TSP. The first one, the one we presented, was to use crossvalidation on the original dataset to determine k and then to apply the method with the selected k on the orginal dataset. The second was presented as we performed the bootstrap resample. Indeed one can use bootstrap resample and perform a k-TSP on each of these (k is chosen through 10-fold crossvalidation on the bootstraped dataset). Then one chooses the value of

k that appeared the most ofen over the boostrap and chooses the k pairs of genes that appeared the most often to form the k-TSP. For example, on the leukemia dataset, we would choose $k = 3$ with the pairs of genes $(463, 720)$, $(923, 1900)$ and $(902, 1972)$.

Figure 9 shows an estimation of the ROC curve for the k-TSP on the Leukemia cancer dataset. We proceeded in the following way. First we simulated 200 bootstrap resamples of the original dataset. On each of these resample we performed a k-TSP, where k was chosen by 10-fold crossvalidation on the bootstrap sample. At each step we used the individuals out of the bootstrap resample to compute a ROC curve. In this way we obtained 200 curves representing 200 ROC curve. We averaged them and used boxplot to present the variability of the sensiblity at the chosen point of the specificity.

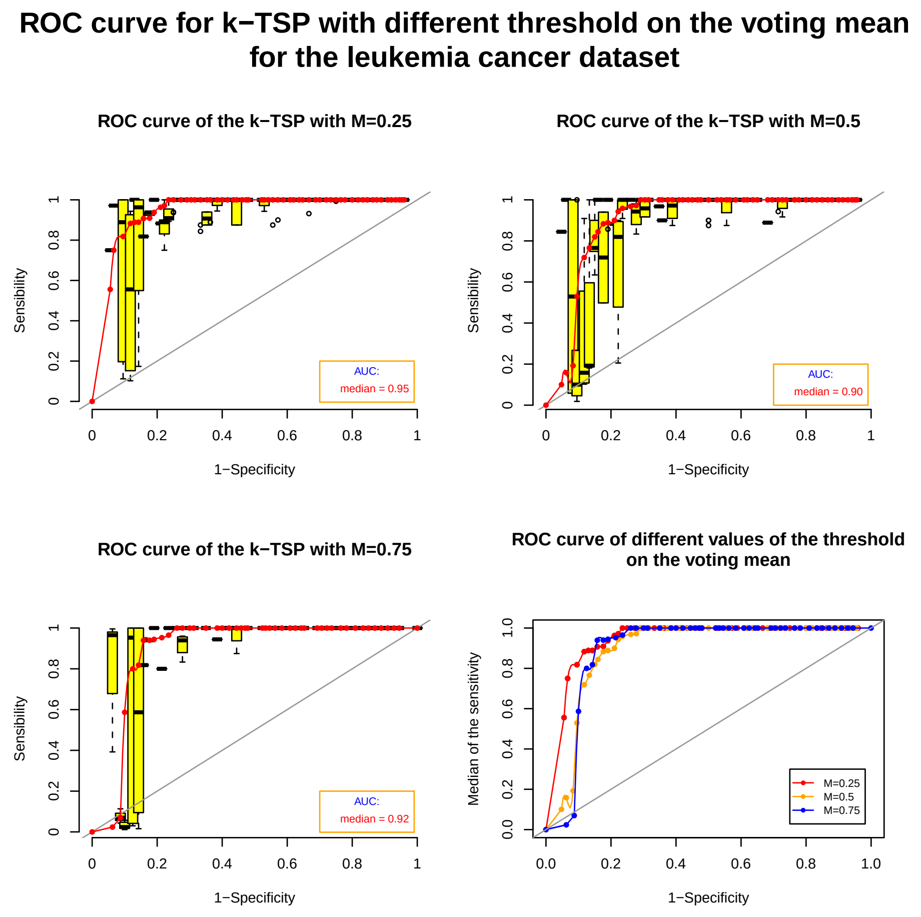

Figure 9: ROC curve of the k-TSP on the leukemia cancer dataset computed on 200 bootstraps and with different values of threshold on the mean of the votes.

For the k-TSP we computed the ROC curve as follow. First we chose $M \in \{0.25, 0.5, 0.75\}$, this stands for the point of decision in the voting system. For example, if $M = 0.25$, this means that at least 25% of the TSPs present in the k-TSP need to classify an individual to the group C_2 in order that the k-TSP classifies this individual in the group C_2. And equivently for $M = 0.5$ and $M = 0.75$. We note that the original k-TSP is a special case when $M = 0.5$. Once M is chosen, we must construct a sequence of points defined by

$$\mathcal{C} = \big\{ -A + c * A |\ c = 0, \frac{a}{2*A}, \frac{2*a}{2*A}, \ldots, 2*A \big\},$$

where $A = \max\limits_{(i,j)\in k-\mathrm{TSP}} |R_i - R_j|$ and a is a smoothing parameter. The notation $(i,j) \in k - \mathrm{TSP}$ stands for all the pairs of genes selected by the current k-TSP. We will perturbate the rules of all TSPs presents in the computed k-TSP as follows

$$R_i < R_j + C,\ C \in \mathcal{C},\ \forall (i,j) \in k - \mathrm{TSP},$$
$$\text{or } R_i > R_j + C,$$

depending on the original definition of the classification rule.

For each $C \in \mathcal{C}$ we compute the sensitivity and the specificity of the individuals out of the current bootstrap. Changing C will give different values of the sensibility and the specificity. This will allow us to compute a ROC curve on the current bootstrap. If several values of the sensibility appear for the same specificity we take the mean of the sensibilities.

The sequence $\mathcal{C}$ was constructed such that all the perturbated rules were able to achieve a sensiblitiy and a specificity of 0, i.e., to join the two extreme points of the ROC curve. The parameter a was chosen such that the sequence $\mathcal{C}$ contains enough points to make the ROC curve as smooth as possible.

The graphs show that the ROC curves on the leukemia cancer dataset for different values of M are roughly equivalent. We note that, for $M = 0.25$, the sensiblity increases faster than on the other graphs. Comparing these graphs with their respective AUC (area under the curve) makes the value of M interesless. However, we will see later that the choice of M can have an influence on the ROC curve. If we consider the median of the sensitivity for differents choices of M, we see that it varies between 0.90 and 0.94 with a mean around 0.93. For the TSP we found that the median of the sensibility was equal to 0.95, which is a little bit higher. Nevertheless this difference is not significative. Indeed if we look on the boxplot of the ROC curve, we note that the variance is big enough to explain this difference.

Along this subsection we saw how the k-TSP could be sensible to perturbations on the dataset and that these pertubations could cause changes on the chosen pair of genes as well as the number of pairs we choose. We will analyse this problem more precisely in the next subsection.

3.2.2 Analytical study of the robustness

The goal of this subsection is to analyse the sensibility of the k-TSP to perturbations of the dataset. There are mainly two points of interest. The first one is how the pairs of genes yield by the k-TSP may change when perturbations in the dataset occur. The second is the number of pairs of genes yield by the k-TSP. The first point is similar to the study we made on the sensibilty of the TSP as the pairs chosen depend on their score. The perturbations will principaly have an impact on the scores and thus produce changes in the ordered list computed in the algorithm 1. This will make the k-TSP to select different pairs of genes. The second point is much more difficult to analyse. Indeed, the value k is chosen by crossvalidation on the dataset, which makes the mathematical analysis of the sensibility quite hard. We will biriefly show the problem dued to this crossvalidation and give some clues to solve it.

First we discuss the notion of sentivity curve. We define it in the same way as for the TSP

$$\mathrm{SC}_{k-\mathrm{TSP}} = I\big(pair(k - \mathrm{TSP}_{x_0}) = pair(k - \mathrm{TSP})\big),$$

where the function $pair(\cdot)$ returns the pair of the k-TSP, $I(\cdot)$ represents the indicator function and $k - \mathrm{TSP}_{x_0}$ stands for the k-TSP on the dataset including the observation (x_0, y_0).

The sensivitiy curve for the k-TSP will have the value 1 if the results don't change when adding the observation x_0 and the value 0 if changes occur. It is obvious that if k changes, then $\mathrm{SC}_{k-\mathrm{TSP}} = 0$.

The value of the breakdown point is determined straighforward from the breakdown point of the TSP. Indeed, the TSP is a special case of the k-TSP when $k = 1$. Thus the breakdown point of the k-TSP can be bigger than the one of the TSP and, for this reason, it is also 0.

Sensibility of the chosen pairs of genes We begin to analyse the sensiblity of the chosen pairs of genes as the dataset is perturbated. We suppose k to be fixed and define the following quantities. We define $\mathcal{M}$ as a list of pairs that are selected by the k-TSP, actually this are the gene pairs present in the k-TSP. We denote by (i,j) the pair from the list $\mathcal{M}$ with the lowest score Δ. Let (o,p) represents the pair with the highest score Δ on the set of the pairs that weren't selected by the k-TSP (we have $(o,p) \notin \mathcal{M}$). The pairs (i,j) and (k,l) are actually two consecutive elements of the list of the ordered pairs with respect to their scores Δ and Γ defined in the algorithm 1. The pair (i,j) represents the point where the k-TSP stops to select pairs. The analyse of the sensibility of the k-TSP will be made in the same way as for the TSP. We add a new observation (x_0, y_0) to the dataset and analyse the changes produced on the results of the k-TSP. We use the following notations, n_1 (resp. n_2) is the number of observation in the group C_1 (resp. C_2). And l_{im} stands for the number of individuals in the group C_i that satisfies the classification rule of the m-th pair of genes of the k-TSP, $m = 1, \dots, k$ and $i = 1, 2$. The score Δ is defined also by

$$\Delta_{u,v} = \left| \frac{l_{1m}}{n_1} - \frac{l_{2m}}{n_2} \right|,$$

where (u,v) is the m-th pair of $\mathcal{M}$. Let us suppose that $y_0 = 1$ and that the individual (x_0, y_0) satisfies the classification rule of the TSP based on the pair (o,p). The score of this pair won't decrease and may be selected by the k-TSP. In order to be selected by the k-TSP its score Δ has to become bigger than at least one of the scores of the genes pairs present in the original k-TSP. The pair (o,p) will be selected by the k-TSP if the inequality

$$\Delta_{u,v}^{(x_0)} > \min_{(i,j)\in\mathcal{M}} \Delta_{i,j}^{(x_0)}$$

is verified, where $\Delta_{i,j}^{(x_0)}$ stands for the score of the pair (i,j) on the dataset containing the observation (x_0, y_0).

It is obvious that if the new individual (x_0, y_0) satisfies all the classification rules for the pairs in $\mathcal{M}$, the ordered list of the pairs computed by the algorithm won't change and this observation won't have any influence. On the other hand, if for one pair the new observation doesn't satisfy the classification rule then its score will decrease and it is possible that this pair of genes will be replaced by the pair (o,p) and the list will be modified. This will result in changes in the pairs yield by the k-TSP. To determine if the addition of a new observation will produces changes, one must determine the pair within the list $\mathcal{M}$ with the lowest score Δ for which the new observation would be missclassified. And then apply the function (8) for this pair and the pair (o,p) and study the sign of this function. This determines if changes in the results of the k-TSP will occur or not. If we are interested in the worst case, i.e., the lowest quantities of data one can add such that the k-TSP remain unchanged, we must use again a function defined previously for the TSP. Indeed, we saw that changes occur if the score of the top pair over the set of non selected genes pair will become higher than one of the pairs in the set $\mathcal{M}$. The easiest is to choose the last pair of $\mathcal{M}$ as it will be the one with the lowest score. The number of observation that we need to add to produce changes is given by applying the function (9) on the pairs (o,p) and (i,j).

We saw that the sensibility of the k-TSP to perturbations in the dataset possesses similarities with the sensiblity of the TSP, this is obvious as the k-TSP is composed of k TSPs. The sensiblity of the k-TSP is highly related to the properties of the dataset. Indeed, we saw that it depends on the values of the score Δ for the pair in the list $\mathcal{M}$ and the best pair that doesn't belong to $\mathcal{M}$. This remark will be used in the next subsection to derive a new way to determine the value k to make the k-TSP less sensitive to pertubations in the dataset.

Sensiblity of the number of chosen pairs The number of chosen pairs is given by k. In our case we defined k through crossvalidation, i.e., if we define S as the entire dataset, we divided it into $V_1, V_2, \ldots, V_{10}$ non overlapping subsets of approximately equal size, see section 3.1.1 for more details on the crossvalidation procedure. The value for k was chosen as follows

$$k = \underset{k=1,3,5,7,9}{\arg\max} \left\{ \frac{1}{n} \sum_{i=1}^{10} 1\Big(\text{predict}\big(k\text{-TSP}^{(-V_i)}\big)(V_i) = \text{grp}(V_i)\Big) \right\}, \tag{13}$$

where k-TSP$^{(-V_i)}$ represents the k-TSP based on the set $S \setminus V_i$, the function predict$\big(k\text{-TSP}^{(-V_i)}\big)(V_i)$ represents the prediction of the observation from V_i based on k-TSP$^{(-V_i)}$. Finaly the function grp(V_i) gives the group to which the individual in V_i belongs. We seek k that maximizes the correct rate prediction.

If we add a new individual, one should look for its impact on the computation of k-TSP$^{(-V_i)}$ when this individual doesn't belong to the subset V_i and to its impact on the correct rate prediction when it belongs to V_i.

3.3 Another way to choose k

In this subsection we present another approach to determine the value of k. We discussed in a previous paragraph how the distance between the score of the last element in $\mathcal{M}$ and the highest score for the pairs not in $\mathcal{M}$ is related to the sensibility of the k-TSP. Indeed, the lowest this distance was, the fewest observations were needed to produce changes in the selected pairs of genes (here the observations were chosen in the worst case, i.e., favorable for the pair out of $\mathcal{M}$ and unfavorable for the pairs in $\mathcal{M}$). For $k = 1, 3, 5, 7, 9$, we define $\mathcal{M}_k$ the ordered list of pairs of genes selected by the k-TSP. Let us denote the distance between the last pair from M_k and the best pair not in $\mathcal{M}_k$ by dist_k. The idea is to choose k such that dist_k is maximum.

Figure 10 shows a plot of the score for the 9 best pairs computed on the leukemia cancer as well as the distance dist_k for $k = 1, 2, 3, 4, 5$. The highest score Δ is 0.98 and then slowly drops until 0.92 for the nineth pair. There are no big jump were we could clearly choose the value of k. The second graph shows the values of these jumps where it would be possible to choose a value of k such that the prediction of observations are clearly defined (k is an odd number). The two first points are the highest one, it is not obvious on this graph that the second has a higher value than the first one. If we choose k such that the distance dist_k is maximum, we would choose here $k = 3$.

Figure 11 shows the results of the k-TSP applied on the 500 bootstraps used to obtain the previous results on the leukmia cancer dataset, in this case we used the alternative way to select k. The results of the frequency of appearance for the single genes are roughly the same. However, the results of the frequency of appearance of the gene pairs are slightly different. The frequency for the pair $(463, 720)$ went down by 5%, from 0.35 to 0.3. The frequency for the pairs $(923, 1900)$, $(902, 1972)$, $(2318, 2614)$ and $(942, 2130)$ also decreases by 5%. Moreover, there are less pairs of genes whose frequency exceeded the chosen threshold of 8%. This is mainly explained by the histogram of the values of k. Indeed, we see on the last graph that the number of times that the methods chose lower k increased. The frequency of $k = 1$ doubled, from 20% to 40%. For all the other values it went down, for example, for $k = 3$ from 30% to 20%. The decrease of the frequency of appearance for the pairs of genes is directly explained by the fact that lower values for k were chosen. Indeed, if k is lower this means that less pairs are present along the bootstraps and results in lower frequency.

The histogram of k shows that the values of k are more stable, and lay 60% of the

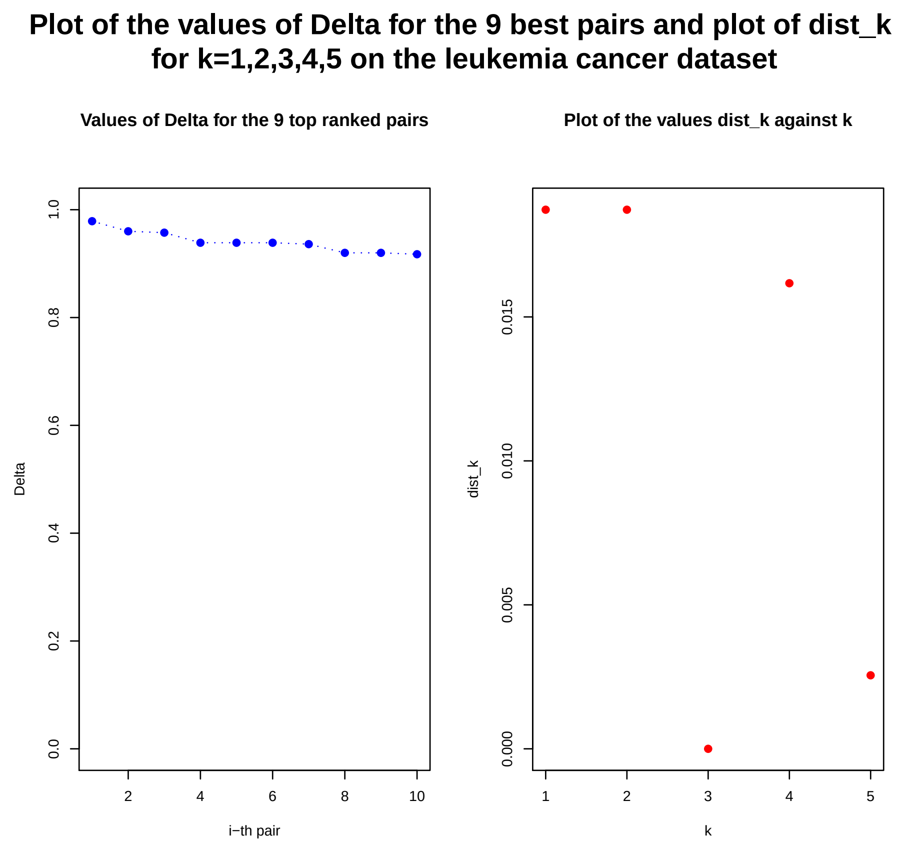

Figure 10: The first graph shows values of Δ for the 9 top ranked pairs. The second graph plots the values of dist_k for $k = 1, 2, 3, 4, 5$.

time between 1 and 3. This result depends on the dataset we analysed. Indeed, we saw in Figure 10 that good values for k which would maximize the distance dist_k would be 1 or 3.

Figure 11 shows big improvement of the stability of k with respect to the last results. Nevertheless, we have to be careful that the gain we made on stability will inevitably result in a loss of accuracy as we replaced a rule to choose k that maximized the accuracy by a rule whose aim was to stabilize the values of k.

In the next section we present another extension of the TSP. It is based on the same idea as the the k-TSP, to choose several pairs of genes. The difference is in the way of computing the probabilities $p_{ij}(C_i)$. Indeed it involves the ratio of the gene expressions instead of the difference.

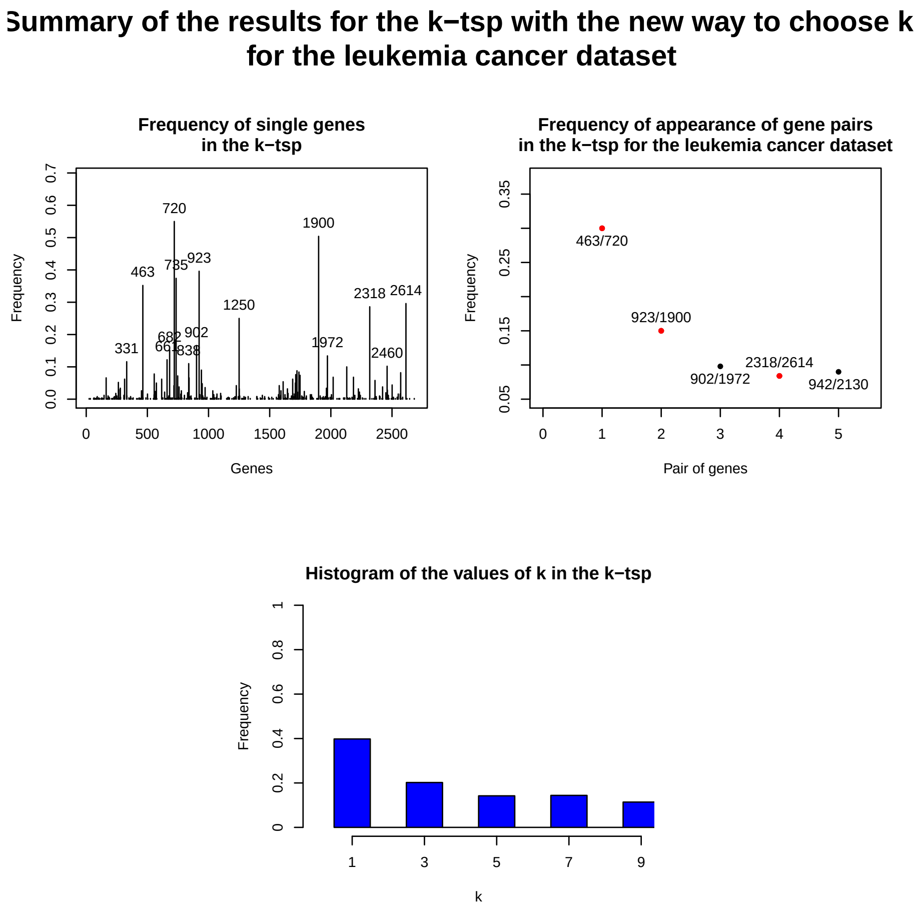

Figure 11: Results of the k-TSP on the leukemia cancer dataset with the proposed way to choose k. The red dots on the second graph represent the pairs of genes selected with the k-TSP on the original dataset with the alternative solution to determine k.

4 The second extension of the TSP: the weighted k-TSP

Microarray datasets usually tend to have huge amount of genes. This leads to relatively high computational complexity, in order to reduce it feature selection is applied before the classification. Unfortunately, k-TSP is pretty sensitive to the feature selection and low number of data may affect significantly the rank and thus have a big impact on the computation and finally result on lower accuracy of the method. The k-TSP also “misses” a part of the information. Indeed let us suppose that two genes are highly correlated with the presence of cancer, let us call them G_1 and G_2. Suppose that in healthy sample the expressions of genes range are $G_1 \in [1, 100]$ and $G_2 \in [1000, 1200]$ and in cancer sample we measure $G_1 \in [500, 800]$ and $G_2 \in [2000, 2200]$. The k-TSP method will never detect this pair because the expressions are in the same order for the case of healthy and cancer samples. This drawback of the method is corrected in the extension, namely the Weighted k-TSP (WTSP). Instead of searching for inverse relative expression it is based on comparison of percentage changes of genes expression in pairs between the different classes.

4.1 Definition of WTSP

Let us denote by S the average values quotient genes in each pair from the training sample. For each pair of gene $(i, j),\ i, j \in \{1, \ldots, P\}, i \neq j$ a single element from S can be written as

$$S_{ij} = \frac{\sum_{m=1}^{N} x_{im}/x_{jm}}{N}.$$

Weight k-TSP method focus on finding pairs of genes (i, j) such that the probability of the event $\{x_{in}/x_{jn} < S_{ij}\},\ n = 1, \ldots, N$ is very different whether an individual n belongs to class C_1 or C_2. The procedure follows the same strategy as for the k-TSP. The probability of this event is computed by frequency of appearance, mathematically this can be described as

$$p_{ij}(C_1) = \frac{1}{|C_1|} \sum_{n=1}^{N_1} I(x_{in}/x_{jn} < S_{ij}),$$

$$p_{ij}(C_2) = \frac{1}{|C_2|} \sum_{n=1}^{N_2} I(x_{in}/x_{jn} < S_{ij}),$$

where $I(\cdot)$ is the indicator function and $|C_i|$ represents the number of observation in class i, also written as N_i. Without loss of generality we supposed that the first N_1 observations come from class C_1 and the last $N - N_1$ from class C_2.

We define the scoreΔ_{ij} of a genes pair (i, j) as for the k-TSP, i.e.

$$\Delta_{ij} = |p_{ij}(C_1) - p_{ij}(C_2)|.$$

This is the first criteria to order the pairs of genes. As for the k-TSP it may happen that several pairs reach the same score, so a second criteria is needed. The criteria for the k-TSP was based on the difference of rank within each class. This is no longer of interest. We define the first part of the second criteria as

$$\gamma_{ij}(C_m) = \frac{\sum_{n\in C_m} \frac{x_{in}/x_{jn}}{S_{ij}+x_{in}/x_{jn}}}{|C_m|},$$

where we added S_{ij} to avoid very small values of the divisor, with this restriction all values belong to $]0,1]$. Finally the second criteria is expressed as

$$\Gamma_{ij} = |\gamma_{ij}(C_1) - \gamma_{ij}(C_2)|.$$

Now we proceed exactly in the same way as for the k-TSP but with the new defintion of the scores. We determine k by crossvalidation and select the genes pairs using algorithm 1.

4.2 Classification of a new observation

Similarly to the previous algorithm we choose pairs that provide the highest score. The procedure for the prediction is modified for the weighted k-TSP. The difference lie in the voting algorithm, an extension is proposed. For a new observation x_{n+1}, the pair of genes (i,j) will predict it as follow

$$h(x_{n+1}) = \begin{cases} C_1, & \text{if } x_{i,n+1}/x_{j,n+1} < S_{ij} \\ C_2, & \text{otherwise,} \end{cases} \tag{14}$$

where $x_{i,n+1}$ denotes the expression level of the ith gene from the new observation. We have the opposite situation when $p_{ij}(C_1) < p_{ij}(C_2)$. It is possible to keep equations (11) and (12) to proceed to the prediction of a new observation. Nevertheless we can also use a weighted prediction, where the weights are given in function of the "goodness" of the predictor. In the case where $p_{ij}(C_1) > p_{ij}(C_2)$ we define

$$I_{wg}\big(h_u(x_{\text{new}}) = C_i\big) = \begin{cases} \frac{S_{ij}}{S_{ij}+x_{i,n+1}/x_{j,n+1}}, & \text{if } h_u(x_{n+1}) = C_i \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

If $p_{ij}(C_1) \leq p_{ij}(C_2)$, the wages change as follow

$$I_{wg}\big(h_u(x_{\text{new}}) = C_i\big) = \begin{cases} \frac{x_{i,n+1}/x_{j,n+1}}{S_{ij}+x_{i,n+1}/x_{j,n+1}}, & \text{if } h_u(x_{n+1}) = C_i \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

The final prediction is given as in (12) but the indicator function $I(\cdot)$ is replaced by the weights $I_{wg}(\cdot)$.

The results of robustness for the WTSP would be made in the same way as for the k-TSP, as an ordered list is constructed. The difference comes from the definition of the scores Δ and Γ. The estimation of these parameters is achieved through the use of the original values of the genes expressions. This would be very sensitive to the presence of outliers.

5 Penalized logistic regression

In this section we present a widely used method for classification, the *Penalized logistic regression*, this method is based on a simpler method called *Logistic regression*. The idea of logistic regression is to model the probability that an individual comes from the group C_1, as a function of a linear combination of the explanatory variables $\mathbf{x}_1, \cdots, \mathbf{x}_n$. Logistic regression provides a good method for classification but it doesn't work for microarray data because there are much more variables than observations. A solution to this problem is to reduce the number of variables, this is exactly what the penalized logistic regression does. It adds a penalization on the coefficients in the linear combination. This makes some parameters to be reduced to zero and finally achieve a model with less parameters which can be solved even for microarray data and which is easier to interpret.

The first step of this section is to present the logistic regression and to highlight the problem caused by the property of "large P, small N" present in datasets such as microarray. Then we introduce the solution to this problem, namely the penalized logistic regression and discuss some of its properties. Finally we apply this method on the dataset of leukemia cancer sample.

5.1 Linear Regression

We use the same notations as previous, the dataset is expressed as $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$.

The goal of logistic regression is to model the probability $\mathbb{P}(x)$ that an array with measured gene expression $\mathbf{X} = \mathbf{x}$ represents an observation from class $Y = C_1$, the dependence between the probability and the explanatory variables is a function of a linear combination of the x_i's. Here we wrote $\mathbb{P}(x)$, because the probability we want to model is a function of $\mathbf{x}$. More formaly, this probability is

$$\mathbb{P}(Y = C_1 | \mathbf{X} = \mathbf{x}).$$

For example a simple regression model would be

$$\mathbb{P}(x) = \alpha + \beta x + \epsilon_i, \quad \text{for } i = 1, \dots, n,$$

with α, $\beta \in \mathbb{R}$ and where n is the number of observation and ϵ stands for the error term. The error is assumed to be independent and normaly distributed with mean 0 and unknown standard deviation σ. The goal is to estimate the regression coefficients (α, β). We want that our model is as much accurate as possible, this means that the $\hat{y}_i$ estimated with $\hat{\alpha} + \hat{\beta} x_i$ will be as close as possible to the true value y_i. We define the residual of observation i to be

$$r_i = y_i - \hat{y}_i = y_i - (\hat{\alpha} + \hat{\beta} x_i).$$

The fitting will be the best if the r_i are "small". The most common way to estimate the regression coefficients is the method of *least squares* (LS), which consists in estimating $(\hat{\alpha}, \hat{\beta})$ to minimize the residual sum of square, i.e.,

$$(\hat{\alpha}, \hat{\beta}) = \underset{(\alpha,\beta)}{\arg\min} \sum_{i=1}^{n} r_i^2 = \underset{(\alpha,\beta)}{\arg\min} \sum_{i=1}^{n} \big(y_i - (\alpha + \beta x_i)\big).$$

Differentiating this equation with respect to α and β and equalizing them to 0 provides the solutions

$$\hat{\alpha} = \bar{y} - \alpha \bar{x}, \quad \hat{\beta} = \frac{\sum y_i (x_i - \bar{x})}{\sum (x_i - \bar{x})^2},$$

where $\bar{x} = \frac{1}{n} \sum x_i$ and $\bar{y} = \frac{1}{n} \sum y_i$.

Here we choose to minimize the residual sum of squares (RSS) of the residuals to estimate the regression coefficients. The square in the RSS makes the estimation very sensitive to outliers, as an outlier will have a high residual, the square will make it even bigger. This problem is clearly presented and discussed in [7]. A solution to the weakness of the least squares is to use a L^1 norm in the RSS, i.e., we now need to solve

$$(\hat{\alpha}, \hat{\beta}) = \underset{(\alpha,\beta)}{\arg\min} \sum_{i=1}^{n} |r_i|.$$

This estimator is the *least absolute deviation* (LAD) or L^1 regression. It is known to be much less sensitive to outliers in the dataset. There exists other solution to robustify the LS method, as using a L^q norm, with $1 < q < 2$, etc.

Once the paramaters α and β are estimated, the classification of a new observation denoted as x_{n+1} is defined through the probability $\mathbb{P}(x_{n+1})$ to belong to the group C_1, estimated by $\hat{\mathbb{P}}(x_{n+1}) = \hat{\alpha}+\hat{\beta}x_i$. An interesting property of the linear regression is that the prediction is given through a probability and gives an indication on how likely the classification to one group is.

Nevertheless the method defined up to this point presents two problems. The value of the probability $\mathbb{P}(x)$ can become negative for special values of the explanatory variables. The second problem is that this probability can exceed one, this gives nonsense to the prediction. A solution is to modifiy the relation between the probability $\mathbb{P}(x)$ and the linear combination of the parameters. This is the topic of the next section.

5.2 Logistic regression

To avoid the problems $\mathbb{P}(x) < 0$ and $\mathbb{P}(x) > 1$, we define probability $\mathbb{P}(x)$ as a function of the linear combination of the explanatory variables or, similarly, the linear combination of the explanatory variables as a function of the probability $\mathbb{P}(x)$. Such function is called a *link function* and can have several definitions.

We transform the probability function $\mathbb{P}(x)$ into the function $\nu(x)$ given by

$$\nu(x) = \log\left(\frac{\mathbb{P}(x)}{1-\mathbb{P}(x)}\right) = \alpha + \beta x.$$

If we invert this function we find $\mathbb{P}(x)$ as a function of the explanatory variables through the relation

$$\mathbb{P}(x) = \frac{1}{1+\exp\big(-\nu(x)\big)}\ .$$

This link function is called the logistic function. There exists other well known link function as the complementary log-log and the normal link function. In this report we will focus only on the logistic function.

Until this point we considered only a model with one explanatory variable, it is straightforward to extend the model to a model with multiple explanatory variables. We introduce the explanatory variables $x_1, \cdots, x_P$ and define

$$\nu(x_1, \cdots, x_P) = \log\left(\frac{\mathbb{P}(\mathbf{x})}{1-\mathbb{P}(\mathbf{x})}\right) = \alpha + \sum_{i=1}^{P} \beta_i x_i.$$

The estimation of the parameters (α, β) will be computed as for the case of the simple regression.

5.3 The penalization

It is possible that the logistic regression provides no zero coefficients and thus the number of coefficients β_i will be equal to the number of explanatory variables, which can become very big for some dataset. In this paper we considered two microarray dataset, which contained several thousands of genes. The logistic regression would be hard to implement and the results would be very complex to interpret. We thus need to reduce the number of non zero coefficients. This is achieved through adding a penalization on the coefficients. The most common penalization is to add the L^1 norm or L^2 norm in the computation of the likelihood. The likelihood is defined as

$$l(\alpha, \beta) = \prod_{i=1}^{n} \mathbb{P}(x_i)^{y_i} \big(1 - \mathbb{P}(x_i)\big)^{y_i},$$

and the log-likelihood as

$$L(\alpha, \beta) = \sum_{i=1}^{n} y_i \log \mathbb{P}(x_i) + y_i \log \big(1 - \mathbb{P}(x_i)\big).$$

The approach of minimizing the RSS is equivalent to maximize the likelihood or the log-likelihood, all yielding the same estimator. To add the penalization we need to work with the log-likelihood. We define the estimators of the penalized logistic regression as the solutions of

$$(\hat{\alpha}, \hat{\beta}) = \underset{(\alpha,\beta)}{\arg\min} \; -L(\alpha, \beta) + \frac{\lambda}{2} J(\alpha, \beta), \tag{17}$$

where the function $J(\cdot)$ represents the penalty on the parameters. It is common to choose the L^2 norm, i.e.,

$$J(\alpha, \beta) = ||(\alpha, \beta)||_2^2,$$

the norm L^1 or L^q is also common. Actually there are arbitrarily many choices of the penalization function, as also mixture of different norms. Once the parameter λ is chosen, the equation (17) can be solved either by direct calculation or in more complicated case with the Newton-Raphson algorithm, see [8] for more details.

We note that when $\lambda = 0$, the penalized logistic regression is the simple linear regression. As λ increases the penalized regression estimates will be shrinked towards 0. There are different ways to choose λ. Indeed one may want that the error prediction rate is minimzed by the choice of λ, other may wish that the deviance of the model will be minimzed (or the RSS) with a restriction on the number of parameters, or to maximize the area under the curve (AUC) for the respective ROC curve, etc.

We used the R package *glmnet* from [9] to compute the penalized logistic regression estimators for the leukemia cancer dataset. It computes the values of the penalized likelihood for a sequence of values for λ and then chooses the λ that minimizes a criteria. The most common criteria are the one presented in the last paragraph.

We applied the method of penalized logistic regression on the leukemia cancer dataset. Altough this dataset was quite big, the glmnet was very fast to compute the coefficients path for different values of λ.

Figure 12 shows a plot of the penalized regression coefficients against a sequence of values λ. We see that some coefficients that had a big influence when λ was large (translated by a big β) are shrinked toward 0 as the values of λ increase.

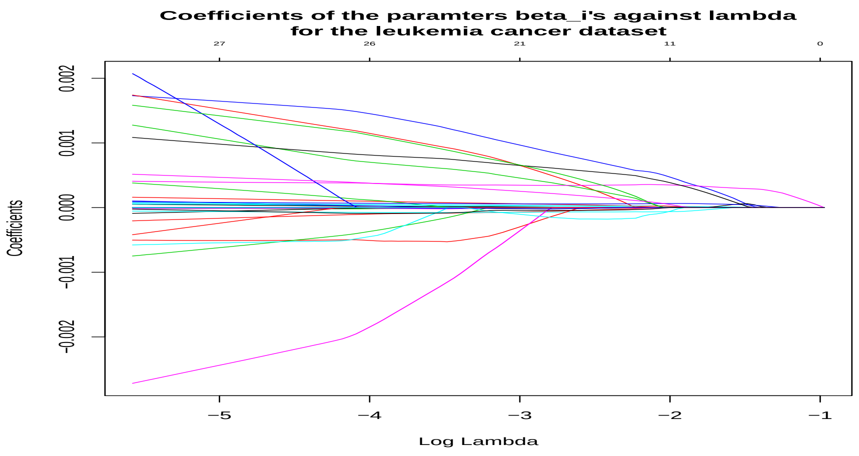

Figure 12: Plot of the coefficients for the parameter β_i's against different values of λ for the Leukemia cancer dataset.

We have values of the coefficients for different values of λ. We now need to choose a criteria to compute λ. In our example, the response is a binary variable and represents a group to which the sample belongs. Here, the goal is to derive a powerful classification method whose error prediction rate is as low as possible. It is then natural to choose as criteria for λ to minimize the error rate prediction.

The package glmnet contains a function *cv.glmnet* which compute cross validation and seeks for values of λ which minimze a given criteria. We used $type = "class"$ in this function to specify that we wish to minimize the error prediction rate.

Figure 13 shows a plot of the missclassification error rate against different values of λ. The R function chose $\log(\lambda) = -4.04$ as value to minimize the error prediction rate. We used again the glmnet function with the chosen value for λ. This provided 27 non zero coefficients for the β_i's including the intercept α.

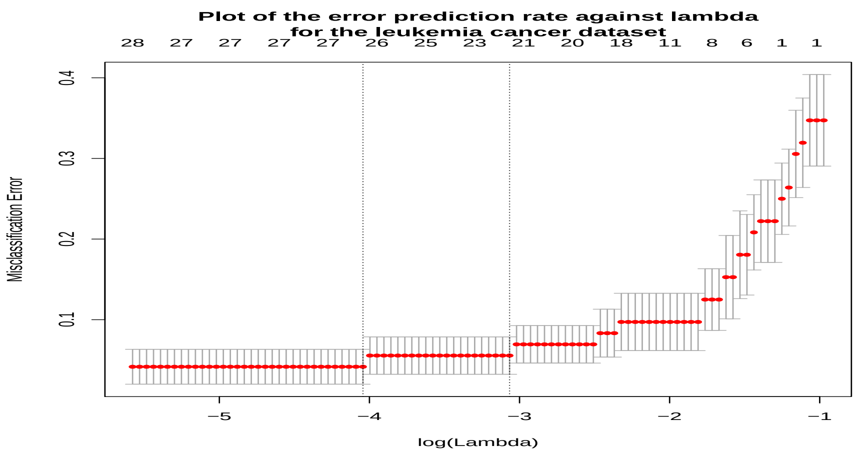

Figure 13: Plor of the missclassification error rate for differente values of λ for the leukemia cancer dataset.

The results are shown in Table 5. We see that the sign of the coefficients can be as well positive as negative. This is of particular interest. Indeed, if a coefficient has a negative value, then high values of the corresponding variable (positive value) will significantly decrease the value of the linear combination and will tend the estimation to one or the other group. It gives an indication to the relation between genes and classes, if high value of the genes will make the patients to be classified to C_1 with a higher (or lower) probability.

The magnitude of the coefficients may also play a role. If a coefficient has a high value (in absolute value) then the bigger the value of the corresponding gene expression the higher (or lower) the probability will be. One must be careful when using this remark. The genes expressions don't have the same distribution and can be big for some genes and low for others. This may influe the magnitude of the coefficients as well.

Penalized logistic coefficients					
gene	value	gene	value	gene	value
intercept	-5.031882e+00	331	-1.041703e-04	501	-1.902320e-05
658	2.835025e-05	682	2.157503e-05	693	9.990825e-05
699	1.233826e-04	720	1.467988e-03	735	4.738104e-05
759	-8.489749e-06	761	-6.389065e-06	767	1.166342e-03
838	2.682606e-05	902	4.689704e-05	1491	7.106581e-04
1637	6.745307e-05	1688	-7.389820e-05	1900	3.782610e-04
1900	8.182355e-04	1938	-5.032853e-04	1947	-3.847228e-04
2073	-4.646599e-04	2223	-1.898748e-03	2363	1.140061e-03
2398	-1.278422e-06	2512	3.819823e-04	2614	-9.039997e-05

Table 5: Coefficients for the penalized logistic regression on the leukemia cancer dataset.

Figure 14 presents the ROC curve for the penalized logistic regression on the leukemia cancer dataset. We performed 200 bootstraps of the original dataset. At each step we fitted a penalized logistic regression on the bootstraped sample and computed the probability that the individual that weren't present in the bootstrap had to be classified to the group C_1. We used the R package [10] in order to compute the ROC curve presented here. This package also allowed us to compute the median of the ROC curve over the 200 bootstraps, which stands at 0.97.

We presented the theory about the penalized logistic regression and discussed some of its properties. We presented also the tools available in R to fit this model on a dataset and to obtain estimation of the accuracy.

Along this paper we presented three methods to classify individuals based on their genes expressions. We performed them all on the leukemia cancer dataset. We also computed their ROC curve to give a notion of performance of the method. In the next two sections, we will compare these methods on two different datasets. The first one is the well known leukemia cancer dataset used along this paper to illustrate our methods. The second one is a dataset from the start up with which this paper was writen. We note that the dataset used is not the most recent one and not the best one.

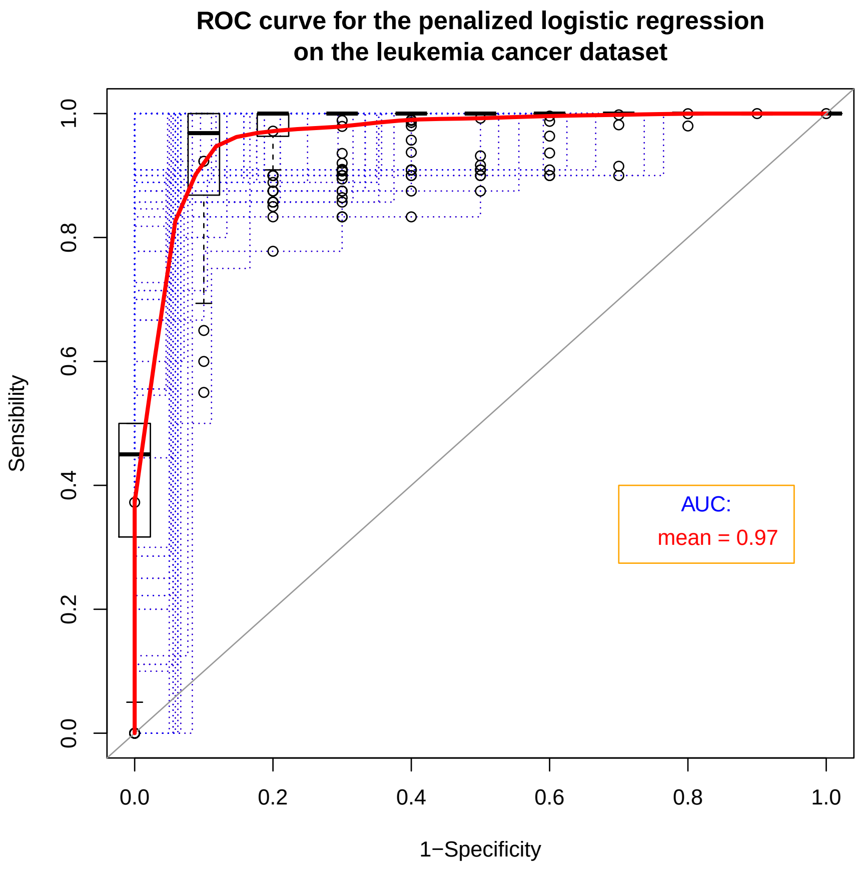

Figure 14: ROC curve for the penalized logistic regression on the leukemia cancer dataset.

6 Comparison of the methods on the leukemia cancer dataset

The aim of this section is to compare the efficiency of the TSP, the k-TSP and the penalized logistic regression (PLR) on the leukemia cancer dataset. Along the paper we presented and discussed the results of these methods on this dataset. The comparison will be principaly based on the ROC curve and the AUC from these curves and which pairs of genes one would select using these methods.

The AUC were 0.95, 0.92 (took mean over the different values of M) and 0.97 for the TSP, the k-TSP and the PLR respectively. This shows that the three methods performed well. This difference in the values of the AUC is not significative compared to the variance presented by the boxplots. Nevertheless it is surprising that the AUC of the TSP is higher than the AUC of the k-TSP. The results of the TSP on the original dataset as the results of the bootstraps suggest that the best pair of genes would be $(463, 720)$. The k-TSP would also select this pair, but would add the pairs $(923, 1900)$ and $(902, 1972)$, which showed again the flexibility of the k-TSP to allow to choose several pairs of interest. For the PLR, 27 genes were selected. Among these genes only the genes 720, 902 and 1900 are present in the k-TSP and the PLR. These genes appear in different pairs and never interact together. This makes stronger the assumption we made previously that some genes are of interest only when compared to another one, this seems to be the case for the genes 463, 923 and 1972. The number of genes selected by the methods is 2, 6 and 27 for the TSP, the k-TSP and the PLR respectively. Altough the number 27 looks big with respect to 6, it is still very low compared to the number of genes in the original dataset (2971 after reduction).

These methods all possesse their advantages and disadvantages presented along this paper. The best solution would be to choose the genes with respect to the properties we want the model to have and using all the methods to select the genes of interest.

In the next section we present a new dataset and apply on it the methods we presented along this paper (except the WTSP).

7 Diagnoplex Dataset

The results presented in this paper were developed in collaboration with Diagnoplex. Diagnoplex is a start-up located in the norh of Lausanne (CH) constitued of around 18 employees. They are developing a biological test whose aim is to detect the colon cancer at an early stage based uniquely on genes expressions. They use several mathematical methods of classification to build their product.

In this section we present the results of the TSP, the k-TSP and the penalized logistic regression on one of their datasets. This dataset is composed of 93 individuals, equaly split in three groups C_1, C_2 and C_3 also labeled as $1, 2$ and 3. The groups are defined as follow, the group C_1 is composed of healthy patients, the group C_2 of patients who developed polyps and C_3 of patients with a colon cancer. Their goal is to develop two classifiers. The first one should classify an individual between the groups C_1 and C_2. The second one between the groups C_1 and C_3. An individual would be sent to conduct a coloscopy if it hasn't been classified to the group C_1 for both classifiers.

To determine which genes were important we made 3 analysis. In the first one we seek for a classifier between groups C_1 and C_3, in the second for a classifier between groups C_1 and C_2 and finally the last one between the groups C_1 and the group C_{23} constitued of the groups C_2 and C_3. Here we will present the results of the 3 analysis. We will mainly discuss the results of the first analysis and then draw general comments and conclusions over the 3 analysis.

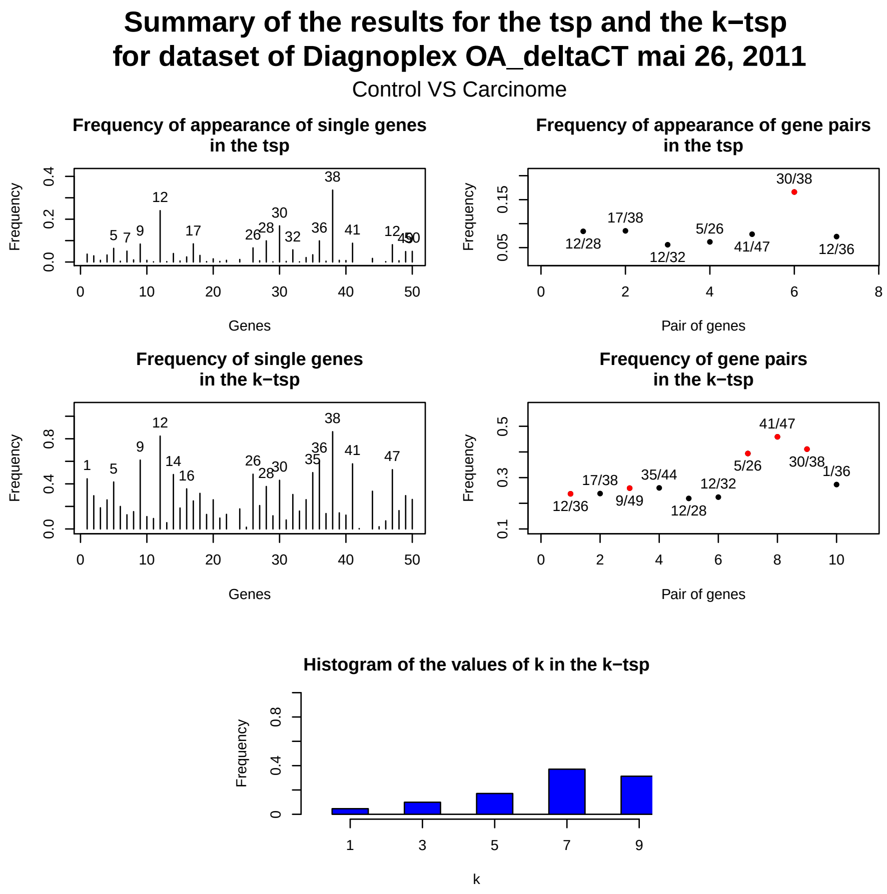

Figure 15: Summary of the results of the TSP and the k-TSP for the groups C_1 and C_3. The red points represent the results of the TSP and the k-TSP applied on the original dataset.

Figure 15 presents the results of the TSP and the k-TSP for the groups C_1 and C_3. For the results of the TSP, the genes 12 and 38 are the one that appear the most often on the single gene pairs. The pair that appears the most often is clearly the pair $(30, 38)$ with a frequency about 18%. This pair was also the one selected by the TSP on the original dataset. For the k-TSP the genes 12 and 36 are equaly the one that appear the most often, but there are a lot of other genes whose frequency of appearance are quite high. The pair $(30, 38)$ is ranked second on the appearance rate, while the pair $(41, 47)$ is ranked first. This pair also appeared on the graph of the frequency of appearance for the TSP, but was ranked fourth. The k-TSP highlighted the importance of this pair, as it appears very often among the boostrap but wasn't ranked first the majority of the time. Probably for this reason the TSP didn't select it. The pairs $(5, 25)$ and $(30, 38)$ appearead also often over the bootstraps, one of this pair was ranked first in the results the TSP, while the other on was ranked fifth. There are 7 other pairs of genes who appeared quite often, their frequency of appearance was about 22%. Finally the histogram for the values of k shows that the value $k = 7$ was chosen the majority of the time, around 40% while $k = 9$ was chosen 35% of the time. This suggests to choose several pairs of genes. For example, in this case, one solution would be to choose the 7 pairs that appeared the most often over the bootstraps as the genes of importance to discriminate between the classes C_1 and C_3.

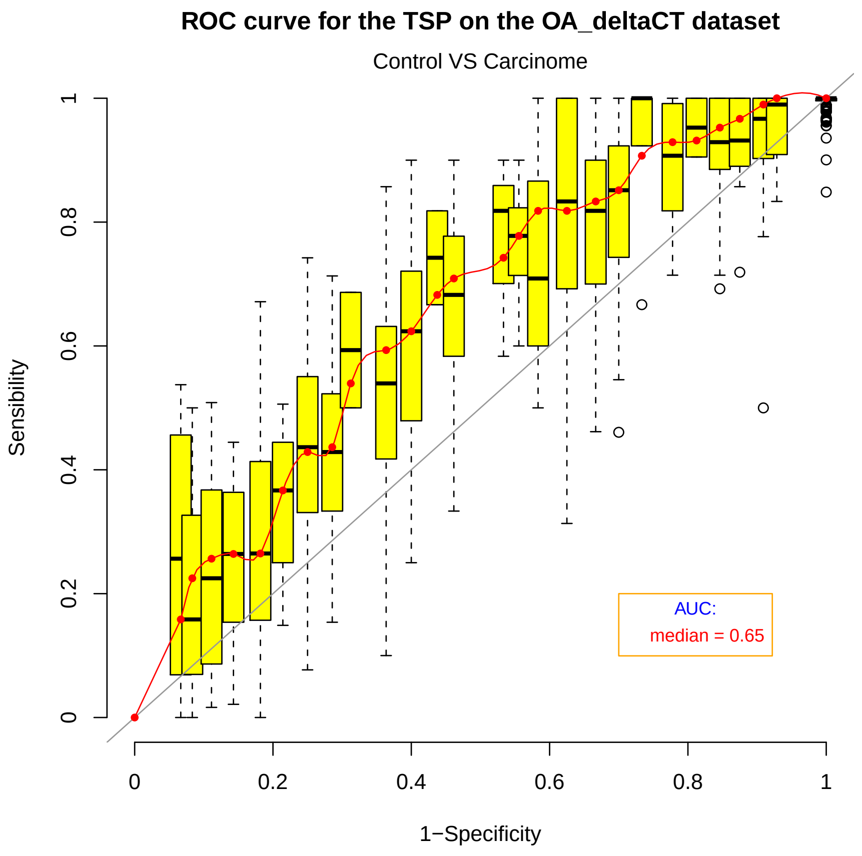

Figure 16: ROC curve of the TSP for groups C_1 and C_3. The red line represents the median for the 200 ROC curves.

Figure 16 and 17 show estimated ROC curves for the TSP and for the k-TSP with different voting threshold for the groups C_1 and C_3 over 200 bootstrap resamples of the original dataset. For more details on the estimation of these curves see section 3.2.1.

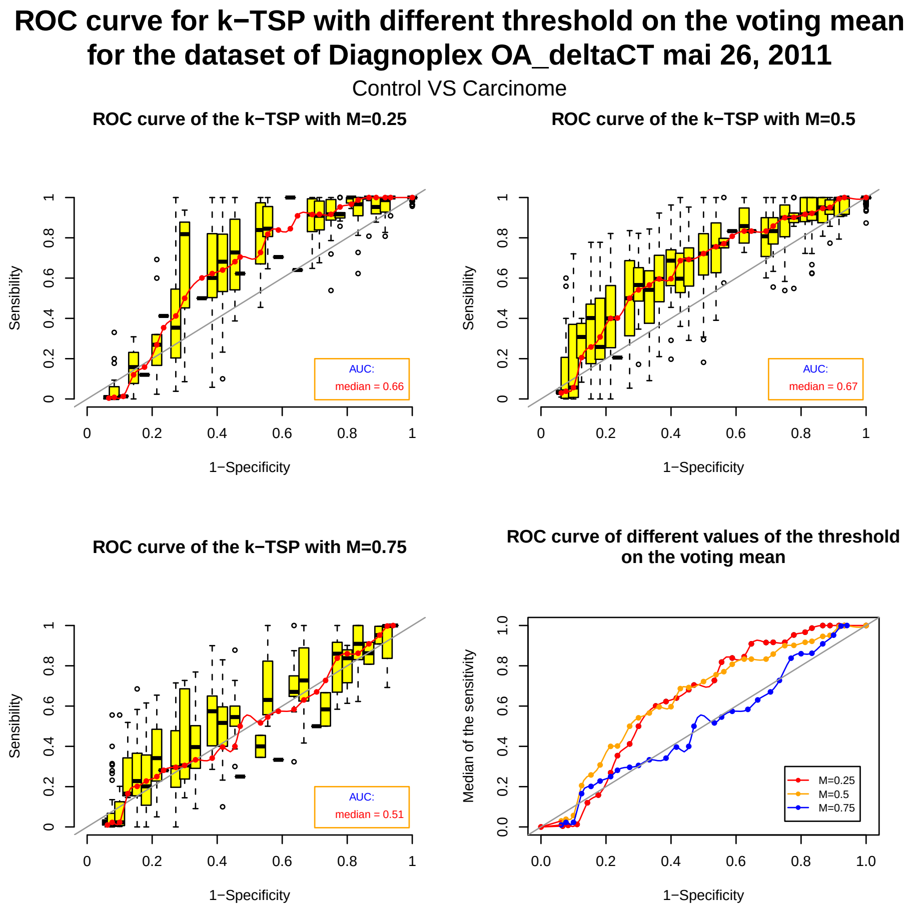

Figure 17: ROC curve of the k-TSP for groups C_1 and C_3. The red line represents the median for the 200 ROC curves.

On the second graph, based on the best AUC one will choose $M = 0.5$, i.e., the orignal k-TSP, as its AUC is the highest one (with a value of 0.67). Moreover the AUC for the TSP is 0.65, in this case the values of the AUC are as expected, the k-TSP has an higher AUC than the TSP. We see that, for $M = 0.25$, the AUC is near the AUC for $M = 0.5$. Thus it wouldn't have a big influence to choose one of this M or the other. By contrast, choosing $M = 0.75$ would significantly reduce the AUC to 0.51 which is really bad. Indeed this means that throwing a coin to perform the classification would have almost the same classification accuracy.

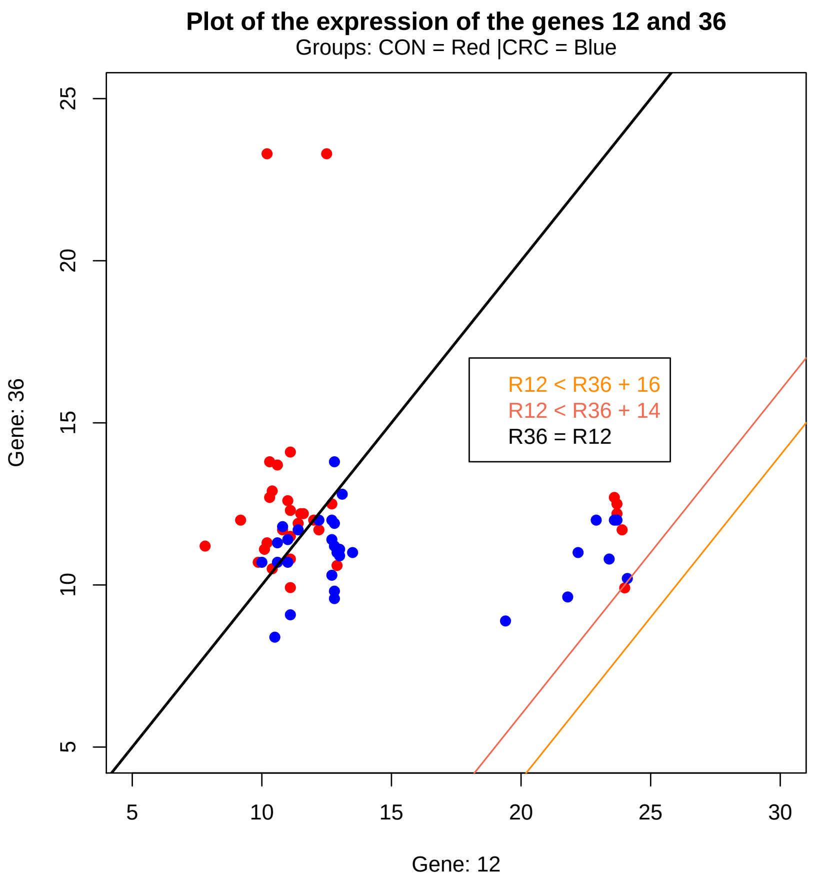

Figure 18: Plot of the values of the genes 12 and 36 with perturbated classification rules.

We can observe on the graphs of the ROC curve for the k-TSP that near the point $(0,0)$, the median of the sensibility stays around 0 whereas the specificity decreases. This is an undesirable property of the k-TSP. Indeed we loose accuracy on the healthy people and we don't gain any accuracy on the sick patients. This problem comes from the methods we use to compute the ROC curve. Indeed we decide to replace the rules

$$R_i < R_j,$$

by

$$R_i < R_j + C,$$

where we make C varies such that the points $(0,0)$ and $(1,1)$ are reached by the curve. The problem comes from the distribution of the gene expressions for each groups. Figure 18 shows a plot of the genes 12 and 36 for patients from C_1 and C_3. The red points represents the healthy patients and the blue points the sick patients. In this case we have $p_{12,36}(C_1) > p_{12,36}(C_3)$, thus the classification rule has the usual order, i.e., if $R_{l,12} < R_{l,36}$, classify the individual l in C_1 and in C_3 otherwise. Moving this rule will change the classification. For example, with $C = 16$ (represented by the yellow curve on the graph), we see that all the healthy patients are well classified and the sick patients are missclassified, this means that the sensibility is 0 and the specificity is 1. This represents the points $(0, 0)$ on the ROC curve. If we augment C until $C = 14$ (represented by the pink curve on the graph), i.e., when the lines is between the red and blue point. Here one healthy patients will be missclassified, as well as all the sick patients. The sensibility won't increase and stays at 0. The specifity will deacreases as one healthy patients is missclassified, for this reason the specificity will be $\frac{n_1-1}{n_1}$, this results in the point $\left(0, 1 - \frac{1}{n_1}\right)$ on the graphs. This points is below the line of slop 1 and intercept 0.

Genes used				
1	4	5	6	8
9	11	12	16	18
19	22	26	32	35
36	38	39	41	43
45	48	50		

Table 6: Genes whose regression coefficients were non zero when comparing the groups C_1 and C_3.

This pair of genes appears 12% of the times over the bootstrap of the k-TSP. Such distribution of the data may occur for several pairs of genes, not only for the pair $(12, 36)$. This can partially explain why the ROC curve behaves so bad in a neihgborhood of the point $(0, 0)$.

This pair appeared only 5% of the times for the TSP, wich seems to be too low to perturbate the ROC curve for the TSP. It is also possible that the TSP only takes into account pairs of genes that don't have this particularity.

We performed also the penalized logistic regression on this dataset, the results are shown in Table 6. The intercept was also non-zero but is not present in the table that summarized the results as we are only interested in the genes used and not their coefficients.

Figure 19 shows the ROC curve for the PLR for the groups C_1 and C_3. This curve was computed with the ROCR R-package [10]. The AUC for the PLR is

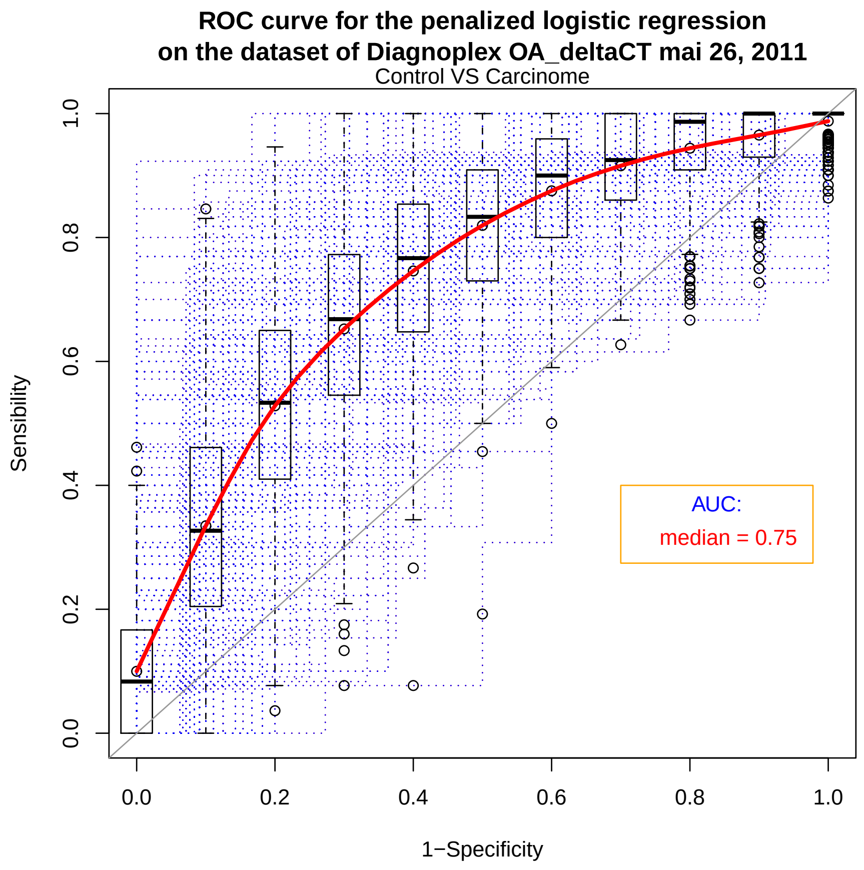

Figure 19: ROC curve of the PLR for groups C_1 and C_3. The red line represents the median for the 200 ROC curves.

0.67 which is as high as the highest AUC for the ROC curve for the k-TSP. This curve has a similar shape as the curve for the k-TSP with $M = 0.5$. The AUCs suggest that the methods k-TSP and PLR are equivalent and the TSP a little lower. Nevertheless the AUC are estimations and thus possesse a variance. The differences on the AUC are not big enough to be significant. They don't allow to claim that one method is better than another one.

There are a some genes that are used in the PLR that are also used in the TSP and the k-TSP. For example the genes 5, 26, 38 and 41. These three first genes appear in the three methods, which make them quite interesting to study. They are genes (like the gene 720 in the study of the leukemia cancer dataset) which were highlighted to be highly correlated with the disease based on the TSP and on the k-TSP, but weren't selected for the PLR. There are also a lot of genes that

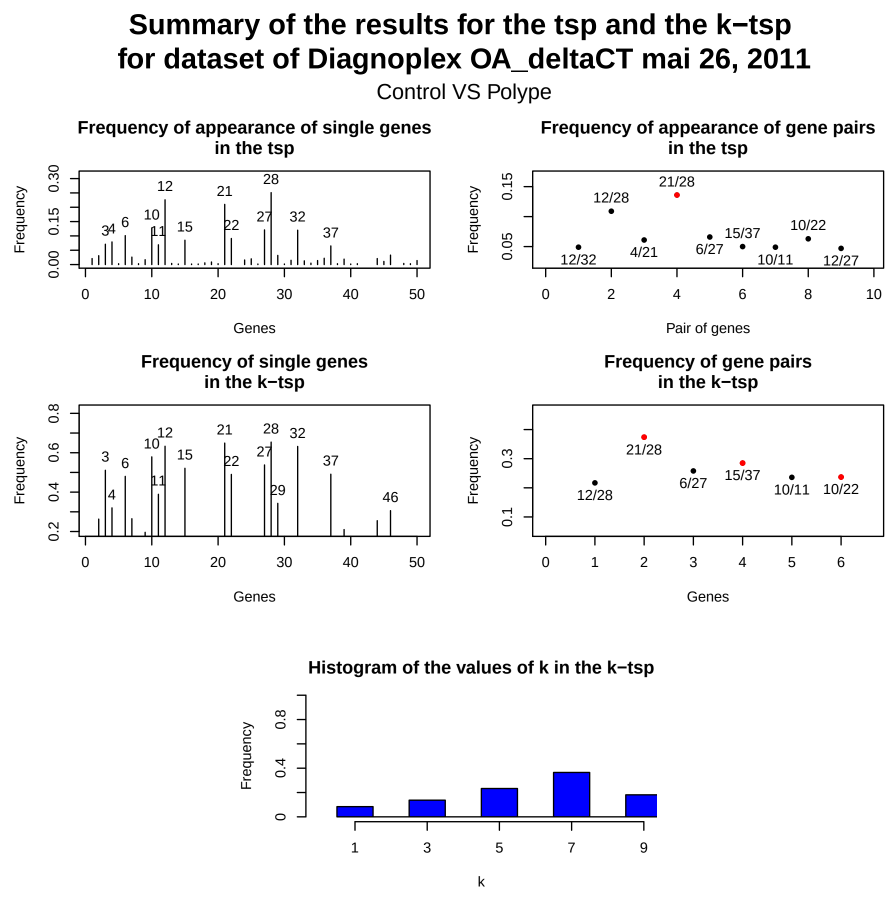

Figure 20: Summary of the results of the TSP and the k-TSP for the groups C_1 and C_2. The red points represent the results of the TSP and the k-TSP applied on the original dataset.

were selected by the PLR but not by the k-TSP. This shows that, altough the three methods are good, they may provide very different results. We briefly present the results obtained when comparing the groups C_1 with C_2 and C_1 with C_{23}.

Figure 20 shows the results obtained by the TSP and the k-TSP when comparing the groups *Control* and *Polyps*. The genes that appeared the most often on the TSP are the genes 12, 21 and 28. The gene 12 was previously detected when comparing the groups *Control* and *Cancer*. The pairs that appeared the most often along the bootstrap is the pair $(21, 28)$ which was also selected by the computation of the TSP on the original dataset. A second pair is quite near the top pair, this pair is $(12, 28)$ which also involves gene 28. For the k-TSP, there are much more single genes that appear with a high frequency. For the pairs of genes, the pair $(21, 28)$

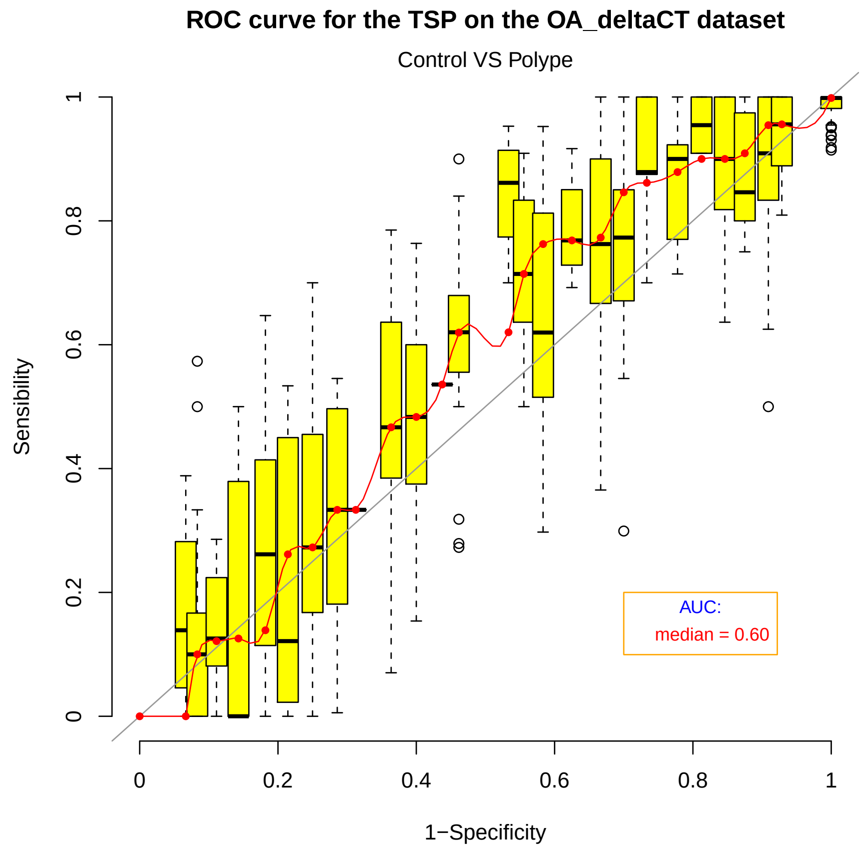

Figure 21: ROC curve of the TSP for groups C_1 and C_2. The red line represents the median for the 200 ROC curves.

appears the most often as for the TSP. There are 5 other pairs that appear also quite often, in particular the pair $(12, 28)$ that was also present on the graph of the pair's appearance for the k-TSP. In this second analysis, the TSP and the k-TSP have much more similar results than in the first analysis. Finally, the histogram of the values for k shows a marked frequency of appearance for $k = 7$. The other values of k have a lower frequency of appearance.

Figure 21 and 22 shows ROC curves for the TSP and the k-TSP based on the groups C_1 and C_2. The AUC are slighly lower than in the first analysis. Indeed, the AUC for TSP is around 0.6, the mean for the k-TSP is 0.58 and the maximimum for the k-TSP is 0.61. Again we don't see a significative difference on the AUC for the k-TSP when choosing $M = 0.25$ or $M = 0.5$. The last graph of the figure illustrate well this fact. Indeed, we saw that the red and the yellow curves are

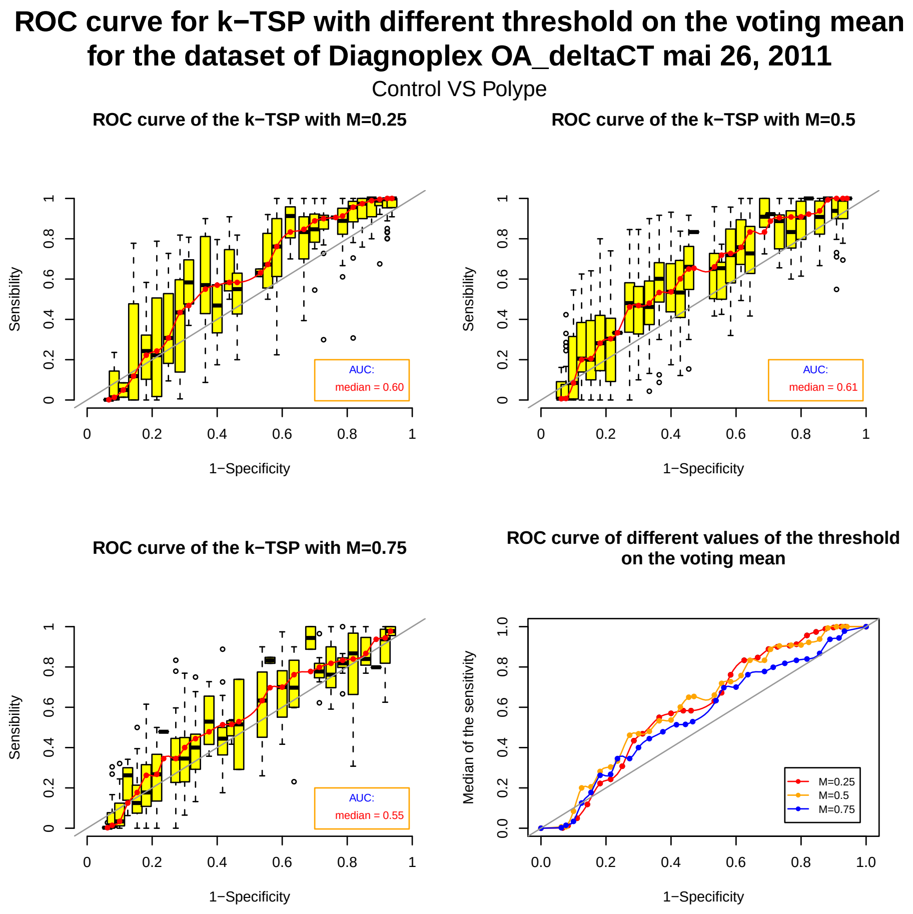

Figure 22: ROC curve of the k-TSP for groups C_1 and C_3. The red line represents the median for the 200 ROC curves.

tied together, whereas the blue one is always under. We see that, for these ROC curves, the problem that the curves go under the line of slop 1 and intercept 0 in the neighborhood of $(0, 0)$ appears also in this case. This means that there exists also pairs of genes highlighted by the k-TSP for the groups C_1 and C_2 for which the problem presented on Figure 18 happen. We note that this problem also occurs for the ROC curve of the TSP.

The results of the PLR are presented in Table 7. The three methods have only two genes in common, namely the genes 21 and 36. In this case the PLR used less genes than in the previous analysis, 14 against 23. The genes chosen with respect to each analysis using the PLR have 3 genes shared. This remark is also true for the case of the TSP and the k-TSP.

Finally, Figure 23 presents the ROC for the PLR when comparing the groups

Genes used			
2	3	7	9
10	11	21	27
32	36	37	42
46	49		

Table 7: Genes whose regression coefficients were non zero when comparing the groups C_1 and C_2.

Control and *Polyps*. The AUC is superior than for the TSP and the k-TSP. It increases fast until a specifity of 0.6 and then increases more slowly until it reaches the point $(1, 1)$.

We present now the results of the last analysis, the groups C_1 compared to the group C_{23}. Figure 24 presents the results for the TSP and the k-TSP. We can see that, as for the last case, the genes 12 and 28 are important. The pair $(12, 28)$ is the one that appeared the most often on the frequency of appearance for the TSP, this pair was ranked second in the last analysis. This pair also appears often in the k-TSP. We note that other pairs reached the same frequency. Finally the frequencies for the values of k are nearly constant for the values $k = 1, 3, 5, 7$, altough $k = 7$ is slighty higher.

On the results of the TSP, we can identify three clusters. The first one is composed of the single top ranked pair $(12, 28)$, the second one of the second best pair, the pair $(12, 32)$. Finally the third cluster is composed of 6 pairs of genes, whose frequencies of appearance are very close.

The graph of the results of the k-TSP highlights only one cluster composed of 7 pairs. We note that these pairs all appear on the graph of the frequency of appearance for the TSP. But the k-TSP selected them the same number of times over the bootstrap.

Figure 25 and 26 present the ROC curves for the TSP and the k-TSP. They have about the same values as for the previous analysis. In this case the choice of $M = 0.75$ wouln't give as bad estimations as in the previous cases. Indeed, on the last graph we see that the three curves are quite close, except for the value of specificity around 0.7, here the blue and yellow lines performed better. If we choose a specificity in the neighborhood of a specificity at least 0.7, here the curve that gives the highest sensitivity is the one with a threshold on the voting mean being $M = 0.75$. For lowest values of specificity it is better to choose $M = 0.25$.

Table 8 presents the genes used for the penalized logistic regression on the last analysis. There are much more genes used than in the previous analysis, in this

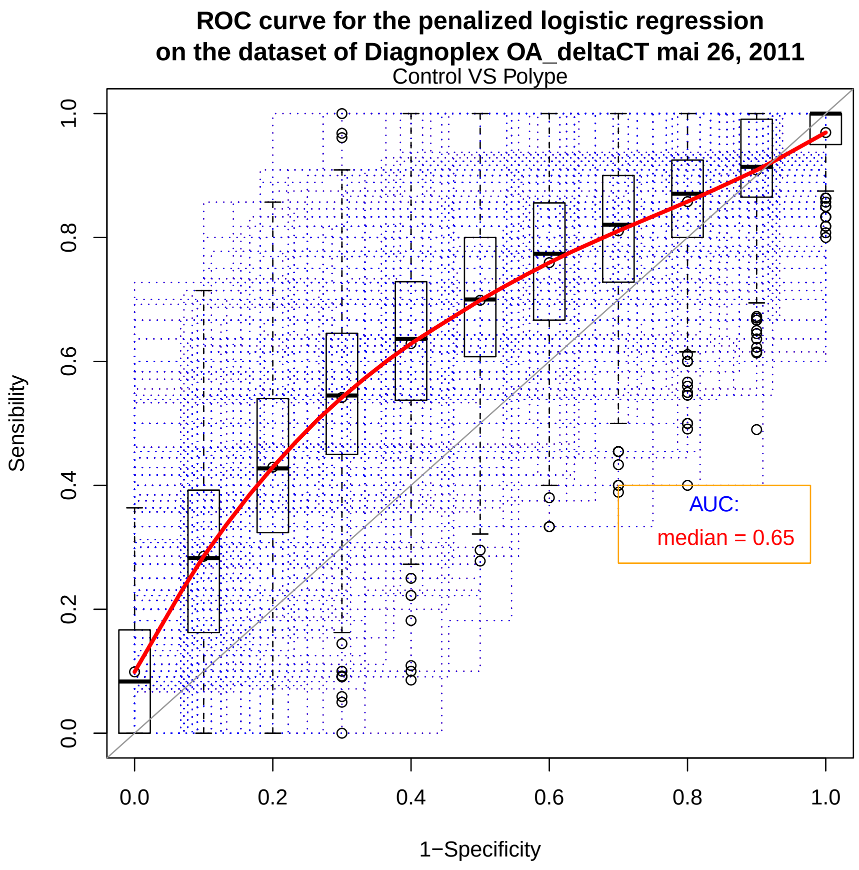

Figure 23: ROC curve of the PLR for groups C_1 and C_3. The red line represents the median for the 200 ROC curves.

case 37 genes are needed. Thus there a lot of genes in common with the previous studies as, in this case, the methods used almost 80% of the genes.

Genes used in the PLR						
1	2	3	4	7	8	9
10	11	12	13	15	16	18
19	20	21	22	23	26	27
28	30	31	32	33	36	37
38	39	41	43	45	46	47
49	50					

Table 8: Genes whose regression coefficients were non zero when comparing the groups C_1 and C_{23}.

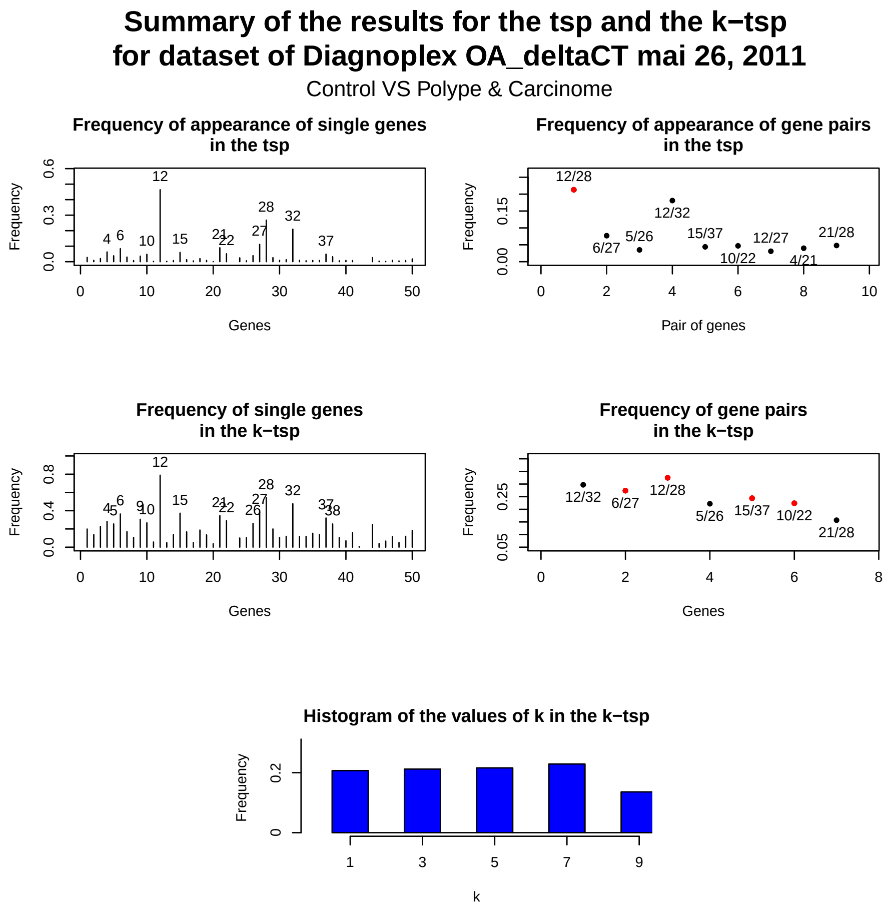

Figure 24: Summary of the results of the TSP and the k-TSP for the groups C_1 and C_{23}. The red points represent the results of the TSP and the k-TSP applied on the original dataset.

Finally, Figure 27 presents the ROC curve for the penalized logistic regression on the last analysis. The AUC reached by this ROC curve is the highest AUC obtained using PLR on the three datasets. Moreover, we observe that, when the specificity is 0, the sensiblity has a non zero value, this means that even if we classifies perfectly all healthy patients, few sick patients will be also well classified. Altough the proportion of well classified sick patients will be very low in this case. This is explained by sick patients who hae a very high probability to be well classified.

The three analysis made on this dataset give a lot of information to the genes that are correlated to the disease. Firstly, we note that the TSP and the k-TSP provided results that were quite different from the results obtained with the

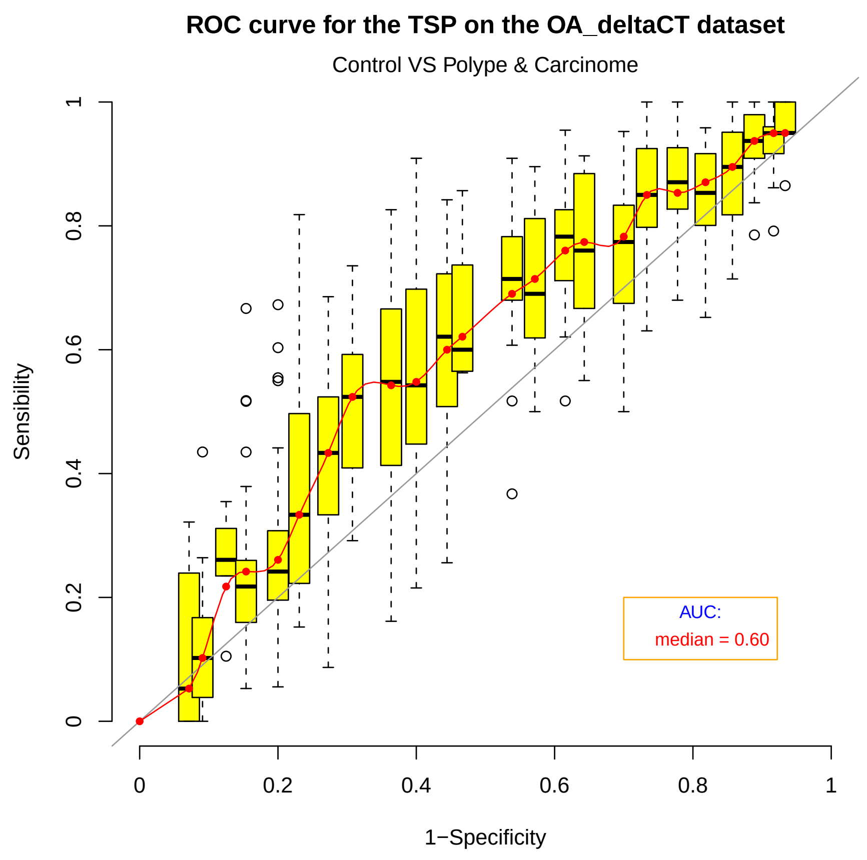

Figure 25: ROC curve of the TSP for groups C_1 and C_{23}. The red line represents the median for the 200 ROC curves.

penalized logistic regression. Indeed, this two methods are very different and use also different rules for the classification, it is then natural that they use different variables to perform their analysis.

The results found when comparing the groups C_1 and C_3 are very different. Only the gene 12 appears often for both comparisons for the TSP and the k-TSP. They also select different pairs in both situations. The k-TSP suggests to choose $k = 7$ in the second analysis, whereas in the first one it would be either 7 or 9. Based on the results of the PLR, the genes used are also different in both situations. Indeed, only 3 genes appear in the results of both analysis. These results suggest that the classifiers used to discriminate between the classes C_1 and C_3 and between the classes C_1 and C_2 are very different. The methods also tends to select more genes in the first analysis than in the second one.

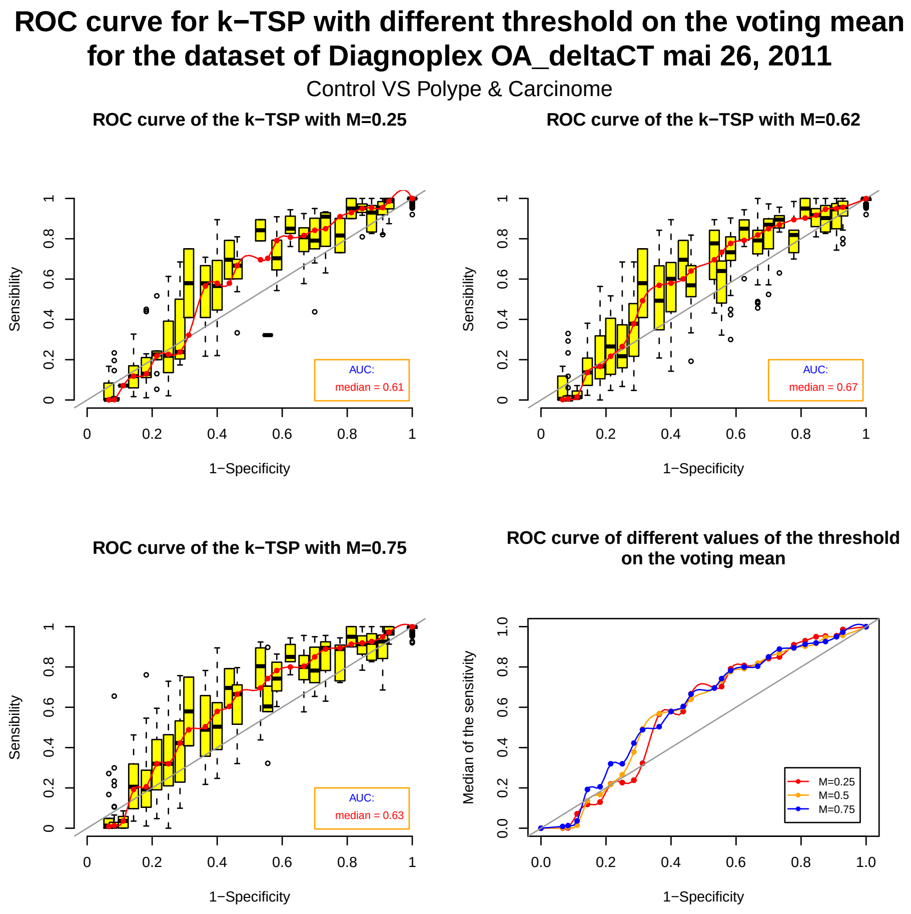

Figure 26: ROC curve of the k-TSP for groups C_1 and C_{23}. The red line represents the median for the 200 ROC curves.

Finally the analysis made between the groups C_1 and C_{23} shows similar results for the pairs of genes chosen for the TSP and the k-TSP compared to the second analysis. The difference is mainly on the number of genes one should use. Indeed, in the second analysis, this number was suggested to be around 7 for the k-TSP, whereas for the last analysis k had nearly the same frequency of appearance for $k = 1, 3, 5, 7$. Based on the TSP and the k-TSP, the similarities between the first and the third analysis are almost inexistant. Indeed, only the gene 12 appears in both results. For the pairs of genes, we can note that two pairs appear in both results, the pairs $(12, 28)$ and $(12, 32)$. Neverthelsess, the ranking of the frequency of appearance are not similar in both cases.

The penalized logistic regression also suggests few similarities between the first and the second analysis. Only the gene 11 appears in both results. The PLR

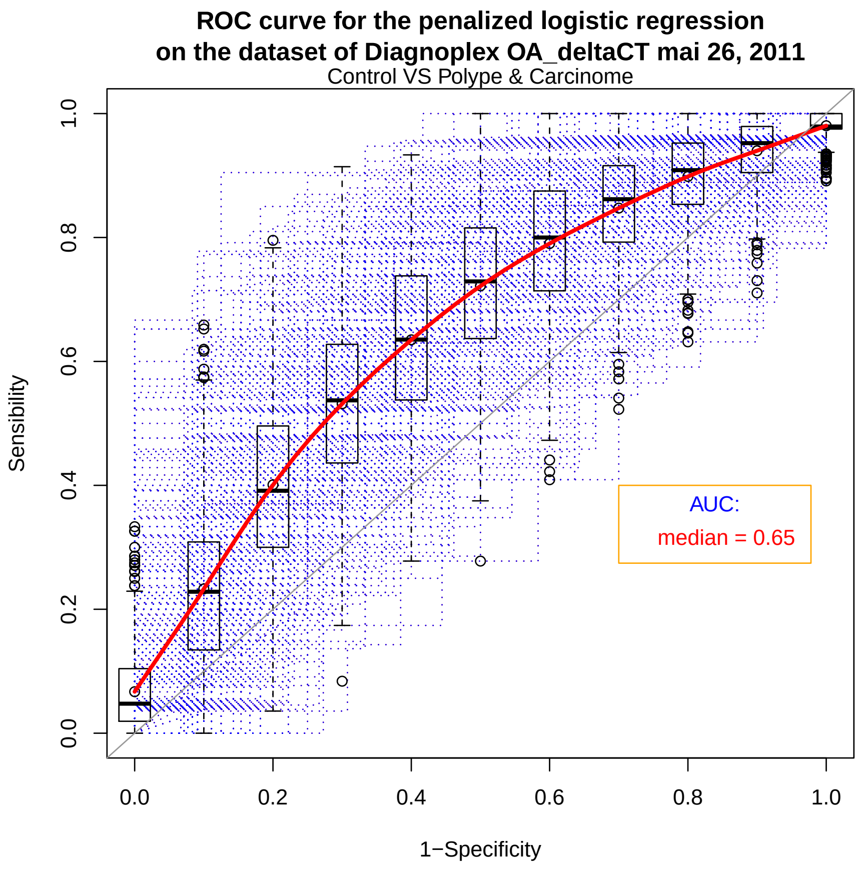

Figure 27: ROC curve of the PLR for groups C_1 and C_{23}. The red line represents the median for the 200 ROC curves.

makes stronger the suppostion drawn from the TSP and the k-TSP that the first classification should be based on more genes than the second classifications. Finally the analysis made on the comparison between the groups C_1 and C_{23} has a lot of similarities with the two other analysis. This is not suprising as it includes a big proportions of the genes present in the dataset.

One of the advantages of the TSP is that it includes only two genes, this gives an important advantage to this method. Indeed, it is very easy to interpret it and to use the results in other studies. The k-TSP also involves few genes. For this reason the k-TSP keeps the same advantages as the TSP. But, because of the fact that it considers k pairs, it allows much more flexibility and detects also other pairs of genes which are correlated with the disease. This will lead to more genes on which a study could be based on. The PLR tended to provide more genes than the two other methods. We mentioned that we used a criteria based on the accuracy precision to compute λ and the β_i's. We could also use a bound on the number of genes we allow the method to use. This would lead to a method with less genes and thus easier interpretable.

8 Conclusion

In this work we presented several methods to perform a classification analysis based on datasets composed of gene expressions. First we presented the TSP, this method is based on the relative ordering of gene expressions within two groups. It uses only two genes for the comparison and searchs for a pair that achieves the highest accuracy. We used bootstrap resample to study how this method was sensible to perturbations in the dataset. It has been showed that this sensibility was linked to the setup of the dataset, more precisely how the score Δ was distributed among the top ranked pairs of genes. We studied the robustness of the TSP through the analyse of robustness for the parameters that we used to compute the TSP. This allowed us to derive a function that was able to predict if the addition of a single observation may produces changes in the dataset. We extended this function in order that it was able to compute the lowest number of observations one would need to add such that changes occur in the results of the TSP.

Then we presented an extension of the TSP, namely the k-TSP. This method is also based on the relative ordering of the gene expressions as in the TSP, but deals with k pairs of genes instead of 1. The case where $k = 1$ is the special case of the TSP. We implemented this method for the R software. We analysed the sensibility of these methods in the same way as for the TSP. We also used bootstrap resample to analyse the behavior of the results provided by the k-TSP as the dataset is modified. We determined the value of k to minimize the error prediction rate through a crossvalidation procedure. The k-TSP, as the TSP, is sensitive to such perturbations and this sensitivity is directly linked to the properties of the dataset we analyse. It depends mainly on the distribution of the score Δ among the top ranked pairs of genes. More precisely on the difference between the lowest score over the k selected pairs and the highest score among the non selected pairs. This led to determine a new way to choose k to make the values of k less sensitive to perturbations. Actually, we choose k such that the distance mentioned is maximized. This conducted to more stable values of k over the bootstrap.

Afterwards we presented the penalized logistic regression. The aim of this method is to explain the response variables as a function of a linear combination of the explanatory variables. The genes are selected through maximizing the log likelihood of the dataset when a penalty on the coefficients is added. The penalty function can have several definition. This will give different results of the methods. In this paper we used the L^1 norm to define the penalization. This yield less non zero coefficients than the L^2 norm but with higher values.

We presented the leukemia cancer dataset at the beginning of this paper and used it to illustrate and to discuss the results provided by our method along the paper. At the end, we presented another dataset from the start up with which we collaborate to produce this paper. We performed the three methods on it and discussed the results. We saw that the TSP and the k-TSP provided similar results. Nevertheless, the k-TSP highlighted pairs of genes that weren't selected by the TSP. Indeed, the k-TSP select k pairs of genes, this allows more flexibility to the results as it will consider the k best pairs of the dataset. For this reason, the k-TSP can detect pairs of genes that are highly correlated to the disease but were never selected by the TSP as another pair would have reached a better score. The results obtained by the penalized logistic regression on these datasets were very different as the one obtained with the TSP and the k-TSP. This is, in part, explained by the fact that the method are based on different concepts. Indeed, the k-TSP compares the relative ordering of gene expressions, whereas the PLR estimates the probability of an individual to belong to one group based on a linear combination of its explanatory variables. This can be seen as a kind of linear regression, where the response variables have been transformed through a function (the link function). The PLR used different genes than the k-TSP to perform its classification. This was observed on every cases we treated. With this remark, the problem is now which method should one use to derive a classification rule. Unfortunately there doesn't exist a clear solution to this problem. Every dataset is different and every method is different. A first way to solve this would be to use only one of these methods based on the properties they possess. For example, we saw that the TSP and the k-TSP involve very few genes, that the k-TSP allows more flexibility to the pairs of genes to be selected, that the PLR provides a probability to belong to one group which gives an indication of how accurate this classification is, etc. Based on these properties, one should choose the methods with respect to the properties he wishes for his model. Another solution would be to use all the methods presented here and apply them on the individuals our dataset contains. The classification would be conduct with respect to a voting system. One can either use a voting system with equivalent weights for each of these methods or use an unweighted voting system. Again there are several solutions on the way of choosing the weights. For example, one can fix arbitrary weights with respect to the properties the methods possess, or proportionately to the accuracy reached by each of these methods. One could also give weights with respect to the robustness reached by the methods, for example, with respect to the frequency of appearance of the chosen model as the dataset is pertubated. This could conduct to a robust method.

To reach this goal another solution would be possible for the TSP and the k-TSP. Indeed, one could modify the way of selecting the genes pairs. For the TSP, one could perform a bootstrap as presented in the paper and select the pair that appears the most often as the top pair. This pair being the one that appears the most often as the dataset is perturbated, it is obvious that selecting this pair will conduct to a less sensitive method. For the k-TSP, we also use bootstrap resample, and, on each step, we perform a k-TSP, where k is chosen either by crossvalidation or to maximize the difference between the correct score Δ (as explained in section 3.3). Finally one will choose the value of k that appears the most often over the bootstrap and select the k pairs of genes that appeared also the most often.

One could also modify the presented method. When we computed the ROC curves we introduced a threshold M on the voting system, one could use this as a tunning parameter and choose it in order to maximize the accuracy or choose the values of M that would maximize the AUC, or produce the highest sensitivity (resp. specificity) for a chosen minimum specificity (sensitivity).

Annexe

We give here the code implementend to compute the k-TSP on the software R. This code is based on the code from the package [6]. We extended the C code such that it gives a list composed of all pairs of genes ordered with repect to the scores Δ and Γ with the restrictions that the genes could appear in at most one pair.

We used also the R code from [6] as a structure to the new code. This code replaces the *NA* by a parameter based on the dataset. It is linked with the C code and retains only the k first TSPs of the list, where k is given as a parameter. We extended all the functions as *plot, summary, predict, etc.* to be able to deal with several pairs of genes and to take into account the presence of *NA* values.

The file C needs to be compiled to create a .dll (under Windows) or a .so (under Linux). See *http://cran.r-project.org/doc/manuals/R-exts.pdf* for more details about this compilation.

```
#include <R.h>
#include <Rinternals.h>
#include <Rmath.h>
#include <R_ext/Utils.h>

/* This function will compute for every possible pair the scores
delta and gamma and construct an ordered list based on them.
It will select the k best pairs (the k first pairs of the list)
with the restriction that a gene can appear in at most one pair
and return them as well as their scores delta and gamma.
The list will be constitued of "blocks" of length 4.
Each block is constitued of two indices (the pair),
the score delta and the score gamma.
The NA in the original dataset have been replaced by
the value of "replacena". So when the program meet an
expression equal to this, it doesn’t treat it.
*/

SEXP cktspair(SEXP Rdat, SEXP Rgrp, SEXP Rnumberk, SEXP Rreplacena)
{
  int i,j,k,l,u,v, replace=0;
  double score, rank;
  int nProtected = 0;
```

```
 int probsum[2]; probsum[0] = 0; probsum[1] = 0;
 int probrank[2]; probrank[0] = 0; probrank[1] = 0;

 double *grp, *dat, *repna;
 grp = REAL(Rgrp);
 dat = REAL(Rdat);
 repna = REAL(Rreplacena);

 int *numberk;
 numberk = INTEGER(Rnumberk);

 int n, m;
 n = LENGTH(Rgrp);
 m = LENGTH(Rdat)/n;

 double n0, n1;
 n0 = 0;
 n1 = 0;

 SEXP list, ans;
 double *rans;

 PROTECT(ans = allocVector(REALSXP,4 * numberk[0]));
 rans = REAL(ans);
 nProtected++;

/*the variable rans will stand for the ordered
list composed of the indices, the score delta
and the score gamma */

 for (i=0; i < 4 * numberk[0]; i++){
   rans[i]=0;
 }

 for(i = 0; i < m; i++){
   for(j = 0; j < i; j++){
     for(k = 0; k < n; k++){
          if(dat[i + k * m]!= *repna && dat[j + k * m]!= *repna){
       if(grp[k] == 1){
        probsum[1] += (dat[i + k * m] < dat[j + k * m]);
        probrank[1] += (dat[i + k *m] - dat[j + k * m]);
        n1++;
        }
if(grp[k] == 0){
  probsum[0] += (dat[i + k * m] < dat[j + k * m]);
  probrank[0] += (dat[i + k *m] - dat[j + k * m]);
  n0++;
}
      }
      }

 /* Compute the score for every pairs based on the
 number of times the comparisons R(i)<R(j) were
 possible */
      score = fabs(probsum[1]/n1 - probsum[0]/n0);
      rank = fabs(probrank[1]/n1 - probrank[0]/n0);

/*Modify the ordered list with respect to the
values of the score of the current pair analysed*/

      if(score > rans[4 * (numberk[0] - 1)] ||
      (score == rans[4 * (numberk[0] - 1)] &&
      rank > rans[4 * (numberk[0] - 1) + 1])){
          rans[4 * (numberk[0] - 1)] = score;
          rans[4 * (numberk[0] - 1) + 1] = rank;
          rans[4 * (numberk[0] - 1) + 2] = j + 1;
          rans[4 * (numberk[0] - 1) + 3] = i + 1;
          }

    for(l=1; l < numberk[0]; l++){
        if(score > rans[4 * (numberk[0]-l-1)] ||
        (score == rans[4 * (numberk[0]-l-1)] &&
        rank > rans[4 * (numberk[0]-l-1) + 1])){
            rans[4 * (numberk[0]-l)] = rans[4 * (numberk[0]-1-l)];
            rans[4 * (numberk[0]-l) + 1] = rans[4 * (numberk[0]-1-l) + 1];
            rans[4 * (numberk[0]-l) + 2] = rans[4 * (numberk[0]-1-l) + 2];
            rans[4 * (numberk[0]-l) + 3] = rans[4 * (numberk[0]-1-l) + 3];
            rans[4 * (numberk[0]-1-l)] = score;
            rans[4 * (numberk[0]-1-l) + 1] = rank;
            rans[4 * (numberk[0]-1-l) + 2] = j + 1;
            rans[4 * (numberk[0]-1-l) + 3] = i + 1;

            }
    }
      probsum[0] = 0;
      probsum[1] = 0;
      probrank[0] = 0;
      probrank[1] = 0;
      n0 = 0;
      n1 = 0;
      void R_CheckUserInterrupt(void);
    }
  }
  /*Replace by 0 the blocks which uses a gene
  that was already used in a previous block
  */
    for(u=0; u < numberk[0]; u++){
        for(v=u+1; v < numberk[0]; v++){
            if(rans[4*v+2]==rans[4*u+2] ||
            rans[4*v+2]==rans[4*u+3] ||
            rans[4*v+3]==rans[4*u+2] ||
            rans[4*v+3]==rans[4*u+3]){
            rans[4*v]=0;
            rans[4*v+1]=0;
            rans[4*v+2]=0;
            rans[4*v+3]=0;
            }
        }
    }
//Delete the block constitued of 0 in the list
    for(replace=0; replace <numberk[0]; replace++){
    for(l=0; l < numberk[0]-1; l++){
        if(rans[4*l] < rans[4 * (l+1)] ||
        ( rans[4*l] < rans[4 * (l+1)] &&
        rans[4*l+1] < rans[4*(l+1)+1])){
            rans[4 * l] = rans[4 * (l+1)];
            rans[4 * l+1] = rans[4 * (l+1) + 1];
            rans[4 * l+2] = rans[4 * (l+1) + 2];
            rans[4 * l+3] = rans[4 * (l+1) + 3];
```

```
          rans[4 * (l+1)]=0;
          rans[4 * (l+1) + 1]=0;
          rans[4 * (l+1) + 2]=0;
          rans[4 * (l+1) + 3]=0;

          }
      }
    }

  PROTECT(list=allocVector(VECSXP,1));
  ++nProtected;
  SET_VECTOR_ELT(list, 0, ans);
  UNPROTECT(nProtected);
  return(list);
}
```

Now we give the code R which contains the main function as well as the extended functions *plot, summary, predict,* etc. The structure of this part is quite similar to the original code, i.e., the definition of the functions cited previously as well as the arguments they take. The implementation of these functions are more complex and are able to handle correctly the presence of *NA*.

```
dyn.load("ktsp.so") ## or dyn.load("ktsp.dll") under windows


kts.pair <- function (dat, grp, k){
## This function computes the k pairs of genes achieving the best score delta
## (use the score gamma in case of ties) using the C code
labels <- as.character(unique(grp)[order(unique(grp))])
## If the dataset contains NA, it declares by which value they have been replaced
replacena <- floor(min(dat[is.na(dat)==FALSE]))-100000
out <- .Call("cktspair", as.double(rank_na(dat,replacena)), as.double(grp),
as.integer(floor(nrow(dat)/2)), as.double(replacena))
## The vector out is a list, whose size is a multiple of 4, it contains
## all the possible pair and is ordered w.r.t. the scores delta and gamma
index <- matrix(ncol=2, nrow=k)
rankscore <- c()
```

```
ktspscore <- c()
## Select only the k best pairs
for (i in 1:k){
if((out[[1]][3 + (i-1) * 4]!=0) && (out[[1]][i * 4]!=0)){
index[i,1] <- out[[1]][3 + (i-1) * 4]
index[i,2] <- out[[1]][i * 4]
rankscore[i] <- out[[1]][2 + (i-1) * 4]
ktspscore[i] <- out[[1]][1 + (i-1) * 4]
}
}
## If less than k pairs were available, it updates k and
## the matrice containing the pairs chosen
if(is.na(index[k,1])==TRUE){
if(min(which(is.na(index[,1])==TRUE))%%2 == 1){
k <- min(which(is.na(index[,1])==TRUE))-2
}
if(min(which(is.na(index[,1])==TRUE))%%2 == 0){
k <- min(which(is.na(index[,1])==TRUE))-1
}
index <- matrix(c(index[1:k,1],index[1:k,2]),ncol=2, nrow=k)

}
## If the number of pair is an even number, it deletes
## the last pair in order to have an odd number of pair
if(k%%2==0){
k <- k-1
index <- matrix(c(index[1:k,1],index[1:k,2]),ncol=2, nrow=k)
}
ktspdat <- dat[c(index[,1],index[,2]),]
## Create an object ktsp, which contains the quantities useful
## for the use of the computed k-TSP
ktsp <- list(index = index, ktspscore = ktspscore[1:k], grp = grp,
ktspdat = ktspdat, k = k, labels =labels, rankscore = rankscore[1:k])
class(ktsp) <- "ktsp"
return(ktsp)
}

ktspcalc <- function (dat, grp, k) {
## This function prepares the dataset to be used by the function
## ktsp.pair in order to use the C code correctly
## It creates an object which contain the name of the classes
## and transfrom the variable of the group into
## a variable containing only 0 and 1 instead of names
if (class(dat) == "ExpressionSet"){
        genenames <- as.character(1:dim(exprs(dat))[1])
         if (!is.null(featureNames(dat))) {
genenames <- featureNames(dat)
        }
        if (length(grp) == 1) {
labels <- as.character(unique(pData(dat)[, grp])[order(unique(grp))])
## Transfrom the variables groups in 0 and 1
grp <- make.consecutive.int(pData(dat)[, grp])
dat <- exprs(dat)
rownames(dat) <- genenames
## Check that the number of groups doesn't exceed two
if (max(grp) != 1) {
                stop("TSPs can only be calculated for variables with two classes")
}
         ktsp <- kts.pair(dat, grp, k)
         ktsp$labels <- labels
         return(ktsp)
}
        else{
labels <- as.character(unique(grp)[order(unique(grp))])
grp <- make.consecutive.int(grp)
dat <- exprs(dat)
rownames(dat) <- genenames
if (max(grp) != 1) {
stop("TSPs can only be calculated for variables with two classes")
}
ktsp <- kts.pair(dat, grp, k)
ktsp$labels <- labels
return(ktsp)
}
}
else {
         labels <- as.character(unique(grp)[order(unique(grp))])
         grp <- make.consecutive.int(grp)
         if (is.null(rownames(dat))) {
rownames(dat) <- as.character(1:dim(dat)[1])
```

```
         }
         if (max(grp) != 1) {
stop("TSPs can only be calculated for variables with two classes")
         }
         ktsp <- kts.pair(dat, grp, k)
         ktsp$labels <- labels
}
return(ktsp)
}

## This function transfrom the variable group into
## a variable that contain only 0 and 1
make.consecutive.int <- function(y) {
oldWarn = getOption("warn")
## Turn off warnings.
options(warn = -1)
if(is.null(y)) {return(NULL)}
if(!is.vector(y))
y = as.vector(as.character(y))
out <- as.integer(as.factor(as.character(y)))-1
options(warn = oldWarn)
   return(out)
}

## This function presents the results of the k-TSP, i.e.,
## which pairs of genes are selected and their score delta

print.ktsp <- function(x){
ktspobj <- x
cat(c("k-TSP object with:",ktspobj$k, "TSPs\n"))
cat(c("Pair:\t\tTSP Score\t\tIndices\n"))
for(i in 1:length(ktspobj$ktspscore)){
cat(c("TSP",i,":","\t",round(ktspobj$ktspscore[i],2),"\t\t\t",
ktspobj$index[i,],"\n"))
}
}

## This functions plot the distributions of the genes expression
## for every pair present in the k-TSP, it is
## also possible to select only one pair by using the variable select
```

```
ktspplot <- function(ktspobj, select=NULL){
grp <- ktspobj$grp
labels <- ktspobj$labels
dat <- ktspobj$ktspdat
index <- ktspobj$index
k <- ktspobj$k

if(is.null(select)){
cat("Number of TSPs: ",k,"\n")
for(i in 1:k){
par(mar=c(4,4,4,4))
plot(dat[i,],dat[(i + k),],xlab=paste("Gene:",rownames(dat)[i],"Expression")
,ylab=paste("Gene:",rownames(dat)[(i+k)],"Expression"),type="n")
mtext(paste("Groups:",labels[1],"= Red |",labels[2],"= Blue; Score:"
,round(ktspobj$ktspscore[i],3)),line=1)
points(dat[i,grp==0],dat[(i+k),grp==0],col="red",pch=19)
points(dat[i,grp==1],dat[(i+k),grp==1],col="blue",pch=19)
abline(c(0,1),lwd=2)
readline(paste("TSP",i,": Hit return for next TSP.\n"))
}
 }

else{
par(mar=c(4,4,4,4))
plot(dat[select,],dat[(select + k),],xlab=
paste("Gene:",rownames(dat)[select],"Expression"),
ylab=paste("Gene:",rownames(dat)[(select+k)],"Expression"),type="n")
mtext(paste("Groups:",labels[1],"= Red |",labels[2],"= Blue; Score:",
round(ktspobj$ktspscore[select],3)),line=1)
points(dat[select,grp==0],dat[(select+k),grp==0],col="red",pch=19)
points(dat[select,grp==1],dat[(select+k),grp==1],col="blue",pch=19)
abline(c(0,1),lwd=2)
}
}

plot.ktsp <- function(x){ktspplot(x)}

## The function summary presents the number of patients (for each group)
## that will be classified to each class under the form of a table.
## From this table it is possible to compute the sensibility and
## the specificity of the method. It is also possible to obtain this
## table for a single pair contained in the k-TSP via the variable select.

summary.ktsp <- function(object,select=NULL){
ktspobj <- object
grp <- ktspobj$grp
k <- ktspobj$k
grplabels <- character(length(grp))
grplabels[grp==0] <- ktspobj$labels[1]
grplabels[grp==1] <- ktspobj$labels[2]
if(!is.null(select)){
cat(paste("Data for TSP:", select,"\n"))
z <- ktspobj$ktspdat[select,] < ktspobj$ktspdat[(select + k),]
print(table(z,grplabels,dnn=list(paste
("1(Gene",rownames(ktspobj$ktspdat)[select],
" < Gene", rownames(ktspobj$ktspdat)[(select + k)],")"),"Group Labels")))
}
if(is.null(select)){
cat(paste("There are",k,"TSPs\n\n"))
  for(i in 1:k){
cat(paste("Data for TSP:", i,"\n"))
z <- ktspobj$ktspdat[i,] < ktspobj$ktspdat[(i + k),]
print(table(z,grplabels,dnn=list(paste
("1(Gene",rownames(ktspobj$ktspdat)[i],
" < Gene", rownames(ktspobj$ktspdat)[(i + k)],")"),"Group Labels")))
cat("\n\n")
readline("Hit return for next TSP")
}
}
}

## The function predict is used to compute predictions. It can predict
## observations from the original dataset (the one used to compute the k-TSP)
## or predcit new observations via the variable dat. It is also possible
## to perform the prediction for a single pair in the k-TSP
## by using the variable select.

predict.ktsp <- function(object,dat=NULL,select=NULL){
ktspobj <- object
```

```
grp <- ktspobj$grp
k <- ktspobj$k
grplabels <- character(length(grp))
grplabels[grp==0] <- ktspobj$labels[1]
grplabels[grp==1] <- ktspobj$labels[2]
predict <- c()



if(is.null(dat) & !is.null(select)){## compute the prediction for a single pair
z1 <- ktspobj$ktspdat[select,] < ktspobj$ktspdat[(select + k),]
## Create the table used to predict individuals based on the
## frequency of appearance of R(i)<R(j) in both groups.
table_label <- dimnames(table(z1,grplabels))[[2]]
if(sum(z1)!=0 && sum(z1)!=length(z1)){
predict[which(z1 == 0)] <- table_label[which.max(table(z1,grplabels)[1,])]
predict[which(z1 == 1)] <- table_label[which.max(table(z1,grplabels)[2,])]
}
z2 <- ktspobj$ktspdat[select,] > ktspobj$ktspdat[(select + k),]
table_label <- dimnames(table(z2,grplabels))[[2]]
if((sum(z1)==0 || sum(z1)==length(z1)) && (sum(z2)!=0 && sum(z2)!=length(z2))){
predict[which(z2 == 0)] <- table_label[which.max(table(z2,grplabels)[1,])]
predict[which(z2 == 1)] <- table_label[which.max(table(z2,grplabels)[2,])]
}
return(predict)
}
if(is.null(dat) & is.null(select)){
predict <- character(length(grp))
vote <- numeric(length(grp))
vote2 <-matrix(nrow=length(grp), ncol=k)
count <- numeric(length(grp))
for(i in 1:k){
## For each pair it creates the table used for the prediction and
## use it to predict the individuals. It makes an count for every
## pair present in the k-TSP.
z1 <- ktspobj$ktspdat[i,] < ktspobj$ktspdat[(i + k),]
if(sum(z1)!=0 && sum(z1)!=length(z1)){
vote2[which(z1 == 0),i] <- which.max(table(z1,grplabels)[1,]) - 1
vote2[which(z1 == 1),i] <- which.max(table(z1,grplabels)[2,]) - 1
count[is.na(z1)==FALSE] <- count[is.na(z1)==FALSE]+1
}
z2 <- ktspobj$ktspdat[i,] > ktspobj$ktspdat[(i + k),]
if((sum(z1)==0 || sum(z1)==length(z1)) && (sum(z2)!=0 && sum(z2)!=length(z2))){
vote2[which(z2 == 0),i] <- which.max(table(z2,grplabels)[1,]) - 1
vote2[which(z2 == 1),i] <- which.max(table(z2,grplabels)[2,]) - 1
count[is.na(z2)==FALSE] <- count[is.na(z2)==FALSE]+1
}
}
table_label <- labels(table(z1,grplabels))[2]
vote <- apply(vote2, 1, sum)
## Predict an individual to the class that obtained the largest prediction
## in the previous step.
predict[(vote/count < 1/2)] <- table_label$grplabels[1]
predict[(vote/count > 1/2)] <- table_label$grplabels[2]
nopred <- which((predict=="")==TRUE)
if(length(nopred)>0){##
##Consider individual that couldn't be predict (because of NA)
predict2 <- character(length(nopred))
cat("For the observation(s): ", nopred, " a part of the data is missing \n")
cat("Their prediction was computed on a subset of the k-TSP\n")
for(i in 1:length(nopred)){## Consider only prediction that were possible
   ## and take the mean.
vote3 <- numeric(length(nopred))
vote3 <- vote2[i,is.na(vote2[i,])==FALSE]
pred <- mean(vote3)
## Classfie an individual to the class that obtained the largest vote
if(pred < 1/2){predict2[i] <- table_label$grplabels[1]}
if(pred > 1/2){predict2[i] <- table_label$grplabels[2]}
else{
pred <- mean(vote3[-length(vote3)])
if(pred < 1/2){predict2[i] <- table_label$grplabels[1]}
if(pred > 1/2){predict2[i] <- table_label$grplabels[2]}
}
predict[nopred] <- predict2
}
}
return(predict)
}
## Same as before but in this case we predict new individuals
## (represented by the matrix dat)
```

```
if(!is.null(dat) & !is.null(select)){
ktspnames <- rownames(ktspobj$ktspdat)[c(select,(select + k))]
predict <- character(dim(dat)[2])
vote <- numeric(length(grp))
vote2 <-matrix(nrow=length(grp), ncol=k)
count <- numeric(length(grp))
if(class(dat) == "matrix"){
if(is.null(rownames(dat))){
cat("No rownames found, using indices \n")
ktspnames <- as.numeric(ktspnames)
if(any(!(ktspnames %in% 1:dim(dat)[1]))){
  stop("Rownames of new data do not include the TSP names")}
}
else{
if(any(!(ktspnames %in% rownames(dat)))){
stop("Rownames of new data do not include the TSP names")}
}

z1 <- ktspobj$ktspdat[select,] < ktspobj$ktspdat[(select + k),]
table_label <- as.vector(dimnames(table(z1,grplabels))[[2]])
if(sum(z1)!=0 && sum(z1)!=length(z1)){
w <- dat[ktspnames[1],] < dat[ktspnames[2],]
predict[which(w == 0)] <- table_label[which.max(table(z1,grplabels)[1,])]
predict[which(w == 1)] <- table_label[which.max(table(z1,grplabels)[2,])]
}
z2 <- ktspobj$ktspdat[select,] > ktspobj$ktspdat[(select + k),]
table_label <- dimnames(table(z2,grplabels))[[2]]
if((sum(z1)==0 || sum(z1)==length(z1)) &&
(sum(z2)!=0 && sum(z2)!=length(z2))){
w <- dat[ktspnames[1],] > dat[ktspnames[2],]
predict[which(w == 0)] <- table_label[which.max(table(z2,grplabels)[1,])]
predict[which(w == 1)] <- table_label[which.max(table(z2,grplabels)[2,])]
}
nopred <- which((predict=="")==TRUE)
if(length(nopred)>0){
cat("For the observation(s): ", nopred,
" a part of the data is missing \n")
cat("Their prediction for the selected pair is not possible\n")
}
return(predict)
}
if(class(dat) == "ExpressionSet"){
if(is.null(featureNames(dat))){
cat("No featureNames info found, using indices \n")
ktspnames <- as.numeric(ktspnames)
         if(any(!(ktspnames %in% 1:dim(exprs(dat))[1]))){
         stop("Rownames of new data do not include the TSP names")}
dat <- exprs(dat)
}
else{
if(any(!(ktspnames %in% featureNames(dat)))){
stop("Rownames of new data do not include the TSP names")}
        genenames <- featureNames(dat)
        dat <- exprs(dat)
        rownames(dat) <- genenames
}
z1 <- ktspobj$ktspdat[select,] < ktspobj$ktspdat[(select + k),]
table_label <- dimnames(table(z1,grplabels))[[2]]
if(sum(z1)!=0 && sum(z1)!=length(z1)){
w <- dat[ktspnames[i],] < dat[ktspnames[i+k],]
predict[which(w == 0)] <- table_label[which.max(table(z1,grplabels)[1,])]
predict[which(w == 1)] <- table_label[which.max(table(z1,grplabels)[2,])]
}
z2 <- ktspobj$ktspdat[select,] > ktspobj$ktspdat[(select + k),]
table_label <- dimnames(table(z2,grplabels))[[2]]
if((sum(z1)==0 || sum(z1)==length(z1)) && (sum(z2)!=0 && sum(z2)!=length(z2))){
w <- dat[ktspnames[i],] > dat[ktspnames[i+k],]
predict[which(w == 0)] <- table_label[which.max(table(z2,grplabels)[1,])]
predict[which(w == 1)] <- table_label[which.max(table(z2,grplabels)[2,])]
}
nopred <- which((predict=="")==TRUE)
if(length(nopred)>0){
cat("For the observation(s): ", nopred, " a part of the data is missing \n")
cat("Their prediction for the selected pair is not possible\n")
}
return(predict)
}
}

else{
```

```
ktspnames <- rownames(ktspobj$ktspdat)
if(class(dat) == "matrix"){
if(is.null(rownames(dat))){
cat("No rownames found, using indices \n")
ktspnames <- as.numeric(ktspnames)
if(any(!(ktspnames %in% 1:dim(dat)[1]))){
stop("Rownames of new data do not include the TSP names")}
}
else{
if(any(!(ktspnames %in% rownames(dat)))){
stop("Rownames of new data do not include the TSP names")}
}
predict <- character(dim(dat)[2])
vote <- numeric(dim(dat)[2])
vote2 <-matrix(nrow=dim(dat)[2], ncol=k)
count <- numeric(dim(dat)[2])
for(i in 1:k){
z1 <- ktspobj$ktspdat[i,] < ktspobj$ktspdat[(i + k),]
if(sum(z1)!=0 && sum(z1)!=length(z1)){
w <- dat[ktspnames[i],] < dat[ktspnames[i+k],]
vote2[which(w == 0),i] <- which.max(table(z1,grplabels)[1,]) - 1
vote2[which(w == 1),i] <- which.max(table(z1,grplabels)[2,]) - 1
count[is.na(w)==FALSE] <- count[is.na(w)==FALSE]+1
}
z2 <- ktspobj$ktspdat[i,] > ktspobj$ktspdat[(i + k),]
if((sum(z1)==0 || sum(z1)==length(z1)) &&
 (sum(z2)!=0 && sum(z2)!=length(z2))){
w <- dat[ktspnames[i],] > dat[ktspnames[i+k],]
vote2[which(w == 0),i] <- which.max(table(z2,grplabels)[1,]) - 1
vote2[which(w == 1),i] <- which.max(table(z2,grplabels)[2,]) - 1
count[is.na(w)==FALSE] <- count[is.na(w)==FALSE]+1
}
}
table_label <- labels(table(z1,grplabels))[2]
vote <- apply(vote2, 1, sum)
predict[(vote/count < 1/2)] <- table_label$grplabels[1]
predict[(vote/count > 1/2)] <- table_label$grplabels[2]
nopred <- which((predict=="")==TRUE)
if(length(nopred)>0){
predict2 <- character(length(nopred))
```

```
cat("For the observation(s): ", nopred,
" a part of the data is missing \n")
cat("Their prediction was computed on a subset of the k-TSP\n")
for(i in 1:length(nopred)){
vote3 <- numeric(length(nopred))
vote3 <- vote2[i,is.na(vote2[i,])==FALSE]
if(length(vote3)<1){
cat("Prediction not possible for observation", nopred[i], " \n")}
else{
pred <- mean(vote3)
if(pred < 1/2){predict2[i] <- table_label$grplabels[1]}
if(pred > 1/2){predict2[i] <- table_label$grplabels[2]}
else{
if(length(vote3)<2){
cat("Prediction not possible for observation", nopred[i], "\n")}
else{
pred <- mean(vote3[-length(vote3)])
if(pred < 1/2){predict2[i] <- table_label$grplabels[1]}
if(pred > 1/2){predict2[i] <- table_label$grplabels[2]}
}
}
}
}
predict[nopred] <- predict2
}
return(predict)
}
if(class(dat) == "ExpressionSet"){
if(is.null(featureNames(dat))){
cat("No featureNames info found, using indices \n")
ktspnames <- as.numeric(ktspnames)
if(any(!(ktspnames %in% 1:dim(exprs(dat))[1]))){
stop("Rownames of new data do not include the TSP names")}
dat <- exprs(dat)
}
else{
if(any(!(ktspnames %in% featureNames(dat)))){
stop("Rownames of new data do not include the TSP names")}
genenames <- featureNames(dat)
dat <- exprs(dat)
```

```
rownames(dat) <- genenames
}
predict <- character(dim(dat)[2])
vote <- numeric(dim(dat)[2])
vote2 <-matrix(nrow=dim(dat)[2], ncol=k)
count <- numeric(dim(dat)[2])
for(i in 1:k){
z1 <- ktspobj$ktspdat[i,] < ktspobj$ktspdat[(i + k),]
if(sum(z1)!=0 && sum(z1)!=length(z1)){
w <- dat[ktspnames[i],] < dat[ktspnames[i+k],]
vote2[which(w == 0),i] <- which.max(table(z1,grplabels)[1,]) - 1
vote2[which(w == 1),i] <- which.max(table(z1,grplabels)[2,]) - 1
count[is.na(w)==FALSE] <- count[is.na(w)==FALSE]+1
}
z2 <- ktspobj$ktspdat[i,] > ktspobj$ktspdat[(i + k),]
if((sum(z1)==0 || sum(z1)==length(z1)) &&
(sum(z2)!=0 && sum(z2)!=length(z2))){
w <- dat[ktspnames[i],] > dat[ktspnames[i+k],]
vote2[which(w == 0),i] <- which.max(table(z2,grplabels)[1,]) - 1
vote2[which(w == 1),i] <- which.max(table(z2,grplabels)[2,]) - 1
count[is.na(w)==FALSE] <- count[is.na(w)==FALSE]+1
}
}
table_label <- labels(table(z1,grplabels))[2]
vote <- apply(vote2, 1, sum)
predict[(vote/count < 1/2)] <- table_label$grplabels[1]
predict[(vote/count > 1/2)] <- table_label$grplabels[2]
nopred <- which((predict=="")==TRUE)
if(length(nopred)>0){
predict2 <- character(length(nopred))
cat("For the observation(s): ", nopred,
" a part of the data is missing \n")
cat("Their prediction was computed on a subset of the k-TSP\n")
for(i in 1:length(nopred)){
vote3 <- numeric(length(nopred))
vote3 <- vote2[i,is.na(vote2[i,])==FALSE]
if(length(vote3)<1){
cat("Prediction not possible for observation", nopred[i], " \n")}
else{
pred <- mean(vote3)
if(pred < 1/2){predict2[i] <- table_label$grplabels[1]}
if(pred > 1/2){predict2[i] <- table_label$grplabels[2]}
else{
if(length(vote3)<2){
cat("Prediction not possible for observation", nopred[i], "\n")}
else{
pred <- mean(vote3[-length(vote3)])
if(pred < 1/2){predict2[i] <- table_label$grplabels[1]}
if(pred > 1/2){predict2[i] <- table_label$grplabels[2]}
}
}
}
}
predict[nopred] <- predict2
}
return(predict)
}
}
}

## This function replaces the NA in the dataset by the variable na,
## by default na is -100000

rank_na <- function (dat, na=-100000){
n <- nrow(dat)
m <- ncol(dat)
for(i in 1:m){
dat[,i][is.na(dat[,i])==FALSE]<- rank(dat[,i][is.na(dat[,i])==FALSE])
}
dat[which(is.na(dat)==TRUE)] <- na
dat <- matrix(dat,n,m)
return(dat)
}
```

References

[1] Sejong Yoon and Saejoon Kim. k-top scoring pair algorithm for feature selection in svm with applications to microarray data classification. *Soft Comput.*, 14:151–159, September 2009.

[2] Aik Choon Tan, Daniel Q. Naiman, Lei Xu, Raimond L. Winslow, and Donald Geman. Simple decision rules for classifying human cancers from gene expression profiles. *Bioinformatics*, 21:3896–3904, October 2005.

[3] Lei Xu, Aik Choon Tan, Daniel Q. Naiman, Donald Geman, and Raimond L. Winslow. Robust prostate cancer marker genes emerge from direct integration of inter-study microarray data. *Bioinformatics*, 21(20):3905–3911.

[4] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, and C. D. Bloomfield. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[5] Ricardo A. Maronna, R. Douglas Martin, and Victor J. Yohai. *Robust statistics*. Wiley Series in Probability and Statistics. John Wiley & Sons Ltd., Chichester, 2006. Theory and methods.

[6] Jeffrey T. Leek. *tspair: Top Scoring Pairs for Microarray Classification*. R package version 1.10.0.

[7] Sahar Hosseinian. *Robust inference for generalizaed linear models: binary and poisson regression*. PhD thesis, EPFL, 2009.

[8] Mee Young Y. Park and Trevor Hastie. Penalized logistic regression for detecting gene interactions. *Biostatistics (Oxford, England)*, 9(1):30–50, January 2008.

[9] Robert Tibshirani (2010) Jerome Friedman, Trevor Hastie. *Regularization Paths for Generalized Linear Models via Coordinate Descent*. R package version 1.10.0.

[10] Niko Beerenwinkel Tobias Sing, Oliver Sander and Thomas Lengauer. Rocr: visualizing classifier performance in r. *Bioinformatics*, 21:3940–3941, 2005.

[11] Donald Geman, Christian d'Avignon, Daniel Q. Naiman, and Raimond L. Winslow. Classifying gene expression profiles from pairwise mRNA comparisons. *Stat. Appl. Genet. Mol. Biol.*, 3:Art. 19, 21 pp. (electronic), 2004.

[12] Marcin Czajkowski and Marek Krętowski. Novel extension of k - tsp algorithm for microarray classification. In *Proceedings of the 21st international*

conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems: New Frontiers in Applied Artificial Intelligence, IEA/AIE '08, pages 456–465, Berlin, Heidelberg, 2008. Springer-Verlag.

[13] Robert Gentleman, Vincent J. Carey, Wolfgang Huber, Rafael A. Irizarry, and Sandrine Dudoit, editors. *Bioinformatics and computational biology solutions using R and bioconductor*. Statistics for Biology and Health. Springer, New York, 2005.