Neighborhood-Oriented Feature Selection and Classification of Duke's Stages on Colorectal Cancer Using High Density Genomic Data

by

LIANG PENG

B.S., Emporia State University, 2008

A REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Statistics

College of Arts and Sciences

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2011

Approved by:

Major Professor

Haiyan Wang

Copyright

Abstract

The selection of relevant genes for classification of phenotypes for diseases with gene expression data have been extensively studied. Previously, most relevant gene selection was conducted on individual gene with limited sample size. Modern technology makes it possible to obtain microarray data with higher resolution of the chromosomes. Considering gene sets on an entire block of a chromosome rather than individual gene could help to reveal important connection of relevant genes with the disease phenotypes. In this report, we consider feature selection and classification while taking into account of the spatial location of probe sets in classification of Duke's stages B and C using DNA copy number data or gene expression data from colorectal cancers. A novel method was presented for feature selection in this report. A chromosome was first partitioned into blocks after the probe sets were aligned along their chromosome locations. Then a test of interaction between Duke's stage and probe sets was conducted on each block of probe sets to select significant blocks. For each significant block, a new multiple comparison procedure was carried out to identify truly relevant probe sets while preserving the neighborhood location information of the probe sets. Support Vector Machine (SVM) and K-Nearest Neighbor (KNN) classification using the selected final probe sets was conducted for all samples. Leave-One-Out Cross Validation (LOOCV) estimate of accuracy is reported as an evaluation of selected features. We applied the method on two large data sets, each containing more than 50,000 features. Excellent classification accuracy was achieved by the proposed procedure along with SVM or KNN for both data sets even though classification of prognosis stages (Duke's stages B and C) is much more difficult than that for the normal or tumor types.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Diagnosis and prognosis of cancer subtypes using genomic information has been an active area of research in recent years. In this report, we focus on colorectal cancer subtypes. Current colorectal cancer prognosis is mainly through pathological staging, which provides limited discrimination for Dukes stages B and C disease due to highly heterogeneous genetic content of colorectal carcinoma. The heterogeneity is a result of multiple mechanisms, including the accumulation of genetic alterations, such as chromosomal instability, gene mutations, and epigenetic abnormality after initiated by inactivation of the adenomatous polyposis coli (APC) tumour-suppressor pathway in a cell within the colon (Markowitz and Bertagnolli 2009; Rajagopalan et al. 2003). There is also experimental evidence that chromosomal instability (CIN) may precede mutation of APC. No matter which occurs first, it is concluded that chromosomal abnormalities occur at an early stage of colorectal neoplasia (Bomme et al. 1998; Hermsen et al. 2002; Pihan et al. 2003). Continuing accrual of genetic changes from the occurrence of the APC mutation to the development of a metastatic cancer generally takes about 20 to 40 years (Rajagopalan et al. 2003). Consequently, tumors are very heterogeneous even when comparing within the same histopathological stage with clonal selections (Alon et al. 1999; Beroukhim et al. 2010; Rajagopalan et al. 2003). These are shown in global gene expression and DNA copy number variation studies obtained from colon cancer patients. Pathogenic alterations can be more heterogeneous in adenomas than in carcinomas. Approximately 15% of colorectal cancers show microsatellite instability and the remaining 85% are aneuploid because of CIN (Rajagopalan et al. 2003).

In both cases, an accelerated rate of gains or losses of whole or large portions of chromosomes allow cells to rapidly acquire genetic advantage for tumorigenesis leading to experimentally verified evidence that most late stage cancer cells contain between 60 and 90 chromosomes. Therefore, we believe that the chromosome copy number alteration might be a good indicator to differentiate the various of stages the cancer cells belong to. However, we have not found any successful application of using copy number alteration to differentiate tumor stages.

Statistical and computational techniques, such as clustering or classification modified from classical setting to suit the current high density setting, were applied to this area to help with the diagnosis and prognosis of various types of diseases using genomic data. For example, binary tree based clustering could separate cancerous from noncancerous tissue and cell lines from in vivo tissues by using patterns of genes expression from oligonucleotide cDNA arrays (Alon et al. 1999). Nearest shrunken centroid classifier was found to be highly efficient in finding genes for classifying small round blue cell tumors and leukemias (Tibshirani et al. 2002). The same nearest shrunken centroid algorithm trained with Dukes stage A and C colorectal cancer can predict poor prognosis outcomes in patients with Dukes stage B and C colorectal cancer (Jorissen et al. 2009). A set of manually selected 34 genes were used in hierarchical clustering based on Pearsons correlation coefficient to classify patients with recurrence and death (Smith et al. 2010). Even though the classifiers build in Jorissen et al. (2009) and Smith et al. (2010) showed little overlap with previously reported prognosis signatures, they provide additional markers for outcome prediction.

Beyond those references cited above on Dukes stages, Recent studies provided various methods to identify a set of marker genes and use these marker genes to classify cancer types or assess the progression of a specific cancer with microarray data. It has been widely recognized that the number of selected features and features themselves are very critical to the classification accuracy (Dagliyan et al. 2011; Zhang and Deng 2007). Therefore, recent tumor classification methods mostly consist of feature selection step and classification step (Wei and Li 2010; Dagliyan et al. 2011; Wang et al. 2010; Zhang et al. 2011), with some methods including an additional filtering step to remove redundant features (Zhang and Deng 2007). One common aspect for selecting features in these methods is as follows: a certain measurement, such as some test statistics or p-values, was chosen

to evaluate the importance of each gene for classifying the phenotypes. Then a set of genes with often an arbitrarily determined size were selected based on the rankings of this measurement. Alternatively, some methods used False Discovery Rate (FDR) control to select features. Classification accuracy from cross validation is typically carried out after the feature selection step using the entire sample. These methods were reported to perform very well in a few widely used microarray datasets with a few thousand genes ranging from 2000 to 12600. The feature selection based on multiple comparison adjustment may fail when the number of features increases dramatically with the technology advances as is the case in high density arrays.

In this study, we will use mRNA expression data and DNA copy number data from colorectal tumors to identify cancer subtypes and biomarkers that are instructive for personalized medicine and predicative of clinical outcomes. Copy number profile consists of copy numbers of singular nucleotide polymorphism (SNP) sites ordered by their physical position on the genome. We will use publicly available Affymetrix SNP array data to obtain the copy numbers of SNPs. For data collected from these arrays, there are a large number of probes (up to 1M probes on each chip) and the observed intensities of some of the probes are correlated. Nearby probes have more chance of sharing common copy numbers leading to a special form of spatial correlation. Most algorithms ignore the unknown correlation structures among observations for different genetic markers from the same biological sample. Appropriately taking into account such correlations can significantly increase the classification accuracy and stability.

In this study we propose a novel method to conduct feature selection of genetic markers and use those selected markers to classify the phenotypes of Duke's stages. This method is particularly suited to high density genomic data which contains rich information from neighborhood yet is challenging for most available methods that focus on feature selection through filtering of individual genetic marker. Instead of traditional methods that evaluate individual genetic marker, our proposed method operates the feature selection based on the blocks of contiguous markers. The selected markers can be utilized along with Support Vector Machine (SVM) or K-Nearest Neighbor (KNN) to perform classification.

The rest of this report is organized as follows. Chapter 2 reviews some of the key

references related to this field of study. Chapter 3 presents the details of the proposed method and the rationale behind it. Chapter 4 is devoted to the application to two publicly available datasets using the proposed method. The programming code in Java and R is listed in the Appendices in the order of data analysis steps.

Chapter 2

Literature Review

The selection of genes for classification of phenotypes for diseases using microarray data has been a active research topic in recent years. Various methods have been proposed and discussed targeting at improving the accuracy of classification while maintaining relative small a number of genes. The following is a literature review of six articles in this field.

2.1 Gene selection for classification of microarray data based on the Bayes error

A research article by Zhang and Deng (2007) presented a problem of gene selection and classification of microarray data using Bayes error. The article first pointed out that several widely used methods, which ranked individual genes based on their discriminative power, resulted in a large number of candidate genes including some unnecessary ones due to redundancy. Then it stated that more recent studies showed using correlation analysis could help reduce the number of genes and increase the accuracy.

The gene selection process of Zhang and Deng (2007) contains two steps: gene preselection and redundancy filter. In the gene preselection step, discriminative power of each gene was measured by an univariate criterion function. Specifically, Wilcoxon test was used here to select the genes based on Family-Wise-Error Rate(FWER) ≤ 0.05 from all genes. In

the redundancy filter step, among the remaining genes, Bayes error, whose upper bound was estimated by Bhattacharyya distance, was used as a criterion to filter out redundant genes. The Bhattacharyya distance, d_B measures the separability between two classes and is defined as:

$$d_B = \frac{1}{8}(M_2 - M_1)^T \left[\frac{\sum_1 + \sum_2}{2}\right]^{-1} (M_2 - M_1) + \frac{1}{2} ln \frac{|(\sum_1 + \sum_2)/2|}{|\sum_1|^{1/2} |\sum_2|^{1/2}}$$

where M_k.... is the mean vector of class k ($k = 1$ or 2); $\sum_k$ is the covariance matrix of class $k(k = 1$ or 2). Then the upper bound of the Bayes error can be derived as

$$\varepsilon_B^* \leq 0.5 \exp(-d_B)$$

It is stated in Zhang and Deng (2007) that ϵ^* monotonically increases in a decelerating manner when d_B increases and the effect of improvement for accuracy becomes negligible after d_B increases to a certain level. They therefore argue that as the number of genes increases, after a certain threshold, the contribution caused by addition of more genes become negligible. Following this principle, ϵ^* being equal to 1.0E-4 was set up as a criterion to control the Bayes error during the classification process to filter out the redundant genes. The following is how the algorithm works: after the gene preselection step, a set B that contains all the remaining genes and an empty set A are constructed. First, the gene ranks first based on Wilcoxon test in set B is picked and moved to set A. This gene is considered as indispensable in A. Second, 1.0E-4 is used as the pre-defined criterion and sequential forward selection is applied to select the genes with great contribution to Bhattacharyya distance between two classes. In other words, sequential forward selection is used to transfer genes from set B to set A until ϵ^* is less than 1.0E-4.

For the classification process, K-Nearest Neighbor(KNN) (k=5, using Euclidean distance) and Support Vector Machine (SVM)(linear kernel) were used as two classifiers. Leave-One-Out Cross Validation (LOOCV)was used to evaluate the performance of the proposed method.

The experiment using the proposed method was conducted on five publicly available datasets, i.e., Colon, DLBCL, Leukemia, Prostate and Lymphoma by Alon et al. (1999), Shipp et al.

(2002), Golub et al. (1999), Singh et al. (2002) and Davis et al. (2000), respectively. These five different datasets have different numbers of genes ranging from 2000 to 4026, different total sample sizes ranging from 62 to 102. However, each of datasets has binary class label. The error which is calculated as the misclassified rate in test data was recorded as the results.

The results were compared with several other methods in early studies for each of the datasets. For the Colon dataset, Ben-Dor et al. (2000) also did an experiment using KNN and SVM without selecting genes first. The errors using Based Bayes error Filter (BBF) were much smaller than those Ben-Dor et al. (2000). In addition, Liu et al. (2005) used "normalized mutual information" with greedy algorithm and simulated annealing algorithm for gene selection and KNN for classification, got the equal errors as BBF, but used more genes. Ding and Peng (2005) used a "Minimum Redundancy - Maximum Relevancy" (MRMR) method for gene selection and SVM for classification got slightly smaller error than BBF with a more genes. For DLBCL dataset, Shipp et al. (2002) who used this dataset originally, used their own weighted combination of informative genes employing KNN and SVM, got same errors as BBF with a much larger number of genes. In addition, Yang et al. (2006) used GS1 and GS2 methods based on the ratio of inter-class and intra-class variation as a criterion function for gene selection, and used KNN and SVM for classification. They produced slightly smaller error than BBF but with much more genes. For Leukemia dataset, the BBF method got all test sample correctly classified using both KNN and SVM with only 3 and 2 genes, respectively. Dettling and Buhlmann (2003) used four classifiers, i.e.,Logit-Boost, AdaBoost, KNN and Classification tree and LOOCV for evaluation. They got error of 1.39% using KNN. Weston et al. (2000) got all samples correctly classified by SVM but used 20 genes. For the Prostate dataset, Dettling and Buhlmann (2003) used supervised clustering method to find gene groups for classification. With KNN and aggregated trees classifiers, they got slightly better accuracy than BFF using more genes. For the lymphoma dataset, Wang et al. (2006) used several methods for gene selection and classification. The smallest error rate was produced by Information-Gain and Neuro-Fuzzy Ensemble model, but was still bigger than BBF. Diaz-Uriarte and Alvarez (2006) employed random forest method for

gene selection and a different method for estimating the error. It reached similar results as BBF but used more genes.

2.2 Exploring the within- and between-class correlation distribution for tumor classification

The research article by Wei and Li (2010) addressed cancer tumor classification problem based on gene-expression data. Their idea was to use the biological intuition that samples from the same tumor class usually have more similarities in terms of profiles of gene expression. Hence, the two samples that come from the same tumor class suppose to have stronger correlation. Based on this idea, a new method named "Distribution Based Classification"(DBC) was proposed.

Like most studies in this filed, overall there are basically two steps: Gene selection and classification.

In the gene selection step, the marker genes are selected by t-statistic. For kth class, the t-statistic for jth gene is defined as follows:

$$t_j^k = \frac{\bar{x}_{kj} - \bar{x}_{k'j}}{\sqrt{\frac{s_{kj}^2}{n_{kj}} + \frac{s_{k'j}^2}{n_{k'j}}}}$$

where $\bar{x}_{kj}$,s_{kj} and n_{kj} are the average, standard deviation and sample size of the expression levels of jth gene across samples belonging to the kth class in the training set and $\bar{x}_{k'j}$,$s_{k'j}$ and $n_{k'j}$ are the average, standard deviation and sample size of the expression levels of jth gene across samples not belonging to the kth class in the training set. For each class, the genes with 20 largest absolute values of t-statistic are selected as marker genes for this class.

After the gene selection step, the classification procedure uses Kullback-Leibler(KL)

distance based on the distribution of correlations. There are two types of distributions of correlations. The first type, i.e., f_{kk}, is the distribution of all the pairwise correlations of the sample profiles within class k; the second one, i.e.,f_{kj}, is the distribution of all the pairwise correlations of the sample profiles between a sample profile in class k and a sample profile in class j ($j \neq k$). KL distance is a measurement for the dissimilarity between two distributions. For classification, a new sample (with distribution f_k^*) is assigned to kth class if the KL distance between f_k^* and f_{kk} is smaller than the KL distance between f_k^* and f_{kj} ($j \neq k$).The formula for computing the Pearson Correlation between two samples is as follow:

$$Corr(\tilde{x}_1, \tilde{x}_2)/\sqrt{Var(\tilde{x}_1)Var(\tilde{x}_2)} = \sum_{i=1}^{P}(x_{1i} - \bar{x}_{1.})(x_{2i} - \bar{x}_{2.})/\sqrt{\sum_{i=1}^{P}(x_{1i} - \bar{x}_{1.})^2 \sum_{i=1}^{P}(x_{2i} - \bar{x}_{2.})^2}$$

where $\tilde{x}_1 = (x_{11}, x_{12}, ..., x_{1p})$ and $\tilde{x}_2 = (x_{21}, x_{22}, ..., x_{2p})$ be the gene-expression profiles of any two samples.

In general, suppose K and N denote the number of classes, and number of training samples, respectively. P represents the number of variables measured on each sample. A similarity score is a measurement of the similarity between two sample profiles. Specifically, in microarray data Pearson correlations are often used as similarity score. During the learning process, the distribution of the similarity scores for pairs of sample profiles from class i and class j could be calculated and denoted as f_{ij}. Then a matrix, i.e.,$\{f_{ij}\}_{K\times K}(i, j = 1, 2, \ldots, K$ and $f_{ij} = f_{jk})$ of similarity distribution for reference could be built. In the prediction process, to determine the class label of a new sample, K parallel hypotheses will be performed simultaneously. The null hypothesis is: the sample belong to kth class. The alternative hypothesis is: the sample does not belong to kth class, where $k = 1..K$. If the conclusion of these K testes are consistent, this new sample will be assigned to a class label based on the results of the tests. Otherwise, the new sample will be assigned an "unclassified" label. For unclassified samples, a new weighed KL distance rule is used for making the decision. Specifically, the new sample is assigned to the kth class that reaches the smallest KL distance, $\sum_{j=1}^{k} w_j \cdot KL(f_j^*, f_{kj})$, among all classes that claim this "unclassified" sample. If there is no class claims this sample, all classes would be considered. To avoid the cancelation of the

effect of overall shifting or rescaling caused by using correlation to build similarity scores, transformation is applied to standardize the sample profile data. Specifically, the normal score transformation is used by the formula $\{\Phi^{-1}(\frac{R_i}{P+1}), i = 1, 2, ..., P\}$, where $\Phi(.)$ is the cumulative normal distribution, R_i is the rank of the ith gene, and P is the total number of maker genes.

The experiment using the methods above was conducted on 22 gene-expression datasets with some contain binary classes and others contain multiclasses. In other words, The gene selection process was done first. Then the classification procedure was conducted by using DBC classifier, as well as several traditional classifiers including Support Vector Machine(SVM), Decision Tree(DT), Naive Bayes(NB), K nearest neighbor(KNN), and linear discriminate analysis(LDA).

For each of these dataset, 3-fold cross validation was ran for 100 simulations, and the average accuracy rates for each classifier (averaging over 100 simulations and over 22 datasets) were recorded. The result shows that DBC reaches nearly same accuracy as SVM, and NB, and is more accurate then other classifiers.

To test the robustness of the DBC method under the condition of higher noise level and switch of baseline in test set. Two simulation studies were also conducted. In the first simulation, suppose there was a binary class model, and the gene-expression data was generated from the following model:

$$y_{jk} = a_j + b_j x_k + e_{jk},$$

where j is the index for maker genes, j = 1,2,...,J, and $x_k \in \{0, 1\}$ is the class label of the kth sample, where k = 1,2,...,K. And e_{jk} is the error term following normal distribution with mean 0 and standard deviation(SD) σ. There were 50 marker genes and 50 samples in each class in the training dataset, which was simulated with a_j and b_j from uniform distribution of [-10,10] and [-2,2], respectively. The $\sigma_{training} = 2$. There were 10 samples in each class in testing dataset. Four different scenarios were simulated: first, the training

dataset and testing dataset had exactly same parameters. Second, the noise was added to the testing dataset so the $\sigma_{testing}$ was increased to 4. Third, a baseline shift was added so the gene expression value was generated from normal distribution of mean 5 and SD 2. Last, both noise and baseline shift were added to testing dataset. As a result, the DBC method performs better than other methods with a larger extent. In the second simulation, the real dataset, i.e., the prostate gene-expression data was used. However, we change the distribution of testing data (one third of the data in the 3-fold cross validation) as the following three scenarios: (1). a random normal noise with mean 0 and SD $0.5\hat{\sigma}$ were added, where $\hat{\sigma}$ was estimated by the sample standard deviation of testing data. (2). a baseline shift was added by generating data from normal distribution with mean 4 and SD $0.5\hat{\sigma}$ to all samples in training dataset. (3). doing both (1) and (2). The result shows that DBC method performs in all three scenarios above comparing with other classifiers.

It shows that the DBC classifier is more robust under the situation that noise and/or baseline shift were added to gene expression data comparing with other classifiers tested in the experiment.

2.3 Optimization Based Tumor Classification from Microarray Gene Expression Data

This article by Dagliyan et al. (2011) addressed the classification problem of tumor types with genes. Unlike several traditional methods that require the optimal parameters. A new method named Hyper-Box Enclosure (HBE),which do not need to adjust an optimal parameters for individual data set was proposed. Three methods in WEKA package were used in gene selection procedure. They are information gain attribute evaluator, relief attribute evaluator, and correlation-based feature method.

The gene classification procedure consists of Integer Programming(IP) and Mixed Integer Linear Programming (MILP) based components. The data points belonging to different classes are discriminated by Hyper-boxes. There are four steps for the Hyper-Box enclosure algorithm: seed finding, construction of boxes with seeds, intersection elimination, and optimal gene set finding.

The authors claimed that the performance of HBE overall was better than other algorithms reported in the literature and classifiers found in WEKA data mining package.

2.4 Tumor classification by combining PNN classifier ensemble with neighborhood rough set based gene reduction

The article by Wang et al. (2010) is a tumor classification problem using the gene expressions. The paper presents a method that combines ensemble of probabilistic neural network (PNN) and neighborhood rough set model based gene reduction. Both single PNN(sPNN) classifier and ensemble system of PNN (ePNN) classifier are introduced, and they both have three steps: gene ranking, gene reduction and sample classification. Two classifiers only differ in the final stage of determining the class label. The ePNN algorithm divide the entire gene set into several subsets, train the model on each subset, assign class label by each model trained, and finally decide the class label using majority rule. For gene pre-selection, two algorithms were introduced: iterative search margin based algorithm and weighted feature score criterion. Afterwards, in gene selection step, FARNeM algorithm is introduced. The experiment has been conducted on 3 public datasets which are commonly used by four computational methods with each uses a combination of the Simba, FARNeM , sPNN, ePNN and WFSC. The results show that the ePNN method not only achieved the high accuracy, but also have relative more stable performance.

2.5 Classification of Colon Tumor Tissues Using Genetic Programming

The paper by Archetti et al. (2008) presented a Genetic Programming (GP) approach for classification problem of cancer types based on gene expression values. An advantage of GP is that it can automatically select the small number of genes that are relevant and produce the classifier. In other words, unlike most other approaches that uses different algorithms for two separate steps which are gene selection and classification. The GP integrate two steps into one. The experiment has been conducted on one public available data set Colon cancer dataset which is widely used. The receiving operator (ROC) area under curve (AUC) and the measure of correctly classified instances (CCI) have been chosen as the measurement for the performance of the classifiers. For GP, the candidate classifiers (individuals) are Lisp-like tree expressions building using function set F=+,-,*,/ and a set T as terminal composed by M floating points variables. The dataset has M=2000 columns. Hence, the GP individuals are arithmetic expressions. The GP will randomly generate a large number of expressions and automatically select a rule to make a binary classification. The other parameters of GP used are: population size of 200 individuals, ramped half-and-half initialization; tournament selection of size 7, maximum tree size equal to 10, subtree crossover rate equal to 0.95, maximum number of generations equal to 500; and generational tree based GP with elitism. For comparison purpose, three non-evolutionary classifiers have also been applied on the dataset along with GP. They are Support Vector Machine (SVM), Voted Perception (VP) and Random Forest(RF). As a result, the best fitness of GP reaches 1.0 for both ROC-AUC and CCI, which is better than those three non-evolutionary classifiers. The average fitness for GP produces 0.9462 and 0.8941 as ROC-AUC, and CCI, respectively. They are competitive as SVM and better than other 2 classifiers.

2.6 Support Vector Machine model for diagnosis of lymph node metastasis in gastric cancer with multidetector computed tomography: a preliminary study

The paper by Zhang et al. (2011) addresses the classification problem for diagnosis of lymph node metastasis in gastric cancer. The imaging techniques in the stomach which have been commonly used have the difficulty of achieving high sensitivity and specificity both at the same time. A preliminary study has been done showing that Support Vector Machine (SVM) has the potential to overcome this difficulty. The dataset comes from 175 patients. The input for SVM are six indicators collected on MDCT images including serosal invasion, tumor classification, tumor maximum diameter, number of lymph nodes, maximum lymph node size and lymph nodes station. The class label being predicted has binary values which are either positive or negative. The experiment has been conducted by using free SVM software named LibSVM 2.89, and RBF has been selected as kernel function. 5-fold cross validation has been employed for training and testing, Receiver operating characteristic (ROC) curve was used to evaluate the performance of SVM. In addition, a statistical analysis including independent-sample T test and Mann-Whitney U test has been conducted on six imaging indicators. The results of classification using SVM reaches 88.5% , 78.5% and 0.876 for sensitivity, specificity, and AUC, respectively. In contrast, the radiologist only reaches 63.4% and 75.6% for sensitivity and specificity, respectively. This shows that SVM produces better results. Also, the independent t test of six indicators between positive and negative classes produces all significant differences.

Chapter 3

Methodology

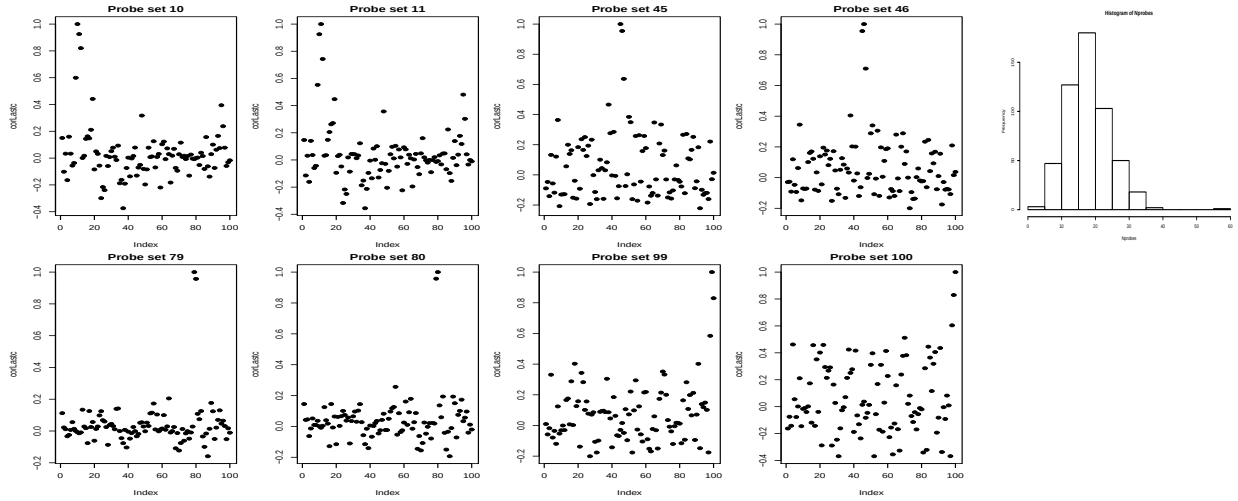
The characteristic of genomic data is that a dataset usually has a large number of features (e.g.: genes, SNPs, or probe sets) and relatively small sample size. Using all features to do the classification on phenotype is not reasonable because usually there are only some features, i.e., a subset of all features, that are biologically relevant to the phenotype. In addition, including irrelevant features could decrease the classification accuracy. Therefore, feature selection step, that is, to find the subset of features that is relevant to phenotype, is important not only for its own, but also for improving accuracy of the classification step afterwards. This paper addresses the feature selection and classification of Duke's stage using high density genomic data. We first describe the specific characteristics of the data with the initial analysis of its correlation structure to find suitable feature selection approach. Then we discuss the approach we propose for feature selection. We will use hypothesis testing-based approach for feature selection. After feature selection step, the selected SNPs will be used as features and B and C of Duke's stages will be used as class label in the classification step. We will use Support Vector Machine (SVM) and K-Nearest Neighbor (KNN) as classifiers and Leave-One-Out Cross Validation (LOOCV)as validation method to compute the accuracy as an evaluation of selected features.

3.1 Preliminary Analysis of Data Characteristic

The particular data we will consider are 94 matched pair chromosome copy number data in the genome of colorectal carcinoma using SNP-typing arrays (Kurashina et al. (2008)) and mRNA expression data from 290 primary colorectal tumor samples from Affymetrix Human Genome U133Plus 2.0 arrays (Jorissen et al. (2009)).

For the copy number dataset, genomic DNA from tumor and paired normal tissues of CRC (n=94) hybridized to Affymetrix Mapping 50K Xba 240 arrays were obtained from NCBI at http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSE11417. Chromosome copy numbers at 58,494 SNP loci were obtained with CNAG2.0 software available at http://www.genome.umin.jp. For the gene expression data, raw mRNA expression data for 54675 probe sets from 290 colorectal tumor samples were obtained from NCBI at http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSE14333. The large number of SNPs or probe sets densely cover the human genome. The genetic distance between neighboring SNPs or probe sets is very short and the expression values of these probe sets or copy numbers of these SNPs tend to have high correlations. The correlation plot in the left panel of Figure 3.1 shows the sample correlation between each probe set and other probe sets in block 23 of size 100. The plot reveals the characteristic that some probe sets have high correlation with neighboring probe sets in the same block. In fact, this characteristic happens for all blocks of size 100. In general, there are about 18 probe sets on average that shows this characteristic in one block. More specifically, the frequency for the number of probe sets that have high correlation with neighbors in all blocks ranges from 4 to 59 (see the histogram in the right panel of Figure 3.1). Due to this properties of probe sets, the expression value of neighbors that are closer might have higher correlation than those that are further away. Appropriately utilize such correlation can improve the power to identify relevant SNPs or probe sets.

Denote X_{ijk} to be the relative copy number of the jth SNP for kth subject in phenotype i.



Correlation of some probe sets with neighbors within block 23 Histogram of Nprobes

Figure 3.1: *Correlation of some probe sets with neighbors within block 23. Only probe sets that have strong correlation* (> 0.8) *with neighbor probe sets in the same block are plotted. Nprobes represents the number of probe sets that has correlation with neighbor probe sets* > 0.8

.

We assume that the series of SNPs or probe sets on a chromosome corresponding to different subjects (such as patients) are independent and each series satisfies an α-mixing condition, i.e., assume that for some sequence $\alpha_m \to 0$ as $m \to \infty$, $|P(A \cap B) - P(A)P(B)| \leq \alpha_m$ holds for all $A \in \sigma(X_{i1k}, \ldots, X_{i\ell k})$, $B \in \sigma(X_{i,\ell+m,k}, X_{i,\ell+m+1,k}, \ldots)$, and all i, k, where $\sigma(\cdot)$ denotes the σ-field generated by the random variables. The α-mixing condition basically requires the correlation between observations from the same subject to decay as the separation distance m increases. This can be seen from the following Lemma:

Lemma 3.1.1. *(Billingsley 1995) If Y is measurable $\sigma(X_{i1k}, \ldots, X_{ijk})$ and $E[Y^4] \leq C$, and if Z is measurable $\sigma(X_{i,j+n,k}, X_{i,j+n+1,k}, \ldots)$, and $E[Z^4] \leq D$, then under the α-mixing condition,*

$$|E[YZ] - E[Y]E[Z]| \leq 8(1 + C + D)\alpha_n^{1/2}. \tag{3.1.1}$$

The left-hand side in the equation above is the covariance of Y and Z. Since the variance of both Y and Z are constant, the correlation between Y and Z is also bounded by $8(1 +$

$C + D)\alpha_n^{1/2}$, which approaches 0 as α is in the order of $O(m^{-5})$.

From the analysis of the correlation structure on probe sets, we can tell it satisfies the weak dependence condition. Therefore, it is reasonable to assume a polynomial or exponential decay rate. For convenience, we use $\alpha_m = O(m^{-5})$. Most published inferences under α-mixing condition also require the stationary condition (Doukhan, Oppenheim, and Taqqu 2003; Bradley 2005; Billingsley 1995) . However, for genomic data, complicated nature of biological processes makes it unreasonable to assume stationary condition for the sequence of SNPs or probe sets. The inferences introduced in (Wang, Higgins, and Blasi 2009; Wang and Akritas 2010a; Wang and Akritas 2010b) do not require stationary condition. Since genomic data on which our studying focuses have either correlation decays fast enough to be described by the polynomial rate, the test of interaction in (Wang and Akritas 2010a; Wang and Akritas 2010b) is suitable to be used and we will employ it to test the interactions between phenotypes of disease (such as Duke's stages of colorectal cancer) and genetic markers (such as SNPs or probe sets). For data that has long range dependence, the test in Wang, Higgins, and Blasi (2009) can be used.

3.2 Feature Selection

In feature selection step, the goal is to select those genetic markers that are relevant to phenotype.

3.2.1 Identification of Significant Blocks Using Interaction Test

If there is an significant interaction effect exists between genetic markers and phenotypes, it implies that they are relevant. Therefore, the interaction test between genetic marker and phenotype can be conducted. We can set up the test in the context of the two-way factorial

design as follows:

- Factor A: phenotype of disease with level $i = 1, ..., a$.
- Factor B: genetic marker with level $j = 1, ..., b$.
- Response: gene-expression value or copy number of SNPs.

The levels of factor A depends on the number of categories of a specific phenotype. For example, the normal/tumor tissue would have 2 levels, and the Duke's stages of colorectal cancers have four levels: A, B, C, and D. The levels of factor B depends on the number of genetic markers, such as the number of genes or SNPs in a dataset.

Suppose the response variable (i.e., copy number or expression value) X_{ijk} has marginal distribution $F_{ij}(x)$ for all $k = 1, \ldots n_i$ for some unknown $F_{ij}(\cdot)$. We assume

$$F_{ij}(x) = M(x) + A_i(x) + B_j(x) + C_{ij}(x), \tag{3.2.1}$$

where $\sum_{i=1}^{a} A_i(x) = \sum_{j=1}^{b} B_j(x) = \sum_{i=1}^{a} C_{ij}(x) = \sum_{j=1}^{b} C_{ij}(x) = 0, \ \forall x$. The null hypothesis for no interaction effect between genetic marker and phenotype $H_0(C)$: all $C_{ij} = 0$ means that the marginal distribution of DNA copy number or expression value is a mixture of two components with equal mixture probability, one depending on the phenotype, and the other one depending on the genetic marker. We use (mid-) rank test (Wang and Akritas 2010b) to test the interaction effect. The test works as following:

Let R_{ijk} represent the mid-rank of observation X_{ijk}, the copy number or expression value of ith phenotype and jth genetic marker from kth subject, $k = 1, ..., n_i$, among all

observations. Denote

$$N=\sum_{i=1}^{a} n_i b, \quad \widetilde{n}=\min_{1\le i\le a}\{n_i\},$$

$$\overline{R}_{ij.}=\frac{1}{n_i}\sum_{k=1}^{n_i} R_{ijk}, \quad \widetilde{R}_{i..}=\overline{R}_{i..}=\frac{1}{b}\sum_{j=1}^{b}\overline{R}_{ij.}, \quad \overline{R}_{i.k}=\frac{1}{b}\sum_{j=1}^{b} R_{ijk},$$

$$\widetilde{R}_{...}=\frac{1}{ab}\sum_{i=1}^{a}\sum_{j=1}^{b}\overline{R}_{ij.}, \quad \widetilde{R}_{.j.}=\frac{1}{a}\sum_{i=1}^{a}\overline{R}_{ij.}, \quad \overline{R}_{...}=\frac{1}{N}\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n_i} R_{ijk},$$

Let ASC_R and ASE_R be the variations of the mean squares defined in (mid-)ranks as follows:

$$\begin{aligned} ASC_R &= \sum_{i,j}\frac{(\overline{R}_{ij.}-\widetilde{R}_{i..}-\widetilde{R}_{.j.}+\widetilde{R}_{...})^2}{(a-1)(b-1)}, \\ ASE_R &= \sum_{i,j}\sum_{k=1}^{n_i}\frac{\left(R_{ijk}-\overline{R}_{ij.}-\overline{R}_{i.k}+\widetilde{R}_{i..}\right)^2}{a(b-1)n_i(n_i-1)}, \\ F_{R,C} &= ASC_R/ASE_R \end{aligned}$$

The asymptotic distribution of $\sqrt{b}(F_{R,C}-1)$ from (Wang and Akritas 2010a) is restated below for convenience.

- If $n_i \to \infty$ as $b \to \infty$, assume $\max_i\{n_i\}/\widetilde{n} = O(1)$. Then

$$\text{under } H_0(C),\ \ \sqrt{b}(F_{R,C}-1) \xrightarrow{d} N\left(0, \widetilde{\tau}^2_{C*}/\widetilde{\sigma}^4_*\right), \tag{3.2.2}$$

- if $n_i \ge 2$ are bounded, then

$$\text{under } H_0(C),\ \ \sqrt{b}(F_{R,C}-1) \xrightarrow{d} N\left(0, \widetilde{\tau}^2_{C}/\widetilde{\sigma}^4\right); \tag{3.2.3}$$

where $\widetilde{\tau}^2_C = \lim_{b\to\infty}\left(\widetilde{\zeta}_1+\widetilde{\zeta}_2/(a-1)^2\right)$, $\quad \widetilde{\tau}^2_{C*} = \lim_{b\to\infty}\widetilde{n}^2\left(\widetilde{\zeta}_1+\widetilde{\zeta}_2/(a-1)^2\right)$,

and

$$\widetilde{\zeta}_1 = \frac{2}{a^2 b}\sum_{j=1}^{b}\sum_{j'=1}^{b}\sum_{i=1}^{a}\frac{\widetilde{\sigma}^2_{ijj'}}{n_i(n_i-1)}, \quad \widetilde{\zeta}_2 = \frac{2}{a^2 b}\sum_{j=1}^{b}\sum_{j'=1}^{b}\sum_{i\ne i'}^{a}\frac{\widetilde{\sigma}_{ijj'}\widetilde{\sigma}_{i'jj'}}{n_i n_{i'}}.$$

The terms $\widetilde{\sigma}^2$ and $\widetilde{\sigma}^2_*$ are defined as follows: let $\widetilde{\sigma}^2 = \lim_{b\to\infty} E(ASE_R)$, and $\widetilde{\sigma}^2_* = \lim_{b\to\infty} E(\widetilde{n}ASE_R)$, where

$$E(ASE_R) = \frac{1}{a(b-1)}\sum_{i,j}\frac{\widetilde{\sigma}^2_{ij}}{n_i} - \frac{1}{ab(b-1)}\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{j'=1}^{b}\frac{\widetilde{\sigma}_{ijj'}}{n_i},$$

and $Y_{ijk} = H(X_{ijk})$, $\widetilde{\sigma}_{ijj'} = cov(Y_{ijk}, Y_{ij'k})$, and $\widetilde{\sigma}_{ijj} = \widetilde{\sigma}^2_{ij} = \mathrm{Var}(Y_{ijk})$, with $H(x) = N^{-1}\sum_{i=1}^{a}\sum_{j=1}^{b} n_i F_{ij}(x)$ being the average distribution function, $c(x,y) = [I(x < y) + I(x \le y)]/2$, and $\widehat{H}(x) = N^{-1}\sum_{i=1}^{a}\sum_{j=1}^{b} n_i \widehat{F}_{ij}(x)$, $\widehat{F}_{ij}(x) = n_i^{-1}\sum_{k=1}^{n_i} c(X_{ijk}, x)$, being its empirical version of the average distribution function.

The equation (3.2.2) and (3.2.3) show that under null hypothesis, i.e. there is no significant interaction effect, the test statistic $\sqrt{b}(F_{R,C} - 1)$ converges in distribution to Normal distribution with mean 0. Using this test statistic, p-value could be calculated to make the conclusion of whether there is a significant interaction between genetic markers and phenotypes for a given significant level, i.e., whether there are some genetic markers behave differently on different levels of phenotypes.

Due to the large number genetic markers for genomic data, we divided all genetic markers into different blocks with each block containing a subset of genetic markers, aligned according to their genetic locations. After division, the blocks and the genetic markers in each block are still in the order of their relative locations. Then the test could be performed on each individual block.

There are several reasons and considerations of conducting the test on divided blocks with appropriate size on each instead of testing all genetic markers at once.

First, the interaction test only produces the conclusion of either significance or non-significance. If the interaction test described above produces a significant result, it just means a subset of all genetic markers being tested are relevant to phenotypes. Further exploration is needed to reduce the number of genetic markers and find the true relevant ones. The more genetic markers a block contains, the more difficult to find the relevant ones

within this block. Hence, we need to control each block to be within a reasonable size.

Second, the inference of the interaction test described earlier is based on the number of levels of genetic markers b approaching to ∞. Simulation studies (Wang and Akritas 2010b) showed that to conduct a valid test, b should be at least 20. Hence, we suggest to avoid having the number of genetic markers in each block being too small.

Third, considering the capacity of computing power and time cost, including too many genetic markers in one test is too expensive. Dividing them into different blocks and testing on each block could make the computing realistic.

Considering the factors discussed above, we recommend to use 100 as the size of each block except for the last block. In other words, the group of all genetic markers are divided into m blocks with each block containing 100 genetic markers and the last block contains the rest of genetic markers, whose number is at least 100 and less than 200.

To decrease the false discovery rate, we partition the subjects (patients) into ten folds. and apply the interaction test on each of the blocks using only training data (i.e. 9 folds each time) of 10-fold Cross Validation(CV). In other words, all the subjects in the sample were partitioned into 10 folds $\mathbf{Z}_{ij}, i = 1, \ldots, 10$, where j refers to jth block. Each time the subjects in $\mathbf{Z}_{ij}$ were excluded and the rest 9 folds were selected as training set and the interaction test was conducted on the training set. Therefore, there are 10 training sets to be tested for no interaction effect for block j. We let p_{ij} = "p-value for fold i and block j" represents the test results using the 9 folds without $\mathbf{Z}_{ij}$. The significance can be determined in one of the three scenarios below listed in the reverse order of conservativeness:

1 The blocks that had significant interactions across all folds were selected at a significance level. Even though testing on all blocks is in the form of multiple comparison, if some blocks produced the p-values$\leq\alpha$ consistently across all 10 folds, these blocks were considered as the ones that had significant interactions due to consistent results

from all folds.

2 For each fold, using multiple comparison control such as False Discovery Rate (FDR), a block is considered to be significant if its ordered p-value is less than or equal to the order divided by the total number of p-values for this fold. That is, denote the ordered p-values for fold i as $P_{i,(1)}, \ldots, P_{i,(m)}$. For a specified α, the kth ordered p-value is significant if $P_{i,(k)} \leq \frac{k}{m}\alpha$. This is typically referred as Banjamini-Hochberg procedure (Benjamini and Hochberg 1995). Alternative FDR control considering either positive or negative dependence exists can be incorporated by applying the procedures in Storey (2003), Clarke and Hall (2009).

3 For each fold, controlling the family-wise error rate such as Bonferroni correction, a block is considered to be significant if its p-value is less than or equal to α divided by the number of blocks.

Similar to the first scenario, the blocks that are significant across all 10 folds are selected. In principle, we recommend to carry out all three procedures above and then start with the list that contains most parsimonious, non-empty set of significant genetic markers. Typically, if all three scenarios produce non-empty set of significant genetic markers, then we would use the one that produces the fewest number of genetic markers. If the scenarios 2 and 3 found no significant genetic markers but the scenario 1 does, we would use the list produced by the scenario 1. Therefore, these significant blocks were selected and some genetic markers in each of these blocks should be relevant to phenotypes.

3.2.2 Detection of Relevant Genetic Markers Within Identified Blocks

Identification of the blocks that have significant interaction effect in previous section suggests that each of these blocks contains some genetic markers that are relevant to phenotypes.

The goal in this step is to detect those relevant genetic markers within each of significant blocks that were selected from previous step.

For each significant block, it contains 100 genetic markers (except the last block might contain more). All these 100 genetic markers are candidate markers that are relevant to phenotypes. To detect the relevant ones, some type of evaluation is needed to measure the significance of each individual genetic marker within each block. Routinely, the feature selection, i.e., the selection of genetic markers, use the methods such as Wilcoxon test or t-test on each individual genetic marker (Zhang and Deng 2007; Zhang et al. 2011). These methods perform fairly well for the datasets with relatively large sample size and middle-size (i.e. usually less than 10000) of genetic markers. However, for the dataset with small sample size and large number of genetic markers, such as the SNP copy number dataset we described at the beginning with only 94 pairs of patients and 58494 SNPs, these methods do not perform well. We tried both Mann-Whitney test (i.e., extension of Wilcoxon test) and t test on individual SNPs on copy number dataset, but failed to detect any significant SNPs using Family-Wise Error Rate with Bonferroni correction, or FDR of Banjamini-Hochberg.

Therefore, we consider to employ the test for interaction again to detect the relevant genetic markers. Since this time we need to test the significance of each individual genetic marker, but the assumption of the interaction test used earlier is that b, the number of genetic markers, should be large, testing on interaction on each genetic marker is not applicable. Instead, we use the test of interaction in a slightly modified fashion as follows:

Suppose there are total of l significant blocks selected. Let $\mathbf{M}_{ij}$ $(j = 1, \ldots, 100)$ represents the jth genetic marker in block i. In order to evaluate the significance of $\mathbf{M}_{sr}$, we perform the interaction test described earlier twice for kth training sample from the 10 folds CV:

- First, using all markers in the block $\{\mathbf{M}_{sq}, q = 1, \ldots, 100\}$ and obtain the p-value, denoted as $p_{s,k}$;

- Second, we perform the same interaction test again using $\{\mathbf{M}_{sq}, q = 1, \ldots, r-1, r+1, 100\}$, i.e., 99 genetic markers by excluding the rth one, and obtain the p-value, denoted as $p_{s,-r,k}$.

Actually, the calculation of $p_{s,k}$ (i.e., the p-value using all 100 genetic markers in one block) was already conducted when we tested the interaction for each block. The only extra test for $p_{s,-r,k}$ also has similar number of genetic markers (i.e.,99). This modified application, to maintain the number of genetic markers to be big enough, so that the interaction test is still valid. To determine the significance of each individual genetic marker we use the following analysis: In one block, since $p_{s,k}$ and $p_{s,-r,k}$ contain the sample evidence from 100 genetic markers (with $\mathbf{M}_{sr}$ present) and that from 99 genetic markers (with $\mathbf{M}_{sr}$ absent), respectively, the only difference between two p-values is caused by genetic marker $\mathbf{M}_{sr}$. Therefore, the comparison between $p_{s,k}$ and $p_{s,-r,k}$ can be analyzed to evaluate the significance of $\mathbf{M}_{sr}$, i.e., the rth genetic marker in block s.

Generally the difference or the ratio could be used to measure the comparison of two values. Considering both $p_{s,k}$ and $p_{s,-r,k}$ could be very small so the difference between the two is also very small. Ratio of the two p-values is a preferred measurement. Therefore, we use the following algorithm to detect the significant genetic markers within each block:

1. For sth significant block, use training data in fold k to compute $p_{s,-r,k}$ for $r = 1, \ldots, 100$, i.e., the p-values for excluding one genetic marker at a time, to obtain 100 p-values.

2. For block s, use training data in fold k to compute $\frac{p_{s,-r,k}}{p_{s,k}}$ for $r = 1, \ldots, 100$, i.e, the ratio of p-values between excluding each genetic marker and using all 100 genetic markers. Denote these ratios as $ratio_{srk}, r = 1, \ldots, 100$;

3. Do (1) and (2) for all significant blocks s, $s = 1, \ldots, l$, to obtain $ratio_{srk}$ for $s = 1, \ldots, l$; $(r = i, \ldots, 100)$.

4. Do (3) for all 10 folds to obtain $ratio_{irk}(i = 1, \ldots, l;\ r = 1, \ldots, 100;\ k = 1, \ldots, 10)$, i.e., the ratios for each genetic marker for all significant blocks of all 10 folds.

5. In each block s, for each genetic marker $\mathbf{M}_r$, the 10 ratios across 10 folds, i.e., $ratio_{srk}$ for $k = 1, \ldots, 10$ is treated as a random sample, denoted as $\{\mathbf{R}_{srk}, k = 1, \ldots, 10\}$.

6. In each block, for each genetic marker, i.e., $\{\mathbf{R}_{sr},\ (s = 1, \ldots, l;\ r = 1, \ldots, 100)$, we conduct the t-test for the null hypothesis "mean of $\mathbf{R}_{sr}$ is greater than 1" at a specified significance level and the lower bound of the confidence interval (CI) is computed to make the conclusion . If there are some outliers, conduct a sign test for the null hypothesis "median of $\mathbf{R}_{srk}$ is greater than 1" instead, and the conclusion is made in the same way.

The rationale behind step 6 is that if the ratio of p-values is significantly greater than 1, it means that the p-value obtained by using all 100 genetic markers is significantly smaller than the p-value using 99 markers. This implies the significance of this omitted genetic marker.

After these 6 steps, the significant genetic markers that are relevant to phenotype have been detected in each block. If there was only one significant block selected earlier, the significant genetic markers detected from this block are considered as the relevant genetic markers for feature selection. If there are more than one significant block selected earlier, then the combination of the relevant genetic markers from all these blocks are considered as the relevant genetic markers for feature selection.

3.2.3 Test for Non-significance of Remaining Genetic Markers

In order to confirm the genetic markers we selected are significant ones and none of those remaining genetic markers is significant, for each block, we conduct the same test for interaction using the set of remaining genetic markers after excluding the markers already

identified. We still use the training set from 10-fold CV to conduct the test. If test result from any of the blocks for any of the 10 folds show significance, more explorations are needed to analyze those blocks. If none of blocks from show significant result in all 10 folds, it confirms the significant genetic markers selected earlier are the key ones to phenotypes and the rest of them are not.

3.3 Classification of Phenotypes Using Selected Genetic Markers

In classification step, the goal is to build a model with training data and predict the class of test data with high accuracy. As a single split of test and training data often yield biased classification accuracy, it's recommended to use Cross Validation (CV).

As mentioned earlier, including too many irrelevant genetic markers in the classification will add noise leading to poor accuracy. Following the feature selection step, the relevant genetic markers we selected will be used as features in classification.

In terms of the choice of classifier, some literatures concluded that in general Support Vector Machine (SVM) produces the highest accuracy among common classifiers in classification problem of phenotypes using genomic data (Zhang and Deng (2007), Archetti et al. (2008), Zhang et al. (2011)). There are also other methods that perform better than SVM, such as the DBC classifier in Wei and Li (2010). KNN is also a commonly used classifier whose performance is in the medium range (Wei and Li 2010). We use SVM and KNN as classifiers to build the model.

As for the choice of validation method, since the sample size of the dataset we use is relatively small and the fact that Leave-One-Out Cross Validation(LOOCV) is an unbiased

estimator of the generalization error. We use LOOCV to validate the learned model and compute the accuracy in classification step. KNN has one parameter, k, the number of nearest neighbors. SVM contains two parameters, γ and cost. To estimate these parameters, we use 10-fold CV within the training data. The entire LOOCV proceeds as follows:

- For each of the training set $\{\mathbf{X}_{-i}, \mathbf{Y}_{-i}\}$, where $\mathbf{X}_{-i}$ is the matrix of selected features observed with subject i removed, and $\mathbf{Y}_{-i}$ is the vector of phenotypes of all subjects excluding subject i.
  - In parameter tuning step, we use 10-fold CV within $\{\mathbf{X}_{-i}, \mathbf{Y}_{-i}\}$, targeting at finding the optimal parameters.
  - After the optimal parameters have been found, we use these parameters to fit the model using $\{\mathbf{X}_{-i}, \mathbf{Y}_{-i}\}$.
- Using the fitted model, we predict the class label using $\mathbf{X}_i$, the observed feature vector for ith subject. This is the prediction step for the single observation in the test data.
- Repeat the steps above for $i = 1, \ldots, n$, where n represents the total number of subjects in the entire dataset.

It should be noted that the optimal parameters vary every time as the training set changes in LOOCV, so does the fitted model. For each step in LOOCV, a single observation is either predicted correctly or incorrectly. Then we will compute the proportion of the correctly classified subjects as the classification accuracy.

The proposed methodology is applied to the two datasets mentioned in the introduction section and the process is illustrated in the following chapter. The implementation is in statistical software R.

Chapter 4

Real Data Analysis

Current classification (prognosis) of colorectal cancer is mainly based on pathological staging. Such prognosis method faces challenge for discriminating Duke's stages B and C. Additional genetic markers are needed to help differentiate Duke's stages B and C. In this chapter, we analyze the two publicly available datasets described in Section 3.1 aiming at finding key genetic markers for classification of Duke's stages B and C with high accuracy.

For the first dataset using copy number, the original authors analyzed copy number alterations (CNA) and pointed that several chromosomes contain CNA and some chromosome arms contain loss of heterozygosity (LOH). However, neither of these CNA nor LOH were used to classify the clinical outcome of Duke's stages. For the second dataset using mRNA expression data, the original authors identified 128 genes using 3 training sets for Duke's stages A and D. They then used these genes and a Prediction Analysis of Microarray (PAM) algorithm to classify patients of Duke's stages B and C into stage A-like/good prognosis or stage D-like/poor prognosis types.

Using the method presented in Chapter 3 on both datasets of chromosome copy number and mRNA expression data, we successfully identified a very small set of genetic markers in

feature selection step, and achieved 100% LOOCV accuracy in classification step of Duke's stages B and C on both datasets.

In the following two sections, we describe the process of applying the proposed method in Chapter 3 to the two datasets. The programming code of the implementation in R is given in the Appendices A.2 and B.2.

4.1 Feature Selection and Classification of Duke's Stage Using Chromosome Copy Number Dataset

4.1.1 Data Preprocessing

The raw chromosome copy number dataset was downloaded directly from NCBI (Accession number: GSE11417) in the zipped CEL file format. The physical characteristics of 94 patients were downloaded from 188 links through the java program in Appendix A.1 (first link: http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSM288035, last link: http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSM288222). The downloaded raw data was preprocessed using the software CNAG (Nannya et al. 2005) with the option of the paired sample to obtain the copy number. The obtained relative copy number from CNAG at each SNP locus is estimated from the log 2 ratio of the normalized signals of each SNP in the diseased sample and paired normal sample. The SNPs are aligned in relative position same as their locations on chromosomes.

4.1.2 Feature Selection of SNP's

After data preprocessing, the dataset consists of the copy number dataset and the corresponding Duke's stage from patients' physical characteristics. The copy number dataset is a 58494×94 matrix with each row represents the copy numbers of one SNP and each column represents one patient. The 58494 SNPs were divided into 584 blocks with each block contains 100 SNPs except that the last block containing 194 SNPs. The number of patients with Duke's stages A, B, C, and D are 3, 46, 37, and 8, respectively. Since our interest is to find SNPs to differentiate Duke's stages B and C, we work only with the 46 patients with stage B disease and 37 patients with stage C disease. Excluding the small number of Duke's stages A and D patients is also consistent with the conditions in the interaction test, which requires that the sample sizes of different levels of phenotypes are of the same order. Therefore, the total sample size of the dataset is 83 which is the total number of patients with Duke's stages B and C. We partition the 46 patients with Duke's stage B into 10 folds, among which 4 (i.e, truncate 46/10) of them are denoted as test set and the rest are training set. Similarly, We partition the 37 patients with Duke's stage B into 10 folds, among which 3 (i.e., truncate 37/10) of them are denoted as test set and the rest are training set. Then we combine the training sets from both stages and apply the test for interaction effect for each block. As we move along the index of the folds, we get 10 training sets.

After application of the test, attempts were made for detecting those blocks for significant results for each fold. With Bonferroni or FDR multiple comparison adjustment, none of the blocks was found to be significant consistently for every fold. Instead, significant blocks were selected based on $\alpha = 0.05$ for each of the 10 folds. Since 115th block showed consistently significant results across all 10 folds, this block was selected as the block from which some SNPs are relevant to the variations among Duke's stages.

Next, we seek to identify the relevant SNPs within 115th block using the training set of LOOCV. To find the significant SNPs in this block, each SNP was excluded from the block

one at a time, and the p-values for testing of no interaction effect with the rest of 99 SNPs for each of the 10 folds. As we go through all the SNPs, we obtained 100 p-values each fold. Equivalently, for each SNP, there are 10 p-values, one from each fold.

Recall that there was also the original p-value calculated using all 100 SNPs for each fold. For each SNP, we calculated the ratio of the p-value using 99 SNPs (after excluding this SNP) to that using 100 SNPs. Then the ratios across 10 folds for each SNP serves as a 'random' sample, and the t-test was conducted on each sample of size 10 to select those SNPs with the mean ratio significantly greater than 1. Fifteen SNPs were found. To make sure that there are no other SNPs relevant to the variation of the Duke's stages, we double check through excluding these 15 SNPs and test the remaining 85 SNPs for each of the 10 folds. The results show that none of them is significant. The 15 SNPs in Table 4.1 are the significant ones such that the rest of the SNPs on the block lead to non-significant test of the interaction effect after these SNPs were removed.

From literature review, other methods for feature selection used t-test or Mann-Whitney test on each individual genetic marker and the selection was based on q-values of those genetic markers ranked top with FDR control (Storey and Tibshirani 2003). In order to compare the results for feature selection using our proposed method with these other methods, we also calculated the p-values of each SNP using t-test and Mann-Whitney test. Different multiple comparison criteria were considered, i.e., Family Wise Error Rate (FWER) with Bonferroni correction and FDR control. The results from neither of the two tests found significant SNP with either FDR or FWER control.

4.1.3 Classification on Duke's Stages

After feature selection step, we used the selected 15 SNPs to classify Duke's stages B and C. SVM and KNN were employed as the classifier and LOOCV was used as validation method in classification step. For each patient as the test set, the LOOCV uses data from the rest

Table 4.1: *Significant SNPs found from block 115*

SNP	p.value	99% lower.bound	Mean Ratio Across 10 folds
SNP_A-1657650	0.0030	2.7766	9.5100
SNP_A-1749435	0.0019	2.7983	7.6780
SNP_A-1689468	0.0012	2.1534	4.5120
SNP_A-1689889	0.0000	2.2672	3.0720
SNP_A-1648196	0.0006	1.4614	2.1800
SNP_A-1727755	0.0026	1.2480	2.0790
SNP_A-1752279	0.0000	1.3767	1.5470
SNP_A-1695975	0.0040	1.1174	1.7010
SNP_A-1702481	0.0003	1.2377	1.5310
SNP_A-1734536	0.0002	1.2397	1.5070
SNP_A-1648758	0.0005	1.1598	1.3860
SNP_A-1741653	0.0010	1.1470	1.4310
SNP_A-1713231	0.0000	1.1453	1.2500
SNP_A-1728065	0.0011	1.0926	1.2800
SNP_A-1678518	0.0060	1.0095	1.0950

of the patients as the training data. The optimal parameters (γ and cost) in SVM were tuned using the 10-fold CV within the training data. Hence the sets of parameters might be different for different patients used in test data as shown in Table 4.2. Then the optimal parameters were used to build the final SVM model for prediction. By comparing the observed class label, each patient's Duke's stage is either correctly classified or incorrectly classified. The result showed that we correctly classified the Duke's stages for all 83 patients in LOOCV.

For KNN classifier, 10-fold CV was used to estimate the optimal parameter k within the training set of LOOCV. Then the trained model with the optimal parameter for each patient was used to predict the class label. The optimal parameters and the classification

results are shown in Table 4.3. The Duke's stages of all patients were correctly classified.

4.2 Feature Selection and Classification of Duke's Stage Using mRNA Expression Data

4.2.1 Data Preprocessing

The raw mRNA expression dataset was downloaded from NCBI (Accession number: GSE14333) in CEL files. The raw data were normalized with bioconductor package gcrma to summarize probe sets expression. The patients' Duke's stages were retrieved from 290 links using the Java program in Appendix B.1. To find the relative locations of the probe sets on their chromosomes, we downloaded the annotation file of human U 133 plus 2.0 array from the Affymatrix website. Based on the "Chromosomal location" column from annotation file, the sex chromosomes were identified and removed. In addition, the remaining probe sets were sorted based on the location information obtained from the "Alignment" column from the annotation file.

4.2.2 Feature Selection of Probe Sets

After data preprocessing step, we obtained the dataset that contains both mRNA expression values and the Duke's stages. For each of the 290 patients, there are 53158 probe sets after removing the probe sets from sex chromosomes aligned based on their chromosome locations (except for 1075 probe sets that did not have alignment information).

The total 53158 probe sets were divided into 531 blocks of size 100 with the exception of the last block containing 158 probe sets. Similarly to the objective for the copy number

dataset, we are still interested in finding relevant probe sets to classify Duke's stages B and C. There are 94 patients with Duke's stage B and 91 patients with Duke's stage C disease. The patients in each Duke's stage were partitioned into 10 folds, among which 1 fold was excluded at a time and the rest 9 folds were used as training data. That is, each of the 10 training sets contains 85 patients with Duke's stage B and 82 patients with Duke's stage C.

The test of interaction effect was applied to each block for each fold. We used Bonferroni and FDR multiple comparison correction for detecting the significant blocks. The 23th bock was detected as the only significant block under both criteria. The next step is to identify the relevant probe sets within this block using the training set of LOOCV. To achieve this, each probe set was excluded from the block and a p-value was obtained by testing the interaction effect of this block with remaining 99 probe sets for each fold. Such calculation was repeated for all probe sets, yielding 100 p-values for each fold. This group of p-values was used along with the original p-value from the entire block to get the ratio of p-values. Different from the analysis in previous copy number dataset, these ratios from different folds do not seem to have common distribution. Instead the ratios for fold 1 are obviously much bigger than those for other folds (see the boxplot in figure 4.1).

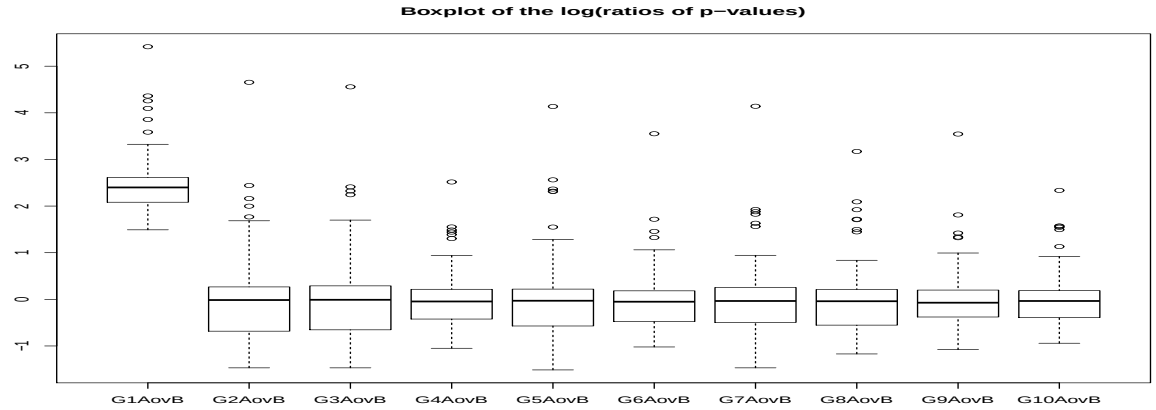


Figure 4.1: *Boxplot of the log(ratios of p-values) between the test with 100 probe sets and 99 probe sets.*

Then the ratios across 10 folds for each probe set serves as a 'random' sample. Since the fold 1 has extreme ratios, the sign test was conducted to select those probe sets with the median ratio significantly greater than 1. Thirty six probe sets were found as shown in

table 4.4.

Further confirmation was conducted to guarantee that there are no other probe sets relevant to the variation of the Duke's stages through excluding these 36 probe sets and test the remaining 64 probe sets for each of the 10 folds. None of them was significant as a result.

4.2.3 Classification on Duke's Stages

We used the selected 36 probe sets from block 23 to classify the Duke's stage with SVM as classifier and LOOCV as validation method. The training sample size was 184, and the test sample size is 1. The class of the test data was predicted based on the model trained with the training set with optimal parameters determined through 10-fold CV on the same training data using SVM and KNN. Incidentally, the optimal parameters for all training sets appear to be the same for SVM. They are $\gamma = 0.125$, cost=4. The class label, i.e, Duke's stages of all 185 of patients, were correctly classified. Therefore, the average accuracy is 100%. We also used K-Nearest Neighbor (KNN) as the classifier using the selected 36 probe sets to classify Duke's stages. Similar to SVM, 10-fold CV was used to estimate the optimal parameter k with the training set of LOOCV. Then the trained model with the optimal parameter for each patient was used to predict the class label. The optimal parameters and the classification result are shown in Table 4.5. Again, the Duke's stages of all patients were correctly classified.

Table 4.2: *Optimal parameter estimates and classification results of SVM for DNA copy number dataset*

Patient	γ	Cost	Correctly classified	Patient	γ	Cost	Correctly classified
Patient 2	0.125	4	Yes	Patient 83	0.125	8	Yes
Patient 3	0.125	4	Yes	Patient 87	0.125	4	Yes
Patient 6	0.125	8	Yes	Patient 91	0.125	4	Yes
Patient 7	0.125	4	Yes	Patient 93	0.125	4	Yes
Patient 8	0.125	4	Yes	Patient 4	0.125	8	Yes
Patient 9	0.125	4	Yes	Patient 5	0.125	8	Yes
Patient 10	0.5	4	Yes	Patient 11	0.125	4	Yes
Patient 12	0.125	4	Yes	Patient 14	0.125	8	Yes
Patient 13	0.125	8	Yes	Patient 17	0.25	4	Yes
Patient 16	0.125	4	Yes	Patient 23	0.25	4	Yes
Patient 18	0.125	4	Yes	Patient 28	0.125	4	Yes
Patient 19	0.125	8	Yes	Patient 31	0.5	4	Yes
Patient 20	0.125	8	Yes	Patient 34	0.125	4	Yes
Patient 21	0.25	4	Yes	Patient 35	0.125	8	Yes
Patient 22	0.125	4	Yes	Patient 36	0.125	8	Yes
Patient 24	0.125	4	Yes	Patient 39	0.125	8	Yes
Patient 25	0.125	4	Yes	Patient 43	0.125	4	Yes
Patient 27	1	4	Yes	Patient 44	0.125	4	Yes
Patient 29	0.125	4	Yes	Patient 48	0.125	4	Yes
Patient 32	0.125	4	Yes	Patient 51	0.125	4	Yes
Patient 33	0.125	4	Yes	Patient 53	0.125	4	Yes
Patient 37	0.125	4	Yes	Patient 54	0.125	8	Yes
Patient 40	0.25	4	Yes	Patient 57	0.125	4	Yes
Patient 41	0.25	4	Yes	Patient 59	0.125	8	Yes
Patient 42	0.125	8	Yes	Patient 62	0.5	4	Yes
Patient 45	0.125	8	Yes	Patient 64	0.125	4	Yes
Patient 46	0.25	4	Yes	Patient 66	0.125	8	Yes
Patient 47	0.125	4	Yes	Patient 67	0.25	4	Yes
Patient 50	0.125	4	Yes	Patient 69	0.125	4	Yes
Patient 52	0.125	8	Yes	Patient 72	0.125	8	Yes
Patient 55	0.125	4	Yes	Patient 73	0.125	4	Yes
Patient 56	0.125	4	Yes	Patient 76	0.125	4	Yes
Patient 58	0.125	4	Yes	Patient 77	0.125	4	Yes
Patient 60	0.125	8	Yes	Patient 84	0.125	4	Yes
Patient 63	0.125	8	Yes	Patient 85	0.25	4	Yes
Patient 65	0.125	8	Yes	Patient 86	0.125	8	Yes
Patient 68	0.125	8	Yes	Patient 88	0.125	4	Yes
Patient 70	0.125	8	Yes	Patient 89	0.25	4	Yes
Patient 71	0.125	4	Yes	Patient 90	0.125	8	Yes
Patient 74	0.125	8	Yes	Patient 92	0.125	4	Yes
Patient 78	0.125	8	Yes	Patient 94	0.125	8	Yes
Patient 81	0.125	8	Yes				

Table 4.3: *Estimate of k and Classification Accuracy of KNN for Copy Number Dataset*

Subject	k	Correctly classified	Subject	k	Correctly classified
Patient 2	16	Yes	Patient 83	23	Yes
Patient 3	24	Yes	Patient 87	25	Yes
Patient 6	14	Yes	Patient 91	25	Yes
Patient 7	14	Yes	Patient 93	16	Yes
Patient 8	17	Yes	Patient 4	18	Yes
Patient 9	17	Yes	Patient 5	16	Yes
Patient 10	15	Yes	Patient 11	18	Yes
Patient 12	17	Yes	Patient 14	15	Yes
Patient 13	22	Yes	Patient 17	14	Yes
Patient 16	16	Yes	Patient 23	17	Yes
Patient 18	15	Yes	Patient 28	23	Yes
Patient 19	14	Yes	Patient 31	13	Yes
Patient 20	22	Yes	Patient 34	22	Yes
Patient 21	15	Yes	Patient 35	21	Yes
Patient 22	15	Yes	Patient 36	13	Yes
Patient 24	17	Yes	Patient 39	17	Yes
Patient 25	17	Yes	Patient 43	15	Yes
Patient 27	23	Yes	Patient 44	14	Yes
Patient 29	18	Yes	Patient 48	18	Yes
Patient 32	14	Yes	Patient 51	16	Yes
Patient 33	16	Yes	Patient 53	19	Yes
Patient 37	18	Yes	Patient 54	14	Yes
Patient 40	16	Yes	Patient 57	14	Yes
Patient 41	16	Yes	Patient 59	14	Yes
Patient 42	17	Yes	Patient 62	18	Yes
Patient 45	16	Yes	Patient 64	14	Yes
Patient 46	14	Yes	Patient 66	16	Yes
Patient 47	17	Yes	Patient 67	16	Yes
Patient 50	16	Yes	Patient 69	19	Yes
Patient 52	13	Yes	Patient 72	20	Yes
Patient 55	14	Yes	Patient 73	18	Yes
Patient 56	18	Yes	Patient 76	22	Yes
Patient 58	19	Yes	Patient 77	13	Yes
Patient 60	18	Yes	Patient 84	14	Yes
Patient 63	16	Yes	Patient 85	14	Yes
Patient 65	19	Yes	Patient 86	15	Yes
Patient 68	14	Yes	Patient 88	15	Yes
Patient 70	25	Yes	Patient 89	18	Yes
Patient 71	18	Yes	Patient 90	13	Yes
Patient 74	25	Yes	Patient 92	14	Yes
Patient 78	13	Yes	Patient 94	15	Yes
Patient 81	15	Yes			

Table 4.4: *Probe sets with median ratios significantly greater than 1*

	99lower.bound	median.ratio.across.folds
1555896_a_at	12.2342	48.7050
213448_at	4.8124	7.4800
210132_at	3.7970	5.4800
236224_at	4.2232	5.8050
241377_s_at	2.6757	5.1950
214532_x_at	3.0840	4.5900
236223_s_at	2.3616	2.8700
209197_at	2.0262	2.4400
221115_s_at	1.5824	2.0150
243463_s_at	1.5777	1.8700
210589_s_at	1.5592	1.8000
244489_at	1.5662	1.7400
223555_at	1.3692	1.5650
209198_s_at	1.3932	1.5450
206635_at	1.3777	1.4900
209093_s_at	1.3592	1.4050
237810_at	1.2770	1.4400
222584_at	1.3292	1.4400
244803_at	1.2477	1.4650
203229_s_at	1.2970	1.4250
1554057_at	1.2762	1.3800
230256_at	1.1977	1.3350
218815_s_at	1.2492	1.2950
218873_at	1.2300	1.2450
226644_at	1.1385	1.3000
212259_s_at	1.1700	1.1950
208286_x_at	1.1885	1.2150
205107_s_at	1.1200	1.1650
217007_s_at	1.1292	1.1750
235145_at	1.1292	1.1450
222480_at	1.1285	1.1550
205661_s_at	1.0192	1.0900
203515_s_at	1.0377	1.0750
202023_at	1.0270	1.0850
212541_at	1.0277	1.0650
241389_at	1.0177	1.0600

Table 4.5: *Estimate of k and Classification Accuracy of KNN for mRNA Expression Dataset*

Subject	k	Correctly classified	Subject	k	Correctly classified	Subject	k	Correctly classified
Patient 2	1	Yes	Patient 170	3	Yes	Patient 107	5	Yes
Patient 3	5	Yes	Patient 171	1	Yes	Patient 112	1	Yes
Patient 6	1	Yes	Patient 174	7	Yes	Patient 116	3	Yes
Patient 7	1	Yes	Patient 176	5	Yes	Patient 123	3	Yes
Patient 10	6	Yes	Patient 177	1	Yes	Patient 129	1	Yes
Patient 15	5	Yes	Patient 182	3	Yes	Patient 130	1	Yes
Patient 16	4	Yes	Patient 184	1	Yes	Patient 131	1	Yes
Patient 18	1	Yes	Patient 187	5	Yes	Patient 134	1	Yes
Patient 19	4	Yes	Patient 189	5	Yes	Patient 136	1	Yes
Patient 24	5	Yes	Patient 195	4	Yes	Patient 139	3	Yes
Patient 25	3	Yes	Patient 197	5	Yes	Patient 141	5	Yes
Patient 28	1	Yes	Patient 202	1	Yes	Patient 142	3	Yes
Patient 30	1	Yes	Patient 209	1	Yes	Patient 143	3	Yes
Patient 40	5	Yes	Patient 217	1	Yes	Patient 146	1	Yes
Patient 41	3	Yes	Patient 218	3	Yes	Patient 151	5	Yes
Patient 42	1	Yes	Patient 219	3	Yes	Patient 153	4	Yes
Patient 45	6	Yes	Patient 224	1	Yes	Patient 154	5	Yes
Patient 48	5	Yes	Patient 227	4	Yes	Patient 159	1	Yes
Patient 51	1	Yes	Patient 229	1	Yes	Patient 161	1	Yes
Patient 53	1	Yes	Patient 243	1	Yes	Patient 164	5	Yes
Patient 54	1	Yes	Patient 251	3	Yes	Patient 166	1	Yes
Patient 55	1	Yes	Patient 254	1	Yes	Patient 167	3	Yes
Patient 56	5	Yes	Patient 257	1	Yes	Patient 173	1	Yes
Patient 60	1	Yes	Patient 259	3	Yes	Patient 175	1	Yes
Patient 63	5	Yes	Patient 263	1	Yes	Patient 178	1	Yes
Patient 64	4	Yes	Patient 269	1	Yes	Patient 179	3	Yes
Patient 67	3	Yes	Patient 270	1	Yes	Patient 183	1	Yes
Patient 68	1	Yes	Patient 271	6	Yes	Patient 185	5	Yes
Patient 70	5	Yes	Patient 274	1	Yes	Patient 186	5	Yes
Patient 74	3	Yes	Patient 276	1	Yes	Patient 190	1	Yes
Patient 77	1	Yes	Patient 283	5	Yes	Patient 192	3	Yes
Patient 79	1	Yes	Patient 290	1	Yes	Patient 194	7	Yes
Patient 82	3	Yes	Patient 1	7	Yes	Patient 199	1	Yes
Patient 83	5	Yes	Patient 4	3	Yes	Patient 200	3	Yes
Patient 84	1	Yes	Patient 9	5	Yes	Patient 204	4	Yes
Patient 88	1	Yes	Patient 11	1	Yes	Patient 206	1	Yes
Patient 94	1	Yes	Patient 12	3	Yes	Patient 211	5	Yes
Patient 95	5	Yes	Patient 14	3	Yes	Patient 212	3	Yes
Patient 98	1	Yes	Patient 20	2	Yes	Patient 213	3	Yes
Patient 101	1	Yes	Patient 23	6	Yes	Patient 216	1	Yes
Patient 103	5	Yes	Patient 26	3	Yes	Patient 220	1	Yes
Patient 104	3	Yes	Patient 27	3	Yes	Patient 223	1	Yes
Patient 108	3	Yes	Patient 29	1	Yes	Patient 225	1	Yes
Patient 109	1	Yes	Patient 31	1	Yes	Patient 231	1	Yes
Patient 114	1	Yes	Patient 36	7	Yes	Patient 233	3	Yes
Patient 115	1	Yes	Patient 37	1	Yes	Patient 235	1	Yes
Patient 119	1	Yes	Patient 39	3	Yes	Patient 238	1	Yes
Patient 120	4	Yes	Patient 44	4	Yes	Patient 239	1	Yes
Patient 127	1	Yes	Patient 50	1	Yes	Patient 240	6	Yes
Patient 128	1	Yes	Patient 52	3	Yes	Patient 241	3	Yes
Patient 133	3	Yes	Patient 65	3	Yes	Patient 242	3	Yes
Patient 135	1	Yes	Patient 66	2	Yes	Patient 246	5	Yes
Patient 137	1	Yes	Patient 76	5	Yes	Patient 247	5	Yes
Patient 138	6	Yes	Patient 78	4	Yes	Patient 250	8	Yes
Patient 140	4	Yes	Patient 87	4	Yes	Patient 252	5	Yes
Patient 144	5	Yes	Patient 92	3	Yes	Patient 262	6	Yes
Patient 147	4	Yes	Patient 93	5	Yes	Patient 278	3	Yes
Patient 150	2	Yes	Patient 96	4	Yes	Patient 279	3	Yes
Patient 162	1	Yes	Patient 97	1	Yes	Patient 282	5	Yes
Patient 163	1	Yes	Patient 99	4	Yes	Patient 286	1	Yes
Patient 165	1	Yes	Patient 105	1	Yes	Patient 287	1	Yes
Patient 169	3	Yes	Patient 106	5	Yes			

Conclusion

Overall, this report reviewed some existing work addressing the feature selection of genes and classification on cancer types and prognosis using genomic data. Regardless of the approaches, this type of study typically could be divided into two steps: gene selection and type classification. Most of the methods we reviewed made selection based on rankings of the individual gene using some criteria. Due to the modern technology, more microarray data with high densities are available today. Our study tackled the feature selection of genetic markers and classification of phenotypes using these data with high densities. Incorporating the possible strong correlation between the genetic markers that are neighbors in terms of their chromosome locations, a novel method was proposed. Instead of selection based on individual genetic marker, we made the selection based on the unit of block (i.e., a group of contiguous genetic markers) first and then select the relevant genetic markers within significant blocks. From statistical perspective, a test for interaction between genetic markers and phenotypes was used throughout to detect significant blocks and the relevant genetic markers within each block. This method was applied to two high density datasets. After feature selection, classification was made on Duke's stages using SVM and KNN as classifiers and LOOCV as test method. Excellent accuracies obtained showed the effectiveness of the proposed procedure for feature selection. The traditional methods which make selection based on individual gene are more flexible on the structure of microarray data and the size the selected genes. Our proposed method could make use of modern data with high density. It takes into account of correlation information based on chromosome locations such that high accuracy could be obtained through precise selection of genetic markers that are discriminative to the phenotypes. Even though perfect accuracies were obtained using our method on two datasets, the allocation of contribution made by selected features and

suitable classifiers are still worth further exploration by applying our proposed method to more different types of datasets using various other classifiers.

Bibliography

Alon, U., N. Barkai, D. Notterman, K. Gish, S. Ybarra, D. Mack, and A. Levine (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Accademy of Siences of the United States of America 96*, 6745–6750.

Archetti, F., M. Castelli, I. Giordani, and L. Vanneschi (2008). Classification of colon tumor tissues using genetic programming. *Proceedings of the Annual Italian Workshop on Artificial Life and Evolutionary Computation*.

Ben-Dor, A., L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and Z. Yakhini (2000). Tissue classification with gene expression profiles. *Proceedings of the fourth annual international Conference on Computational molecular biology*, 54–64.

Benjamini, Y. and Y. Hochberg (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. R. Statist. Soc. B 57*, 289–300.

Beroukhim, R., C. H. Mermel, D. Porter, G. Wei, S. Raychaudhuri, J. Donovan, J. Barretina, J. S. Boehm, J. Dobson, M. Urashima, K. T. Mc Henry, R. M. Pinchback, A. H. Ligon, Y.-J. Cho, L. Haery, H. Greulich, M. Reich, W. Winckler, M. S. Lawrence, B. A. Weir, K. E. Tanaka, D. Y. Chiang, A. J. Bass, A. Loo, C. Hoffman, J. Prensner, T. Liefeld, Q. Gao, D. Yecies, S. Signoretti, E. Maher, F. J. Kaye, H. Sasaki, J. E. Tepper, J. A. Fletcher, J. Tabernero, J. Baselga, M.-S. Tsao, F. Demichelis, M. A. Rubin, P. A. Janne, M. J. Daly, C. Nucera, R. L. Levine, B. L. Ebert, S. Gabriel, A. K. Rustgi, C. R. Antonescu, M. Ladanyi, A. Letai, L. A. Garraway, M. Loda, D. G. Beer, L. D. True, A. Okamoto, S. L. Pomeroy, S. Singer, T. R. Golub, E. S. Lander, G. Getz, W. R. Sellers, and M. Meyerson (2010). The landscape of somatic copy-number alteration across human cancers. *Nature 463*(7283), 899–905.

Billingsley, P. (1995). *Probability and Measure*. Third Edition, Wiley, New-York.

Bomme, L., G. Bardi, N. Pandis, C. Fenger, O. Kronborg, and S. Heim (1998). Cytogenetic analysis of colorectal adenomas: Karyotypic comparisons of synchronous tumors. *Cancer Genetics and Cytogenetics 106*(1), 66 – 71.

Bradley, R. (2005). Basic properties of strong mixing conditions. A survey and some open questions. *Probability Surveys 2*, 1549–5787.

Clarke, S. and P. Hall (2009). Robustness of multiple testing procedures against dependence. *Ann. Statist. 37*, 332–358.

Dagliyan, O., F. Uney-Yuksektepe, I. H. Kavakli, and M. Turkay (2011, 02). Optimization based tumor classification from microarray gene expression data. *PLoS ONE 6*, e14579.

Davis, E. E., C. Ma, A. Rosenwald, H. Sabet, T. Tran, X. Yu, L. Yang, T. Moore, J. Hudson, L. Lu, R. Tibshirani, G. Sherlock, R. Warnke, R. Levy, W. Wilson, and D. Botstein (2000). Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature 403*, 503–511.

Dettling, M. and P. Buhlmann (2003). Boosting for tumor classification with gene expression data. *Bioinformatics 19*, 9.

Diaz-Uriarte, R. and d. A. S. Alvarez (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics 7*, 3.

Ding, C. and H. Peng (2005). Minimum redundancy feature selection from microarray gene expression data. *Bioinform. Comput. Biol. 3*, 185–205.

Doukhan, P., G. Oppenheim, and M. S. Taqqu (2003). *Theory and Applications of Long-Range Dependence*. Birkhauser Boston, c/o Springer-Verlag: New York.

Golub, T., D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, and E. Lander (1999). Molecular classification of cancer: dass discovery and class prediction by gene expression monitoring. *Science 286*, 531–537.

Hermsen, M., C. Postma, J. Baak, M. Weiss, A. Rapallo, A. Sciutto, G. Roemen, J.-W. Arends, R. Williams, W. Giaretti, A. de Goeij, and G. Meijer (2002). Colorectal ade-

noma to carcinoma progression follows multiple pathways of chromosomal instability. *Gastroenterology 123*(4), 1109–1119.

Jorissen, R., P. Gibbs, M. Christie, S. Prakash, L. Lipton, J. Desai, D. Kerr, L. Aaltonen, D. Arango, M. Kruhoffer, T. Orntoft, C. Andersen, M. Gruidl, V. Kamath, S. Eschrich, T. Yeatman, and O. Sieber (2009). Metastasis-associated gene expression changes predict poor outcomes in patients with dukes stage B and C colorectal cancer. *Clin. Cancer Res. 15*, 7642–7651.

Jorissen, R. N., P. Gibbs, M. Christie, S. Prakash, L. Lipton, J. Desai, D. Kerr, L. A. Aaltonen, D. Arango, M. Kruhoffer, and et al. (2009). Metastasis-associated gene expression changes predict poor outcomes in patients with dukes stage b and c colorectal cancer. *Clinical Cancer Research 15*(24), 7642–7651.

Kurashina, K., Y. Yamashita, T. Ueno, K. Koinuma, J. Ohashi, H. Horie, Y. Miyakura, T. Hamada, H. Haruta, H. Hatanaka, M. Soda, Y. Choi, S. Takada, Y. Yasuda, H. Nagai, and H. Mano (2008). Chromosome copy number analysis in screening for prognosis-related genomic regions in colorectal carcinoma. *Cancer Sci. 99*, 1835–40.

Liu, X., A. Krishnan, and A. Mondry (2005). An entropy-based gene selection method for cancer classification using microarray data. *BMC Bioinformatics 6*, 76.

Markowitz, S. D. and M. M. Bertagnolli (2009, December). Molecular origins of cancer: Molecular basis of colorectal cancer. *The New England journal of medicine 361*(25), 2449–2460.

Nannya, Y., M. Sanada, K. Nakazaki, N. Hosoya, L. Wang, A. Hangaishi, M. Kurokawa, S. Chiba, D. K. Bailey, G. C. Kennedy, , and S. Ogawa (2005). A robust algorithm for copy number detection using high-density oligonucleotide single nucleotide polymorphism genotyping arrays. *Cancer Res. 65*, 6071–6079.

Pihan, G. A., J. Wallace, Y. Zhou, and S. J. Doxsey (2003). Centrosome abnormalities and chromosome instability occur together in pre-invasive carcinomas. *Cancer Research 63*(6), 1398–404.

Rajagopalan, H., M. A. Nowak, B. Vogelstein, and C. Lengauer (2003). The significance

of unstable chromosomes in colorectal cancer. *Nature reviews. Cancer 3*(9), 695–701.

Shipp, M., K. Ross, P. Tamayo, A. Weng, J. Kutok, R. Aguiar, M. Gaasenbeek, M. Angelo, M. Reich, G. Pinkus, T. Ray, M. Koval, K. Last, A. Norton, T. Lister, J. Mesirov, D. Neuberg, E. Lander, J. Aster, and T. Golub (2002). Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nat. Med. 8*, 68–74.

Singh, D., P. Febbo, K. Ross, D. Jackson, J. Manola, C. Ladd, P. Tamayo, A. Renshaw, A. D'Amico, J. Richie, E. Lander, M. Loda, P. Kantoff, T. Golub, and W. Sellers (2002). Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell 1*, 203–209.

Smith, J. J., N. G. Deane, F. Wu, N. B. Merchant, B. Zhang, A. Jiang, P. Lu, J. C. Johnson, C. Schmidt, C. E. Bailey, and et al. (2010). Experimentally derived metastasis gene expression profile predicts recurrence and death in patients with colon cancer. *Gastroenterology 138*(3), 958–968.

Storey, J. and R. Tibshirani (2003). Statistical significance for genome-wide studies. *Proc. Natl. Acad. Sci. USA 100*, 9440–9445.

Storey, J. D. (2003). The positive false discovery rate: A bayesian interpretation and the q-value. *Ann. Statist. 31*, 2013–2035.

Tibshirani, R., T. Hastie, B. Narasimhan, and G. Chu (2002). Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences of the United States of America 99*(10), 6567–6572.

Wang, H. and M. Akritas (2010a). Inference from heteroscedastic functional data. *J. Nonparametr. Stat. 22*(2), 149–168.

Wang, H. and M. Akritas (2010b). Rank test for heteroscedastic functional data. *J. Multivariate Anal. 101*, 1791–1805.

Wang, H., J. Higgins, and D. Blasi (2009). Distribution-free tests for no effect of treatment in heteroscedastic functional data under both weak and long range dependence. *Statistics and Probability Letters 80*, 390–402.

Wang, S.-L., X. Li, S. Zhang, J. Gui, and D.-S. Huang (2010). Neighborhood rough set reduction-based gene selection and prioritization for gene expression profile analysis and molecular cancer classification. *J. Biomed. Biotechnol. 40*, 179–189.

Wang, Z., V. Palade, Xu, and Y. (2006). Neuro-fuzzy ensemble approach for microarray cancer gene expression data analysis. *Proc of the Second International Symposium on Evolving Fuzzy System (EFS'06), IEEE Computational Intelligence Society*, 241–246.

Wei, X. and K.-C. Li (2010). Exploring the within- and between-class correlation distributions for tumor classification. *Proc. Natl. Acad. Sci. USA 107*(15), 6737–6742.

Weston, J., S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik (2000). Feature selection for svms. *Advances in Neural Information Processing Systems*.

Yang, K., Z. Cai, L. J., and L. G (2006). A stable gene selection in microarray data analysis. *BMC Bioinformatics 7*, 228.

Zhang, J.-G. and H.-W. Deng (2007). Gene selection for classification of microarray data based on the bayes error. *BMC Bioinformatics 8*(1), 370.

Zhang, X.-P., Z.-L. Wang, L. Tang, Y.-S. Sun, K. Cao, and Y. Gao (2011). Support vector machine model for diagnosis of lymph node metastasis in gastric cancer with multidetector computed tomography: a preliminary study. *BMC Cancer 11*(1), 10.

Appendix A

Java and R code for Copy Number Dataset

A.1 Java Code to Retrieve Characteristics Data for Chromosome Copy Number Dataset

```
import java.io.*; import java.net.*; import java.util.Arrays; public
class JavaGetUrl { public static void main(String[] args) throws
IOException {
   String record = new String();
   String[] URL = new String[190];

   URL[0]="http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSM288035";
   int i = 0;

   //BufferedWriter out = new BufferedWriter(new FileWriter("outfile.csv"));
```

```
   //String str = "";

FileWriter writer = new FileWriter("test.csv");
    writer.append("disease_state, Patient_ID, Age, Sex, Position, Pathology,
            Duke's_grade, Lymph_node_metastasis, MSI, Observation, Censor");
    writer.append('\n');
    while (i < 188) {

        String digit = URL[i].substring(62,68);

        int idigit = Integer.parseInt(digit);

        int idigit2 = idigit + 1;

        String sdigit2 = Integer.toString(idigit2);

        URL[i+1] = URL[i].substring(0,62) + sdigit2;

        record = Arrays.toString(ret(URL[i])).replace("[", "").replace("]", "");

        //System.out.println(record);

        writer.append(record);
        writer.append('\n');

        i++;
        }
         writer.flush();
         writer.close();
```

```
}

  @SuppressWarnings("deprecation")
  public static String[] ret(String url) {

      URL u;
      InputStream is = null;
      DataInputStream dis;
      String s;

      int[] idx= new int[22];

      String[] feature = new String[11];

      try{
          u = new URL(url);

          is = u.openStream();

          dis = new DataInputStream(new BufferedInputStream(is));

          while ((s = dis.readLine()) != null){
              if (s.contains("disease_state = ")){
                 //System.out.println(s);

                 idx[0] = s.indexOf("disease_state =") + 16;
                 idx[1] = s.indexOf("<br>Patient_ID");

                 idx[2] = s.indexOf("Patient_ID =") + 13;
                 idx[3] = s.indexOf("<br>Age");
```

```
idx[4] = s.indexOf("Age =") + 6;
idx[5] = s.indexOf("<br>Sex");

idx[6] = s.indexOf("Sex =") + 6;
idx[7] = s.indexOf("<br>Position");

idx[8] = s.indexOf("Position =") + 11;
idx[9] = s.indexOf("<br>Pathology");

idx[10] = s.indexOf("Pathology =") + 12;
idx[11] = s.indexOf("<br>Duke's_grade");

idx[12] = s.indexOf("Duke's_grade =") + 15;
idx[13] = s.indexOf("<br>Lymph_node_metastasis");

idx[14] = s.indexOf("Lymph_node_metastasis =") + 24;
idx[15] = s.indexOf("<br>MSI");

idx[16] = s.indexOf("MSI =") + 6;
idx[17] = s.indexOf("<br>Observation_(days)");

idx[18] = s.indexOf("Observation_(days) =") + 21 ;
idx[19] = s.indexOf("<br>Censor");

idx[20] = s.indexOf("Censor =") + 9;
idx[21] = s.indexOf("<br></td>");


feature[0] = s.substring(idx[0],idx[1]);
```

```
            feature[1] = s.substring(idx[2],idx[3]);
            feature[2] = s.substring(idx[4],idx[5]);
            feature[3] = s.substring(idx[6],idx[7]);
            feature[4] = s.substring(idx[8],idx[9]);
            feature[5] = s.substring(idx[10],idx[11]);
            feature[6] = s.substring(idx[12],idx[13]);
            feature[7] = s.substring(idx[14],idx[15]);
            feature[8] = s.substring(idx[16],idx[17]);
            feature[9] = s.substring(idx[18],idx[19]);
            feature[10] = s.substring(idx[20],idx[21]);


          }
        }

  //System.out.println(idx);

  int i;
  for (i=0; i < 11; i++) {
  //System.out.println(feature[i]);
  }

  }
catch (MalformedURLException mue){
  System.out.println("Ouch -a MalformedURLExeption happened.");
  mue.printStackTrace();
  System.exit(1);
}
catch (IOException ioe) {
```

```
            System.out.println("Oops- an IOException happend");
            ioe.printStackTrace();
           System.exit(1);
         } finally{

            try {
               is.close();
            } catch (IOException ioe) {

               }
         }
      return feature;
      }
   }
```

A.2 R Code for Chromosome Copy Number Dataset

A.2.1 Combine Individual Copy Number Files

```
setwd("C:\\Users\\plg519\\Documents\\Academic\\MS Report\\R Files")
i=35
filename = paste("C:\\Users\\plg519\\Documents\\Academic\\MS Report\\Output files\
\GSM2880",i,"NonSelf.txt",sep="")

x1 = read.table(filename, header=T)
cat("@RELATION CNV", "\n", file="CNVALL.arff", append=T)
```

```
nv=nrow(x1)
for (j in 1:nv){
cat(paste("@ATTRIBUTE ", x1[j,2], " NUMERIC", sep=""), "\n", file="CNVALL.arff",
append=T)
}

cha = read.csv("C:\\Users\\plg519\\Documents\\Academic\\MS Report\\test.csv")
Resp = character()
for (i in 1:nrow(cha)){
Resp =c(Resp, paste("",cha[i,6], cha[i,7],cha[i,8],cha[i,9],"") )
}

cat("@ATTRIBUTE Age NUMERIC",  "\n", file="CNVALL.arff", append=T)
cat("@ATTRIBUTE Sex {F,M}","\n", file="CNVALL.arff", append=T)
cat("@ATTRIBUTE Position {Left,Right,Rectum}","\n", file="CNVALL.arff", append=T)
cat("@ATTRIBUTE Observation NUMERIC", "\n", file="CNVALL.arff", append=T)
cat("@ATTRIBUTE Censor {Yes,No}", "\n", file="CNVALL.arff", append=T)


levels=unique(Resp)
waht=c( rep(",", length(levels)-1), "")
cat("@ATTRIBUTE class {",paste("\"",levels,"\"",   waht, sep=""),"}","\n",
file="CNVALL.arff", append=T)

cat("\n\n",file="CNVALL.arff", append=T)
cat("@DATA","\n", file="CNVALL.arff", append=T)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for (ii in 20:110) {
i=2*ii+1
j=2*ii+2
filenameC = paste("C:\\Users\\plg519\\Documents\\Academic\\MS Report\\Output files
\\GSM288", ifelse(ii<50, "0", ""),i,"NonSelf.txt",sep="")

filenameN = paste("C:\\Users\\plg519\\Documents\\Academic\\MS Report\\Output files
\\GSM288", ifelse(ii<49, "0", ""),j,"NonSelf.txt",sep="")

filenameC
filenameN

x1 = read.table(filenameC, header=T)
x2 = read.table(filenameN, header=T)

cat(c(x1[,6]-x2[,6]),",",sep=",", file="CNVALL.arff", append=T)

cat(cha[i-34,3],",",file="CNVALL.arff", append=T)

cat(as.character(unlist(cha[i-34, 4])),",", file="CNVALL.arff", append=T)
cat(as.character(unlist(cha[i-34, 5])),",", file="CNVALL.arff", append=T)
cat(cha[i-34,10],",", file="CNVALL.arff", append=T)
cat(as.character(cha[i-34,11]),",", file="CNVALL.arff", append=T)

cat( paste("\"", cha[i-34,6], cha[i-34,7],cha[i-34,8],cha[i-34,9],"\""), "\n",
file="CNVALL.arff", append=T)
}


setwd("C:\\Users\\plg519\\Documents\\Academic\\MS Report\\R Files")
```

```
dat=read.table("CNVALL_2.txt", sep=",")

dat3=read.table("CNV3.arff", sep=",",skip=58504)
dim(dat3)
n=ncol(dat)
dat3[,(n-6) :n]
ndat=rbind(dat3[,-((n-6) :n)],  dat[,-((n-6) :n)] )
chr=rbind(dat3[,(n-6) :n],  dat[,(n-6) :n] )

write.table(t(ndat),file="tdat.txt",row.names=F)
write.table(chr,file="character.txt",row.names=F)
```

A.2.2 Test for Interaction Functions

File "Heter.gamma.r" to be called in later functions

```
Heter.gamma<-function(data,a,b, mn, mcon, coln=5){
        N<-sum(mn)*b
        d1<-data[, 1]
    d2<-data[, 2]
    d3<-data[, 3]
    ranks<-data[, coln]
    Rij<-as.matrix(tapply(ranks, list(d1, d2), mean) )
           # matrix with \bar{R}<-{ij.} as the (i,j) element
    Rik<-as.matrix(tapply(ranks, list(d1, d3), mean) )
    # matrix with \bar{R}<-{i.k} as the (i,j) element  might have NA if unbalanced.
    Rik<-replace(Rik, is.na(Rik), 0)    # replace NA's in matrix Rik by 0.
    # note: after replace, the number of rows still correct, but the number
```

```
    #of columns
    # would be all same as max<-i n<-i instead of n<-i columns for the ith row.

    Ri<-apply(Rij, 1, mean)  # returns a vector (\wtR<-{1..}, ..., \wtR<-{a..})
    Rj<-apply(Rij, 2, mean)

    Rim<-kronecker(Ri, t(as.vector(rep(1,b))) )
      # a a by b matrix with all elements of the ith row same as \wtR<-{i..}
    Rjm<-kronecker(t(Rj), as.vector(rep(1,a)))
      # a a by b matrix with all elements of the ith column same as \wtR<-{.j.}


## calculate test statistics

    MSgamma<- sum((Rij-Rim-Rjm+mean(Ri) )^2 )/((a-1)*(b-1) )

      tmp1<-tapply(ranks^2, d1, sum)
       # returns a vector  with ith element
       #\sum<-{j=1}^b \sum<-{k=1}^{n<-i} R<-{ijk}^2
      tmp2<-sum( tmp1/(mn*(mn-1)) )
      # \sum<-{i,j, k}\frac{X<-{ijk}^2}{n<-i(n<-i-1)}


    MSEphi<- tmp2/(a*b)   - mean( apply(Rij^2, 1, mean)/(mn-1) )
    MSE<- MSEphi *b/(b-1) - mean(apply(Rik^2, 1, sum)/(mn*(mn-1)) ) *b/(b-1) +
    mean(Ri^2/(mn-1)) *b/(b-1)

## another way to calculate MSE
#   MSE<-  tmp2/(a*(b-1)) - mean(apply(Rik^2, 1, sum)/(mn*(mn-1)) ) *b/(b-1) -
         mean(apply((Rij-Rim)^2, 1, sum)/(mn-1) )/(b-1)
```

```
    Fgamma<-MSgamma/MSE
    Fphi<-MSphi/MSEphi
    Dgamma<-MSgamma-MSE
    Dphi<-MSphi-MSEphi
## and

        euijk<-apply(data, 1, e<-function(x) eu(x, coln, Rij, Rik))
    e<-euijk[1,]  # returns the e<-{ijk} as a vector, same as ranks structure
    u<-euijk[2,]  # returns the u<-{ijk} as a vector, same as ranks structure

    vars<-taufun(u, ranks, d1, d2, d3, a, b, mn, mcon, coln)



    #TSbeta<-sqrt(b)*(Fbeta-1)
    TSgamma<-sqrt(b)*(Fgamma-1)
    #TSphi<-sqrt(b)*(Fphi-1)
    #DSbeta<-sqrt(b)*Dbeta
    DSgamma<-sqrt(b)*Dgamma
    #DSphi<-sqrt(b)*Dphi
        if (coln==5) {
        #DSbeta<-DSbeta/N^2
    DSgamma<-DSgamma/N^2
        #DSphi<-DSphi/N^2
        }
```

```
    #varTSbeta<-vars$taubeta2/vars$EMSE^2
    varTSgamma<-vars$taugamma2/vars$EMSE^2
    #varTSphi<-vars$tauphi2/vars$EMSEphi^2




    #pbeta<- 2*(1-pnorm(abs( TSbeta/sqrt(varTSbeta) )))
    #pphi<- 2*(1-pnorm( abs(TSphi/sqrt(varTSphi) )))
    pgamma<- 2*(1- pnorm(abs(TSgamma/sqrt(varTSgamma) )) )

    #Dpbeta<- 2*(1-pnorm(abs( DSbeta/sqrt(vars$taubeta2) )))
    #Dpphi<- 2*(1-pnorm(abs( DSphi/sqrt(vars$tauphi2) )))
    #Dpgamma<- 2*(1- pnorm(abs(DSgamma/sqrt(vars$taugamma2))) )


       list(pgamma=pgamma)

       #results=c(pbeta, pgamma, pphi)
       #names(results)=c("palpha", "pbeta", "pgamma", "pphi")
}
```

File "faster.heter.gamma.r" to be called in later functions

```
taufunNew<-function(u, ranks, d1, d2, d3, a, b, mn, mcon, coln){
   R<-ranks
   usigmaijj1<-usigmaijj12<-array(rep(0, a*b*b), c(a, b, b))
```

```
usigma2<-0
us <-numeric()
for (i in 1:a){
  us[i]<-0
  trti.dat=t(matrix(u[(d1==i)], nrow=b))
   usigmaijj1[i,,]=cov(trti.dat)

  Rtrti.dat=t(matrix(R[(d1==i)], nrow=b))
  bb1=rep(1:b, b:1)
  b2=cbind((1:b), b)
  bb2=unlist(apply(b2,1, function(x) x[1]:x[2]))
  Rpairs=rbind(Rtrti.dat[, bb1], Rtrti.dat[, bb2])
  tempusig = apply(Rpairs, 2, sigijj12jacknew)

  indexUe= (bb1 != bb2)
  midd=cbind(c(bb1, bb2[indexUe]), c(bb2, bb1[indexUe]),
  c(tempusig, tempusig[indexUe] ) )

  usigmaijj12[i, ,]=matrix(midd[order(midd[,1], midd[,2]), 3],  nrow=b)
  usigma2=usigma2+sum(diag(usigmaijj1[i,,]))/mn[i]
  us[i]<-us[i]+ sum(diag(usigmaijj1[i,,]))/mn[i]

}

   usigma2<-usigma2/(a*b)
   EMSEphi<-usigma2

 #  ucsi1<-2*sum(apply(usigmaijj12, 1, sum) /(mn*(mn-1))  )/(a^2*b)
        # estimate of $\zeta<-1$ in thm 2.1
```

```
pucsi1<-2*apply(usigmaijj12 /(mn*(mn-1)), c(2, 3), sum)   /(a^2*b)
pucsi2<-2*((apply(usigmaijj1/mn, c(2, 3), sum) )^2 -
apply(usigmaijj1^2 / mn^2, c(2, 3), sum) )/(a^2*b)

ucsi1<-sum(pucsi1)  # estimate of $\zeta<-1$ in thm 2.1 using all sum
ucsi2<-sum(pucsi2)  #estimate of $\zeta<-2$ in thm 2.1. using all sum

psum<-apply(usigmaijj1/mn, c(2, 3), sum)

zeta1<-zeta2<-partsum<-numeric()
#mc<-c(1/4, 1/3, 2/5, 9/20)
mc<-mcon
ll<-0
for (l3 in mc[-length(mc)]){
  ll<-ll+1
  tu1<-tu2<-parts0<-0
 for (j0 in 1:b){
# for (j2 in seq(round(-b^l3 ), round(b^l3))){
for (j2 in seq(-l3, l3)){
          if ((j0+j2>0)& (j0+j2<=b)  )  {
        tu1<-tu1+pucsi1[j0, j0+j2]
        tu2<-tu2+pucsi2[j0, j0+j2]
        parts0<- parts0+ psum[j0, j0+j2]
           }

  }
  }
zeta1[ll]<-tu1
zeta2[ll]<-tu2
```

```
partsum[ll]<-parts0
}

 zeta1[length(mc)]<- ucsi1
 zeta2[length(mc)]<- ucsi2
 partsum[length(mc)]<- sum(usigmaijj1/mn)
 esigma2<- a*b*usigma2/(a*(b-1))- partsum/(a*b*(b-1))



 N<-sum(mn)*b


tauphi2<-taubeta2<- taugamma2<-numeric()

 tauphi2<-zeta1 + zeta2/(a-1)^2
 taubeta2<-zeta1 + zeta2
 taugamma2<-zeta1 + zeta2/(a-1)^2

  if (coln==5) {
      tauphi2<-tauphi2/N^4
      taubeta2<-taubeta2/N^4
      taugamma2<-taugamma2/N^4
      esigma2<-esigma2/N^2
      usigma2<-usigma2/N^2
     }
  tauphi2[(tauphi2 <=0)] <- 10^(-15)
  taubeta2[(taubeta2 <=0)] <-10^(-15)
  taugamma2[(taugamma2 <=0)] <-10^(-15)
```

```
      list( EMSE=esigma2, tauphi2=tauphi2, taubeta2=taubeta2, EMSEphi=usigma2,
      taugamma2=taugamma2, zeta1=zeta1, zeta2=zeta2)

      }
#dataformat2 function take argument data in format1 and convert to dataformat2
#x_{ijk}, k=1, ..., n_i are the kth observation from the ith subject at time j.
# 1 1 1 x111
# 1 1 2 x112
# 1 2 1 x121
# 1 2 2 x122
# 1 2 3 x123
#nofactor is the number of fixed factors (including time)
dataformat2= function(data, nofactor=2){ m=ncol(data)-nofactor
 y=c(t(data[,-(1:nofactor)]))
 Time=rep(1:m, nrow(data))
 sub=kronecker( data[,1],rep(1, m))
 mydat=numeric()
 for (i in 2:nofactor  ){
     temp=kronecker( data[,i],rep(1, m))
     mydat=cbind(mydat, temp )
     }
 resu=cbind(mydat, Time, sub, y)
 colnames(resu)=c(paste("class", 1:(i-1), sep=""), "SNP", "PatID", "y"  )
 resu
 }


Heter.gamma2<-function(data,a,b, mn, mcon, coln=5, Ca=cbind(as.vector
        (rep(1, a-1)), -diag(a-1)) ){
        N<-sum(mn)*b
```

```
    d1<-data[, 1]
  d2<-data[, 2]
  d3<-data[, 3]
  ranks<-data[, coln]
  Rij<-as.matrix(tapply(ranks, list(d1, d2), mean) )
        # matrix with \bar{R}<-{ij.} as the (i,j) element
  Rik<-as.matrix(tapply(ranks, list(d1, d3), mean) )
  # matrix with \bar{R}<-{i.k} as the (i,j) element  might have NA if
  # unbalanced.
  Rik<-replace(Rik, is.na(Rik), 0)    # replace NA's in matrix Rik by 0.
  # note: after replace, the number of rows still correct, but the number
  # of columns
  # would be all same as max<-i n<-i instead of n<-i columns for the ith row.

  Ri<-apply(Rij, 1, mean)  # returns a vector (\wtR<-{1..}, ..., \wtR<-{a..})
  Rj<-apply(Rij, 2, mean)

  Rim<-kronecker(Ri, t(as.vector(rep(1,b))) )
    # a a by b matrix with all elements of the ith row same as \wtR<-{i..}
  Rjm<-kronecker(t(Rj), as.vector(rep(1,a)))
    # a a by b matrix with all elements of the ith column same as \wtR<-{.j.}


## calculate test statistics

  MSbeta<- a* sum( (Rj-mean(Rj))^2 ) /(b-1)
  MSgamma<- sum((Rij-Rim-Rjm+mean(Ri) )^2 )/((a-1)*(b-1) )
  MSphi<- sum((Rij-Rjm)^2 )/((a-1)*b  )

    tmp1<-tapply(ranks^2, d1, sum)
```

```
  tmp2<-sum( tmp1/(mn*(mn-1)) )
   # \sum<-{i,j, k}\frac{X<-{ijk}^2}{n<-i(n<-i-1)}


MSEphi<- tmp2/(a*b)   - mean( apply(Rij^2, 1, mean)/(mn-1) )
MSE<- MSEphi *b/(b-1) - mean(apply(Rik^2, 1, sum)/(mn*(mn-1))
                ) *b/(b-1) + mean(Ri^2/(mn-1)) *b/(b-1)

Fbeta<-MSbeta/MSE
Fgamma<-MSgamma/MSE
Fphi<-MSphi/MSEphi


    euijk<-apply(data, 1, e<-function(x) eu(x, coln, Rij, Rik))
e<-euijk[1,] #returns the e<-{ijk} as a vector, same as ranks structure
u<-euijk[2,] #returns the u<-{ijk} as a vector, same as ranks structure

vars<-taufunNew(u, ranks, d1, d2, d3, a, b, mn, mcon, coln)

TSbeta<-sqrt(b)*(Fbeta-1)
TSgamma<-sqrt(b)*(Fgamma-1)
TSphi<-sqrt(b)*(Fphi-1)


varTSbeta<-vars$taubeta2/vars$EMSE^2
varTSgamma<-vars$taugamma2/vars$EMSE^2
varTSphi<-vars$tauphi2/vars$EMSEphi^2

pbeta<- 2*(1-pnorm(abs( TSbeta/sqrt(varTSbeta) )))
pphi<- 2*(1-pnorm( abs(TSphi/sqrt(varTSphi) )))
```

```
    pgamma<- 2*(1- pnorm(abs(TSgamma/sqrt(varTSgamma) )) )

    list(pgamma=pgamma)
}
```

File "function.s"

```
## store the data, they are stored in data like this: # 1 1 1 x111 #
1 1 2 x112 # 1 2 1 x121 # 1 2 2 x122 # 1 2 3 x123 # x<-{ijk}, k=1,
..., n<-{ij} are iid from distribution defined as a function distr #
The overall rank is calculated and put at the last colum # coln=4 or
5. if coln=4, test use org data ;  if coln=5, test use rank, 5 is
default. #  mn is a matrix with (i,j) element n<-{ij}


# eu is a function to calaculate residue x<-{ijk}-\bar{x}<-{ij.}
eu<-function(x, coln, Rij, Rik){
        d1<-x[ 1]
    d2<-x[ 2]
        d3<-x[ 3]
        R<-x[ coln]
        # e<- R-Rij[d1,d2]+Rik[d1, d3]-mean(Rij[d1,])
        u<- R-Rij[d1,d2]
        result<-rbind(u,u)   #used to be rbind(e,u)
        result
        }
# returns a vector with two elements. The first one is e<-{ijk} and
the second is u<-{ijk}
```

```
# in the following function, u, d1, d2 are all b*sum(mn) dimentional
vectors, same as the last # column of dat


#  Unbiased estimate of \sigma<-{ijj1}^2 # calculate the u-stat of
vectors x=(x<-1, x<-2, \cdots, x<-{ni})   y=(y<-1, y<-2, \cdots,
y<-{ni}), # where $X<-j$ and $Y<-j$ are correlated, but $X<-j$ and
$Y<-j1$ are indept if j \ne j1. # : \sum<-{k1 \ne k2 \ne k3 \ne k4}
(x<-{k1}-x<-{k2})}(y<-{k1}-y<-{k2)}) (x<-{k3}-x<-{k4)})
(y<-{k3}-y<-{k4)}) # unbiased est. of  4*ni*(ni-1)*(ni-2)*(ni-3)
[E(X<-{ijk}-\mu<-{ij} u<-{ij1k}) ]^2 #fun.sigijj12$sigmaijj12 will
give the unbiased est of $\sigma<-{ijj1}^2$ #fun.sigijj12$ssijj1
will give the unbiased est of $\sigma<-{ijj} \sigma<-{ij_1j_1}$.

fun.sigijj12<-function(x, y){
        ni<-length(x)
    sigmaijj12<- 0
        ssijj1<- 0
    for (m1 in 1:ni){
     for (m2 in 1:ni){
       for (m3 in 1:ni){
         for (m4 in 1:ni){
         flag<- (m1!=m2)&(m1 !=m3)&(m1 !=m4) &(m2 !=m3) &(m2 !=m4) & (m3 !=m4)
         sigmaijj12<-sigmaijj12+ (flag==T)* ( (x[m1]-x[m2])*(y[m1]-y[m2])*
         (x[m3]-x[m4])*(y[m3]-y[m4]) )
             ssijj1<-ssijj1+(flag==T)*  (x[m1]-x[m2])^2*(y[m3]-y[m4])^2
          }
        }
```

```
       }
      }
     sigmaijj12<-sigmaijj12/(4*ni*(ni-1)*(ni-2)*(ni-3) )
        ssijj1 <- ssijj1/(4*ni*(ni-1)*(ni-2)*(ni-3) )
     result<- list(sigmaijj12=sigmaijj12, ssijj1=ssijj1)
     result
}


### # tauphi2 have five components corresponds to partial sum # up
to b^c(1/4, 1/3, 2/5, 9/20)



taufun<-function(u, ranks, d1, d2, d3, a, b, mn, mcon, coln){
   R<-ranks
   usigmaijj1<-usigmaijj12<-array(rep(0, a*b*b), c(a, b, b))
   usigma2<-0
   us <-numeric()
   for (i in 1:a){
     us[i]<-0
    for (j in 1:b){
    for (j1 in 1:b){
    usigmaijj1[i, j, j1]<-sum(u[((d1==i)&(d2==j))] * u[((d1==i)&(d2==j1))])
     /(mn[i]-1)  # unbiased est. of $\sigma<-{ijj1}$

    x<-R[((d1==i)&(d2==j))]
    y<-R[((d1==i)&(d2==j1))]
    usigmaijj12[i, j, j1]<-sigijj12jack(x ,  y)
```

```
   }  # end of j1
   usigma2<-usigma2+usigmaijj1[i, j, j]/mn[i]
   us[i]<-us[i]+usigmaijj1[i, j, j]/mn[i]

 }

}

  usigma2<-usigma2/(a*b)
  EMSEphi<-usigma2

#  ucsi1<-2*sum(apply(usigmaijj12, 1, sum) /(mn*(mn-1))  )/(a^2*b)
       # estimate of $\zeta<-1$ in thm 2.1
#  tmpucsi2<-sum( (apply(usigmaijj1/mn, c(2, 3), sum) )^2 ) -
sum(usigmaijj1^2  / mn^2) #  ucsi2<-2*tmpucsi2/(a^2*b)  #estimate of
$\zeta<-2$ in thm 2.1.

pucsi1<-2*apply(usigmaijj12 /(mn*(mn-1)), c(2, 3), sum)   /(a^2*b)
pucsi2<-2*((apply(usigmaijj1/mn, c(2, 3), sum) )^2 -
apply(usigmaijj1^2  / mn^2, c(2, 3), sum) )/(a^2*b)
ucsi1<-sum(pucsi1)  # estimate of $\zeta<-1$ in thm 2.1 using all
sum ucsi2<-sum(pucsi2)  #estimate of $\zeta<-2$ in thm 2.1. using
all sum

psum<-apply(usigmaijj1/mn, c(2, 3), sum)

zeta1<-zeta2<-partsum<-numeric() #mc<-c(1/4, 1/3, 2/5, 9/20)
mc<-mcon ll<-0 for (l3 in mc[-length(mc)]){
```

```
  ll<-ll+1
  tu1<-tu2<-parts0<-0
 for (j0 in 1:b){
# for (j2 in seq(round(-b^l3 ), round(b^l3))){ for (j2 in seq(-l3,
l3)){
          if ((j0+j2>0)& (j0+j2<=b)  )  {
        tu1<-tu1+pucsi1[j0, j0+j2]
        tu2<-tu2+pucsi2[j0, j0+j2]
        parts0<- parts0+ psum[j0, j0+j2]
           }

  }
  }
zeta1[ll]<-tu1 zeta2[ll]<-tu2 partsum[ll]<-parts0 }

 zeta1[length(mc)]<- ucsi1
 zeta2[length(mc)]<- ucsi2
 partsum[length(mc)]<- sum(usigmaijj1/mn)
 esigma2<- a*b*usigma2/(a*(b-1))- partsum/(a*b*(b-1))



 N<-sum(mn)*b


tauphi2<-taubeta2<- taugamma2<-numeric()

 tauphi2<-zeta1 + zeta2/(a-1)^2
 taubeta2<-zeta1 + zeta2
 taugamma2<-zeta1 + zeta2/(a-1)^2
```

```
  if (coln==5) {
      tauphi2<-tauphi2/N^4
      taubeta2<-taubeta2/N^4
      taugamma2<-taugamma2/N^4
      esigma2<-esigma2/N^2
      usigma2<-usigma2/N^2
     }
  tauphi2[(tauphi2 <=0)] <- 10^(-15)
  taubeta2[(taubeta2 <=0)] <-10^(-15)
  taugamma2[(taugamma2 <=0)] <-10^(-15)

  list( EMSE=esigma2, tauphi2=tauphi2, taubeta2=taubeta2,
    EMSEphi=usigma2,taugamma2=taugamma2, zeta1=zeta1, zeta2=zeta2)
  }
```

```
# calculate the u-stat of vector x=(x<-1, x<-2, \cdots, x<-{nij})
where # X<-i are iid with variance \sigma^2. This u-stat will give #
unbiased estimate of  \sigma^4 # : \sum<-{k1 \ne k2 \ne k3 \ne k4}
(x<-{k1}-x<-{k2})}^2 (x<-{k3}-x<-{k4})}^2 # sigij4<-function(x){
        nij<-length(x)
    sigmaij4<- 0
    for (m1 in 1:nij){
     for (m2 in 1:nij){
       for (m3 in 1:nij){
         for (m4 in 1:nij){
         flag<- (m1!=m2)&(m1 !=m3)&(m1 !=m4) &(m2 !=m3) &(m2 !=m4) & (m3 !=m4)
```

```
          sigmaij4<-sigmaij4+ (flag==T)*(x[m1]-x[m2])^2 * (x[m3]-x[m4])^2
           }
         }
        }
       }
      sigmaij4<-sigmaij4/(4*nij*(nij-1)*(nij-2)*(nij-3) )
      sigmaij4
}


Heter.test<-function(data,a,b, mn, mcon, coln=5,
Ca=cbind(as.vector(rep(1, a-1)), -diag(a-1)) ){
        N<-sum(mn)*b
        d1<-data[, 1]
    d2<-data[, 2]
    d3<-data[, 3]
    ranks<-data[, coln]
    Rij<-as.matrix(tapply(ranks, list(d1, d2), mean) )
           # matrix with \bar{R}<-{ij.} as the (i,j) element
    Rik<-as.matrix(tapply(ranks, list(d1, d3), mean) )
    Rik<-replace(Rik, is.na(Rik), 0)    # replace NA's in matrix Rik by 0.

    Ri<-apply(Rij, 1, mean)
    Rj<-apply(Rij, 2, mean)

    Rim<-kronecker(Ri, t(as.vector(rep(1,b))) )
    # a a by b matrix with all elements of the ith row same as \wtR<-{i..}
    Rjm<-kronecker(t(Rj), as.vector(rep(1,a)))
    # a a by b matrix with all elements of the ith column same as \wtR<-{.j.}
```

```
## calculate test statistics

    MSbeta<- a* sum( (Rj-mean(Rj))^2 ) /(b-1)
    MSgamma<- sum((Rij-Rim-Rjm+mean(Ri) )^2 )/((a-1)*(b-1) )
    MSphi<- sum((Rij-Rjm)^2 )/((a-1)*b  )

      tmp1<-tapply(ranks^2, d1, sum)
      tmp2<-sum( tmp1/(mn*(mn-1)) )



    MSEphi<- tmp2/(a*b)   - mean( apply(Rij^2, 1, mean)/(mn-1) )
    MSE<- MSEphi *b/(b-1) - mean(apply(Rik^2, 1, sum)/(mn*(mn-1))
        ) *b/(b-1) + mean(Ri^2/(mn-1)) *b/(b-1)







        Fbeta<-MSbeta/MSE
    Fgamma<-MSgamma/MSE
    Fphi<-MSphi/MSEphi
    Dbeta<-MSbeta-MSE
    Dgamma<-MSgamma-MSE
    Dphi<-MSphi-MSEphi
## and

        euijk<-apply(data, 1, e<-function(x) eu(x, coln, Rij, Rik))
    e<-euijk[1,]  # returns the e<-{ijk} as a vector, same as ranks structure
    u<-euijk[2,]  # returns the u<-{ijk} as a vector, same as ranks structure
```

```
vars<-taufun(u, ranks, d1, d2, d3, a, b, mn, mcon, coln)



TSbeta<-sqrt(b)*(Fbeta-1)
TSgamma<-sqrt(b)*(Fgamma-1)
TSphi<-sqrt(b)*(Fphi-1)
DSbeta<-sqrt(b)*Dbeta
DSgamma<-sqrt(b)*Dgamma
DSphi<-sqrt(b)*Dphi
    if (coln==5) {
    DSbeta<-DSbeta/N^2
DSgamma<-DSgamma/N^2
    DSphi<-DSphi/N^2
    }

varTSbeta<-vars$taubeta2/vars$EMSE^2
varTSgamma<-vars$taugamma2/vars$EMSE^2
varTSphi<-vars$tauphi2/vars$EMSEphi^2




pbeta<- 2*(1-pnorm(abs( TSbeta/sqrt(varTSbeta) )))
pphi<- 2*(1-pnorm( abs(TSphi/sqrt(varTSphi) )))
pgamma<- 2*(1- pnorm(abs(TSgamma/sqrt(varTSgamma) )) )

Dpbeta<- 2*(1-pnorm(abs( DSbeta/sqrt(vars$taubeta2) )))
Dpphi<- 2*(1-pnorm(abs( DSphi/sqrt(vars$tauphi2) )))
```

```
    Dpgamma<- 2*(1- pnorm(abs(DSgamma/sqrt(vars$taugamma2))) )



# test for main group effect
        #Ca<-cbind(as.vector(rep(1, a-1)), -diag(a-1))
        N<-sum(mn)*b
        etai<-matrix(0, length(mcon), a)
        ll<-0
        for (l in mcon){
        ll<-ll+1
        for (i in 1:a){
         for (j in 1:b){
#          for (j1 in seq(round(-b^l ), round(b^l))){
       for (j1 in seq(-l, l) ){
       if ((j+j1>0)& (j+j1<=b)  )  {
          tmpinc<- sum((ranks[((d1==i)&(d2==j))] -Rij[i, j])*(ranks
          [((d1==i)&(d2==(j+j1)))] -Rij[i, (j+j1)]))*sum(mn)/(b*mn[i]*(mn[i]-1))
          etai[ll, i]<-etai[ll, i]+tmpinc
          }
         } #end of j1
         } #end of j
        }  #end of i
      } # end of l
        Ri<-as.vector(Ri)
       TSalphastat<-function(etaii, Ca, N, Ri) N * t(Ri)%*% t(Ca)%*%solve(
       Ca%*% diag(etaii)%*% t(Ca) ) %*% Ca %*% Ri
       TSalpha<- apply(etai, 1, TSalpha<-function(etaii) {TSalphastat(etaii, Ca,
        N, Ri) }  )
```

```
       palpha<-1-pchisq(TSalpha, nrow(Ca))
       palpha1<- palpha
       results=c(palpha, pbeta, pgamma, pphi)
       names(results)=c("palpha", "pbeta", "pgamma", "pphi")
 list(TSalpha=TSalpha, Dpalpha=palpha1, Dpbeta=Dpbeta, Dpgamma=Dpgamma,
 Dpphi=Dpphi, palpha=palpha1, pbeta=pbeta, pgamma=pgamma, pphi=pphi,
 results=results)
}



sigijk4<-function(x){
        nijk<-length(x)
    sigmaijk4<- 0
    for (m1 in 1:nijk){
     for (m2 in 1:nijk){
       for (m3 in 1:nijk){
         for (m4 in 1:nijk){
         flag<- (m1!=m2)&(m1 !=m3)&(m1 !=m4) &(m2 !=m3) &(m2 !=m4) & (m3 !=m4)
         sigmaijk4<-sigmaijk4+ (flag==T)*(x[m1]-x[m2])^2 * (x[m3]-x[m4])^2
          }
        }
       }
      }
     sigmaijk4<-sigmaijk4/(4*nijk*(nijk-1)*(nijk-2)*(nijk-3) )
     sigmaijk4
}

thetahat<-function(x) (mean((x-mean(x))^2 ))^2
```

```
#thetahatijj1<-function(x, y) (mean((x-mean(x))*(y-mean(y))))^2
thetahatijj1<-function(x, y) (cov(x,y))^2



sigijk4jack<-function(x){
    n<-length(x)
    s4hat<- thetahat(x)
    result <- n *s4hat
        for (i in 1:n) result<-result- (n-1)/n* thetahat(x[-i])
        result
        }

sigijk4boot<-function(x){
    n<-length(x)
    thetahatstar<-mean(apply(matrix(sample(x, 3*1000, replace = T), 1000, 3)
    , 1, thetahat ) )
    result<-2*thetahat(x)-thetahatstar
    result
    }



## Jackknife estimate of \sigma_{ijj'}^2 ## x=(X_{ij1}, \cdots,
X_{ijn_i}), y=(X_{ij'1}, \cdots, X_{ij'n_i}),

sigijj12jack<-function(x, y){
    n<-length(x)
    s4hat<- thetahatijj1(x, y)
    result <- n *s4hat
        for (i in 1:n) result<-result- (n-1)/n* thetahatijj1(x[-i], y[-i])
        result
```

```
        }

sigijj12jacknew=function(z) {n=length(z);
sigijj12jack(unlist(z[1:(n/2)]), unlist(z[-(1:(n/2))]))}

### test for contrast effect

tcontrast<-function(data,a,b, mn, mcon, coln=5,
Ca=cbind(as.vector(rep(1, a-1)), -diag(a-1)) ){
        N<-sum(mn)*b
    d1<-data[, 1]
    d2<-data[, 2]
    d3<-data[, 3]
    ranks<-data[, coln]
    Rij<-as.matrix(tapply(ranks, list(d1, d2), mean) )
           # matrix with \bar{R}<-{ij.} as the (i,j) element
    Ri<-apply(Rij, 1, mean)
       N<-sum(mn)*b
        etai<-matrix(0, length(mcon), a)
        ll<-0
        for (l in mcon){
        ll<-ll+1
        for (i in 1:a){
         for (j in 1:b){
#          for (j1 in seq(round(-b^l ), round(b^l))){
       for (j1 in seq(-l, l) ){
          if ((j+j1>0)& (j+j1<=b)  )  {
            tmpinc<- sum((ranks[((d1==i)&(d2==j))] -Rij[i, j])
            *(ranks[((d1==i)&(d2==(j+j1)))] -Rij[i, (j+j1)]))
            *sum(mn)/(b*mn[i]*(mn[i]-1))
```

```
          etai[ll, i]<-etai[ll, i]+tmpinc
          }
         } #end of j1
         } #end of j
        }  #end of i
      } # end of l
        Ri<-as.vector(Ri)
       TSalphastat<-function(etaii, Ca, N, Ri) N * t(Ri)%*% t(Ca)
       %*%solve( Ca%*% diag(etaii)%*% t(Ca) ) %*% Ca %*% Ri
       TSalpha<- apply(etai, 1, TSalpha<-function(etaii)
       {TSalphastat(etaii, Ca, N, Ri) }  )
       palpha<-1-pchisq(TSalpha, nrow(Ca))
       palpha1<- palpha
       list(TSalpha=TSalpha, Dpalpha=palpha1)
}

#format1 # sub trt      time1      time2      time3      time4
time5 #   1   1  2.4644642  1.7233498 -1.1374695 -0.5242729
-2.379145 #   2   1  2.5746848  1.0181738 -0.8325308 -2.4873067
-3.463602 #   3   1  2.5813995 -0.7528324 -3.1457645 -3.3135573
-4.364621 #   4   1  0.8232141  0.2394987 -2.2073150 -3.3583005
-6.073399 #   5   1  0.8274860  0.8323298 -2.1028060 -2.6015848
-3.291307 #   1   2 -2.2217084  0.6779049  3.6310542  3.2052691
4.310316 #   2   2 -3.3954705 -0.7827040  3.1364749  3.7184895
5.118996



#dataformat2 function take argument data in format1 and convert to
```

```
dataformat2 #### x_{ijk}, k=1, ..., n_i are the kth observation from
the ith subject at time j. # 1 1 1 x111 # 1 1 2 x112 # 1 2 1 x121 #
1 2 2 x122 # 1 2 3 x123 #nofactor is the number of fixed factors
(including time) dataformat2= function(data, nofactor=2){
m=ncol(data)-nofactor
 y=c(t(data[,-(1:nofactor)]))
 Time=rep(1:m, nrow(data))
 sub=kronecker( data[,1],rep(1, m))
 mydat=numeric()
 for (i in 2:nofactor  ){
     temp=kronecker( data[,i],rep(1, m))
     mydat=cbind(mydat, temp )
     }
 resu=cbind(mydat, Time, sub, y)
 colnames(resu)=c(paste("class", 1:(i-1), sep=""), "SNP", "PatID", "y"  )
 resu
 }


 Heter.gamma<-function(data,a,b, mn, mcon, coln=5){
        N<-sum(mn)*b
        d1<-data[, 1]
    d2<-data[, 2]
    d3<-data[, 3]
    ranks<-data[, coln]
    Rij<-as.matrix(tapply(ranks, list(d1, d2), mean) )
           # matrix with \bar{R}<-{ij.} as the (i,j) element
    Rik<-as.matrix(tapply(ranks, list(d1, d3), mean) )
    Rik<-replace(Rik, is.na(Rik), 0)    # replace NA's in matrix Rik by 0.
```

```
    Ri<-apply(Rij, 1, mean)  # returns a vector (\wtR<-{1..}, ..., \wtR<-{a..})
    Rj<-apply(Rij, 2, mean)

    Rim<-kronecker(Ri, t(as.vector(rep(1,b))) )
      # a a by b matrix with all elements of the ith row same as \wtR<-{i..}
    Rjm<-kronecker(t(Rj), as.vector(rep(1,a)))
      # a a by b matrix with all elements of the ith column same as \wtR<-{.j.}


## calculate test statistics

    MSgamma<- sum((Rij-Rim-Rjm+mean(Ri) )^2 )/((a-1)*(b-1) )

      tmp1<-tapply(ranks^2, d1, sum)
      tmp2<-sum( tmp1/(mn*(mn-1)) )


    MSEphi<- tmp2/(a*b)   - mean( apply(Rij^2, 1, mean)/(mn-1) )
    MSE<- MSEphi *b/(b-1) - mean(apply(Rik^2, 1, sum)/(mn*(mn-1)) )
    *b/(b-1) + mean(Ri^2/(mn-1)) *b/(b-1)
## another way to calculate MSE #   MSE<-  tmp2/(a*(b-1)) -
mean(apply(Rik^2, 1, sum)/(mn*(mn-1)) ) *b/(b-1) -
mean(apply((Rij-Rim)^2, 1, sum)/(mn-1) )/(b-1)



    Fgamma<-MSgamma/MSE
    Fphi<-MSphi/MSEphi
    Dgamma<-MSgamma-MSE
```

```
    Dphi<-MSphi-MSEphi
## and

        euijk<-apply(data, 1, e<-function(x) eu(x, coln, Rij, Rik))
    e<-euijk[1,]  # returns the e<-{ijk} as a vector, same as ranks structure
    u<-euijk[2,]  # returns the u<-{ijk} as a vector, same as ranks structure

    vars<-taufun(u, ranks, d1, d2, d3, a, b, mn, mcon, coln)



    #TSbeta<-sqrt(b)*(Fbeta-1)
    TSgamma<-sqrt(b)*(Fgamma-1)
    #TSphi<-sqrt(b)*(Fphi-1)
    #DSbeta<-sqrt(b)*Dbeta
    DSgamma<-sqrt(b)*Dgamma
    #DSphi<-sqrt(b)*Dphi
        if (coln==5) {
        #DSbeta<-DSbeta/N^2
    DSgamma<-DSgamma/N^2
        #DSphi<-DSphi/N^2
        }

    #varTSbeta<-vars$taubeta2/vars$EMSE^2
    varTSgamma<-vars$taugamma2/vars$EMSE^2
    #varTSphi<-vars$tauphi2/vars$EMSEphi^2
```

```
    #pbeta<- 2*(1-pnorm(abs( TSbeta/sqrt(varTSbeta) )))
    #pphi<- 2*(1-pnorm( abs(TSphi/sqrt(varTSphi) )))
    pgamma<- 2*(1- pnorm(abs(TSgamma/sqrt(varTSgamma) )) )

    #Dpbeta<- 2*(1-pnorm(abs( DSbeta/sqrt(vars$taubeta2) )))
    #Dpphi<- 2*(1-pnorm(abs( DSphi/sqrt(vars$tauphi2) )))
    #Dpgamma<- 2*(1- pnorm(abs(DSgamma/sqrt(vars$taugamma2))) )


       list(pgamma=pgamma)

       #results=c(pbeta, pgamma, pphi)
       #names(results)=c("palpha", "pbeta", "pgamma", "pphi")
 #list(TSalpha=TSalpha, Dpalpha=palpha1, Dpbeta=Dpbeta, Dpgamma=Dpgamma
 , Dpphi=Dpphi, palpha=palpha1, pbeta=pbeta, pgamma=pgamma, pphi=pphi,
  results=results)
}



# LOOCV with libsvm # The training data set Yu has a column called
response # Other columns of Yu are feature variables svm.test=
function(Yu, testuse, crossv=10){

 library(e1071)
       obj = tune(svm, response~., data = Yu,
              ranges = list(gamma = 2^(-3:1), cost = 2^(2:4)),
               tunecontrol = tune.control(cross=crossv )
              )
       bestGamma = obj$best.parameters[[1]]
```

```
       bestC = obj$best.parameters[[2]]

#Build model using SVM
 svm1=svm(response~., data=Yu, cost=bestC, gamma=bestGamma )

#Compute the predicted class for test data
totest=data.frame(testuse[,-ncol(testuse)])
colnames(totest)=colnames(Yu)[-ncol(testuse)]
 pred <- as.character(predict(svm1, totest) )
 pred

#Construct confusious matrix (table)
 confuTable=table(pred,testuse$response)

#Compute accuracy
 accuracy=sum(diag(confuTable))/(sum(confuTable))
 accuracy

}
```

A.2.3 Test for Interaction Effect for Each Block

```
#setwd("C:\\Users\\plg519\\Documents\\Academic\\MS Report\\R Files")

source("functions.s")
source("faster.heter.gamma.r")
```

```
#Name the snaps data (i.e. a 58494(snaps)*94(paris of patiances) matrix) "dat"
dat=read.table("tdat.txt",header=T)


#Name Physical characteristic data "cha"
cha=read.csv("test.csv")


#Extract the column of Duke stage from characteristic data, call it "Duke"
#One Duke Stage from each pair since each pair have same stages
Duke=as.character(cha[(1:94)*2-1,7])


#NDuke=1*(Duke==" A")+2*(Duke==" B")+3*(Duke==" C")+4*(Duke==" D")
#order(NDuke)
#table(NDuke)

#Convert Duke Stages to numerical values (B=1, C=2). Only use B and C here first
NDuke=1*(Duke==" B")+2*(Duke==" C")

#Get a sequence of positions corresponding to "B" stage, call it Blist
Blist=seq(length(NDuke))[Duke==" B"]
#Get a sequence of positions corresponding to "C" stage, call it Clist
Clist=seq(length(NDuke))[Duke==" C"]

n=8
#Get the length of Blist (i.e. count the number of "B" stages) (=46)
```

```
nB=length(Blist)
#Get the length of Clist (i.e. count the number of "C" stages)(=37)
nC=length(Clist)
bp=sample(1:nB,nB); cp=sample(1:nC,nC)

step0=100;alpha=0.1
###############################################
fold=10; pv.all=numeric();sigblocks.all=vector("list",fold)
accuracy=numeric()
for (group in 1:fold){



    ngb=trunc(nB/fold);ngc=trunc(nC/fold)
    testpo=c(((1:ngb)+ngb*(group-1)),(1:ngc)+ngc*(group-1))
    testposition=c(Blist[testpo[1:ngb]],Clist[testpo[ngb+(1:ngc)]])
    trainposition=c(Blist[-testpo[1:ngb]],Clist[-testpo[ngb+(1:ngc)]])
    testclass=NDuke[testposition]
    trainclass=NDuke[trainposition]

    total=0;pv=numeric();nblock=0
    eventotal=(trunc(nrow(dat)/step0)-1)*step0
    while (total<nrow(dat)){
    #57290){
            nblock=nblock+1

            step=ifelse(total<eventotal, step0,nrow(dat)-total)
            testX=dat[total+(1:step),testposition]
            trainX=dat[total+(1:step),trainposition]
            patID=c(1:(nB-ngb),1:(nC-ngc)  )
            traindata=data.frame(patID,trainclass,t(trainX))
```

```
            traindataformat1=dataformat2(traindata)
            ranks=rank(traindataformat1[,4])
            mydat=cbind(traindataformat1, ranks)
            org=Heter.gamma2(mydat, 2, step, mn=c((nB-ngb),(nC-ngc)), 5)
            pv=c(pv,org$pgamma)
            total=total+step
                    cat(c(group,nblock,org$pgamma),"\n",file=paste(
                    "pv.for", group, ".txt", sep=""),append=T)
            } #end of while loop
    }
```

A.2.4 Detection of Significant Blocks

```
#####################
#test for interaction effect for each block of size 100 SNPs
#Boffroni or FDR did not find any significance
#We did not do SVM.
#Instead, significances for the blocks was selected based just on
#alpha=0.05 for each of the fold training data
#Then the significant blocks were selected as those that are
#significant for all folds


#setwd("C:\\Users\\plg519\\Documents\\Academic\\MS Report\\R Files")

source("functions.s")
source("faster.heter.gamma.r")
```

```
#Name the snaps data (i.e. a 58494(snaps)*94(paris of patiances) matrix) "dat"
dat=read.table("tdat.txt",header=T)


#Name Physical characteristic data "cha"
cha=read.csv("test.csv")


#Extract the column of Duke stage from characteristic data, call it "Duke"
#One Duke Stage from each pair since each pair have same stages
Duke=as.character(cha[(1:94)*2-1,7])


#NDuke=1*(Duke==" A")+2*(Duke==" B")+3*(Duke==" C")+4*(Duke==" D")
#order(NDuke)
#table(NDuke)

#Convert Duke Stages to numerical values (B=1, C=2). Only use B and C here first
NDuke=1*(Duke==" B")+2*(Duke==" C")

#Get a sequence of positions corresponding to "B" stage, call it Blist
Blist=seq(length(NDuke))[Duke==" B"]
#Get a sequence of positions corresponding to "C" stage, call it Clist
Clist=seq(length(NDuke))[Duke==" C"]

n=8
#Get the length of Blist (i.e. count the number of "B" stages) (=46)
```

```
nB=length(Blist)
#Get the length of Clist (i.e. count the number of "C" stages)(=37)
nC=length(Clist)
#bp=sample(1:nB,nB); cp=sample(1:nC,nC)

step0=100;alpha=0.05
##############################################
fold=10; #pv.all=numeric();
sigblocks.all=vector("list",fold)
accuracy=numeric()
for (group in 1:fold){


    ngb=trunc(nB/fold);ngc=trunc(nC/fold)
    testpo=c(((1:ngb)+ngb*(group-1)),(1:ngc)+ngc*(group-1))
    testposition=c(Blist[testpo[1:ngb]],Clist[testpo[ngb+(1:ngc)]])
    trainposition=c(Blist[-testpo[1:ngb]],Clist[-testpo[ngb+(1:ngc)]])
    testclass=NDuke[testposition]
    trainclass=NDuke[trainposition]

    #total=0;pv=numeric();
    nblock=trunc(nrow(dat)/step0)
    #eventotal=(trunc(nrow(dat)/step0)-1)*step0

    pv=read.table(paste("pv.for", group, ".txt", sep="") )[,3]

    #pv.all=cbind(pv.all,pv)
    lengthpv=length(pv)
    #pv[order(pv)] <(1:lengthpv)/lengthpv*alpha
    # Bonferroni correction
```

```
   #sigblocks=(1:nblock)[pv[order(pv)] <(1:lengthpv)/lengthpv*alpha]
   #FDR control

   sigblocks=(1:nblock)[pv<alpha]
    # no multiple-comparison adjustment
    # Then use common significant blocks from all 'group's.
           ##################################################
   sigblocks.all[[group]]=sigblocks

           if (length(sigblocks)>0){
               #need to use svm to build model using the snps found.
               #Then test on the test data and report accuracy
               step=ifelse(max(sigblocks)<nblock, step0, nrow(dat)-nblock
               *step0)

               sigSNP.id=c( t(kronecker(matrix(1:step,nrow=1), matrix(
               1, nrow=length(sigblocks) ))+(sigblocks-1)*step))

# need to select representative features from each block and filter
# out other variables that are correlated with
# the representative features.
               trainuse=data.frame(t(dat[sigSNP.id, trainposition])
               , response=as.factor(trainclass))

               testuse=data.frame(t(dat[sigSNP.id,testposition])
               ,response=as.factor(testclass))

               #testX has dimension 16*40
               testX=data.frame(t(dat[sigSNP.id,testposition]))
```

```
                ######################
                #Use SVM
                #accuracy=c(accuracy,svm.test(trainuse,testuse))

    }
  # write.table(pv,file=paste("pv.for.fold", group, ".txt", sep=""))

} #end of group loop

#write.table(pv.all,file="pv.txt",row.names=F)
write.table(accuracy,file="accuracy.txt")
cat("*significant blocks\n",file="blocks.txt", append=F)
for(k in 1:fold) {
 cat(paste("fold ", k,sep=""), unlist(sigblocks.all[[k]]), "\n",
 file="blocks.txt", append=T)
}
```

A.2.5 Calculation of Delete-One p-values

```
#######################################
#### For each SNP in block 115, to find the significant SNPs within block 115.
##Each SNP was deleted from the blcok at a time, the p-value for testing  of
#the interaction effect with rest of 99
#### SNPs was calculated for each of the 10 folds.
```

```
####


#setwd("C:\\Users\\plg519\\Documents\\Academic\\MS Report\\R Files")

source("functions.s")
source("faster.heter.gamma.r")



#Name the snaps data (i.e. a 58494(snaps)*94(paris of patiances) matrix) "dat"
dat=read.table("tdat.txt",header=T)


#Name Physical characteristic data "cha"
cha=read.csv("test.csv")


#Extract the column of Duke stage from characteristic data, call it "Duke"
#One Duke Stage from each pair since each pair have same stages
Duke=as.character(cha[(1:94)*2-1,7])


#NDuke=1*(Duke==" A")+2*(Duke==" B")+3*(Duke==" C")+4*(Duke==" D")
#order(NDuke)
#table(NDuke)

#Convert Duke Stages to numerical values (B=1, C=2). Only use B and C
#here first
NDuke=1*(Duke==" B")+2*(Duke==" C")
```

```
#Get a sequence of positions corresponding to "B" stage, call it Blist
Blist=seq(length(NDuke))[Duke==" B"]
#Get a sequence of positions corresponding to "C" stage, call it Clist
Clist=seq(length(NDuke))[Duke==" C"]

n=8
#Get the length of Blist (i.e. count the number of "B" stages) (=46)
nB=length(Blist)
#Get the length of Clist (i.e. count the number of "C" stages)(=37)
nC=length(Clist)
#bp=sample(1:nB,nB); cp=sample(1:nC,nC)

step0=100;alpha=0.05
##############################################
fold=10; pv.all=NULL;
sigblocks.all=vector("list",fold)
accuracy=numeric()
for (group in 1:fold){
    #total=0;pv=numeric();
    nblock=trunc(nrow(dat)/step0)
    #eventotal=(trunc(nrow(dat)/step0)-1)*step0

    pv=read.table(paste("pv.for", group, ".txt", sep="") )[,3]

    pv.all=cbind(pv.all,pv)
    lengthpv=length(pv)
    #pv[order(pv)] <(1:lengthpv)/lengthpv*alpha
    # Bonferroni correction
```

```
    #sigblocks=(1:nblock)[pv[order(pv)] <(1:lengthpv)/lengthpv*alpha]
    #FDR control

    sigblocks=(1:nblock)[pv<alpha]
     # no multiple-comparison adjustment
     # Then use common significant blocks from all 'group's.
            ###################################################
    sigblocks.all[[group]]=sigblocks

}

T.orF=logical()
uniquesigs=unique(unlist(sigblocks.all))
for (h in 1:length(uniquesigs)){
     T.orF=c(T.orF, all(unlist(lapply(sigblocks.all, function(x)
      uniquesigs[h] %in% x))) )
}
sig.in.all=uniquesigs[T.orF]

 # need to select representative features from each block and filter
 #out other variables that are correlated with
 # the representative features.
    #   cat("",file="split.pv.for.group.txt",append=F)

# The next for loop code tests if left or right half of the blocks
#are significant.
########## not helpful.
# for ( hsig in 1:length(sig.in.all) ){
    # step=ifelse(sig.in.all[hsig]<nblock, step0, nrow(dat)-nblock*step0)
    # sigSNP.id=c( t(kronecker(matrix(1:step,nrow=1), matrix(1, nrow=
```

```
#length(sig.in.all[hsig]) ))+(sig.in.all[hsig]-1)*step))

# for (group in 1:fold){
# ngb=trunc(nB/fold);ngc=trunc(nC/fold)
# testpo=c(((1:ngb)+ngb*(group-1)),(1:ngc)+ngc*(group-1))
# testposition=c(Blist[testpo[1:ngb]],Clist[testpo[ngb+(1:ngc)]])
# trainposition=c(Blist[-testpo[1:ngb]],Clist[-testpo[ngb+(1:ngc)]])
# testclass=NDuke[testposition]
# trainclass=NDuke[trainposition]
# pvsplit=numeric()
# for (split in 1:2){
    # testX=dat[sigSNP.id[(split-1)*50+1:(step/2)],testposition]
    # trainX=dat[sigSNP.id[(split-1)*50+1:(step/2)],trainposition]

        # patID=c(1:(nB-ngb),1:(nC-ngc)  )
        # traindata=data.frame(patID,trainclass,t(trainX))

        # traindataformat1=dataformat2(traindata)
        # ranks=rank(traindataformat1[,4])
        # mydat=cbind(traindataformat1, ranks)
        # org=Heter.gamma2(mydat, 2, step/2, mn=c((nB-ngb),(nC-ngc)), 5)
        # pvsplit=c(pvsplit,org$pgamma)
# }   #end of split loop
   # cat(c(group,sig.in.all[hsig],pvsplit),"\n",file=
   "split.pv.for.group.txt",append=T)

# }
```

```
########################3 do delete one SNP at a time and recalculate p-values
group=group0

#for (group in 1:fold){
#pv.delete1=numeric()
for ( hsig in 1:length(sig.in.all) ){
    step=ifelse(sig.in.all[hsig]<nblock, step0, nrow(dat)-nblock*step0)
    sigSNP.id.block=c( t(kronecker(matrix(1:step,nrow=1), matrix(1, nrow
    =length(sig.in.all[hsig]) ))+(sig.in.all[hsig]-1)*step))

       pv.100=numeric()
       for ( jjj in 1:step){
          sigSNP.id= sigSNP.id.block[-jjj]

        ngb=trunc(nB/fold);ngc=trunc(nC/fold)
        testpo=c(((1:ngb)+ngb*(group-1)),(1:ngc)+ngc*(group-1))
        testposition=c(Blist[testpo[1:ngb]],Clist[testpo[ngb+(1:ngc)]])
        trainposition=c(Blist[-testpo[1:ngb]],Clist[-testpo[ngb+(1:ngc)]])
        testclass=NDuke[testposition]
        trainclass=NDuke[trainposition]
                testX=dat[sigSNP.id,testposition]
        trainX=dat[sigSNP.id,trainposition]
        patID=c(1:(nB-ngb),1:(nC-ngc)  )
        traindata=data.frame(patID,trainclass,t(trainX))

        traindataformat1=dataformat2(traindata)
        ranks=rank(traindataformat1[,4])
        mydat=cbind(traindataformat1, ranks)
        org=Heter.gamma2(mydat, 2, step-1, mn=c((nB-ngb),(nC-ngc)), 5)
```

```
        pv.100=c(pv.100,org$pgamma)
        cat(c(group,sig.in.all[hsig], jjj,pv.all[sig.in.all[hsig],group],
        org$pgamma),"\n",file=paste("delete1.pv.for.group", group,
                ".txt", sep=""),append=T)

      }
    cat(c(group,sig.in.all[hsig], jjj,pv.all[sig.in.all[hsig],group], pv.100)
    ,"\n",file="delete1.pv.txt",append=T)
  #  pv.delete1=rbind(pv.delete1, pv.100)

 }
```

A.2.6 Identification of Significant SNPs within Block 115

```
#########################
####  After observing the some of differences of p-values between using
##100 SNPs and 99 SNPs seem small.  To find a criterion for selecting
### the significant SNPs within block 115.
####  The ratio of p-values between using 100 SNPs and 99 SNPs were
#calcuated. Then the ratios across 10 folds for each SNP serves as
#a random sample, and the t-test was conducted
####  to select those SNPs with the mean ratio significant greater
#than 1. 15 SNPs were selected.

#### After finding the 15 significant SNPs within block 115
#### Excluding these 15 SNPs, the remaining 85 SNPs were tested for
#each of the 10 folds.
#### The resutls show that non of them are significant
```

```
####




nblock
all.g=NULL
reverse.rank=NULL


for (i in 1:10) {
#get all 100 p-values from 10 fold
pi=read.table(paste("delete1.pv.for.group",i,".txt", sep=""))
#construct the matrix including p-values and corresponding ratios
#with/without one SNP
gi=as.matrix(cbind(pi[,c(1,3:5)], pi[,5]/pi[,4]))
all.g=cbind(all.g,gi)
reverse.rank=cbind(reverse.rank, 101-rank(pi[,5]/pi[,4]) )
}

idx=order(apply(reverse.rank,1,mean))
SNPid=(1:100)[idx]
ratio=round(all.g[idx,5*(1:10)],2)
colnames(ratio)=paste("G", 1:10,"AovB", sep="")
row.names(ratio)=SNPid


###  For each SNP in this block, calculate a 99% confidence lower
#bound to see if the mean is > 1.
myT=function(x) unlist( t.test(x, alternative = "greater", mu = 1,
```

```
conf.level = 0.99)[3:5])

t.test.result=t(apply(ratio,1, myT))
#myT(ratio[1,])
t.test.result[,1]<0.01

sig.SNPs=subset(t.test.result, t.test.result[,1]<0.01)
SNP.num=as.numeric(row.names(sig.SNPs))

### Exclude these 15 significant SNPs from 115th blcok(has 100 SNPs)
#and test to see whether
### the remaining SNPs are still significant

dat=read.table("tdat.txt",header=T);
step0=100
nblock=trunc(nrow(dat)/step0)
#nblock =584
 step=ifelse(pi[1,2]<nblock, step0, nrow(dat)-nblock*step0)
 SNP.id.block=c( t(kronecker(matrix(1:step,nrow=1), matrix(1, nrow=
 length(pi[1,2]) ))+(pi[1,2]-1)*step))

  sigSNP.id=SNP.id.block[-SNP.num]

##########get the training data for above SNP ids and do one test
  #Name Physical characteristic data "cha"
  cha=read.csv("test.csv")
  #Extract the column of Duke stage from characteristic data, call it "Duke"
  #One Duke Stage from each pair since each pair have same stages
  Duke=as.character(cha[(1:94)*2-1,7])
```

```
 #Convert Duke Stages to numerical values (B=1, C=2). Only use
 # B and C here first
 NDuke=1*(Duke==" B")+2*(Duke==" C")

 #Get a sequence of positions corresponding to "B" stage, call it Blist
 Blist=seq(length(NDuke))[Duke==" B"]
 #Get a sequence of positions corresponding to "C" stage, call it Clist
 Clist=seq(length(NDuke))[Duke==" C"]

 n=8
 #Get the length of Blist (i.e. count the number of "B" stages) (=46)
 nB=length(Blist)
 #Get the length of Clist (i.e. count the number of "C" stages)(=37)
 nC=length(Clist)
 fold=10
 source("functions.s")
 source("faster.heter.gamma.r")
pv.fold=numeric()
 for (group in 1:fold){
           ngb=trunc(nB/fold);ngc=trunc(nC/fold)
       testpo=c(((1:ngb)+ngb*(group-1)),(1:ngc)+ngc*(group-1))
       testposition=c(Blist[testpo[1:ngb]],Clist[testpo[ngb+(1:ngc)]])
       trainposition=c(Blist[-testpo[1:ngb]],Clist[-testpo[ngb+(1:ngc)]])
       testclass=NDuke[testposition]
       trainclass=NDuke[trainposition]
              testX=dat[sigSNP.id,testposition]
       trainX=dat[sigSNP.id,trainposition]
       patID=c(1:(nB-ngb),1:(nC-ngc)  )
       traindata=data.frame(patID,trainclass,t(trainX))
```

```
        traindataformat1=dataformat2(traindata)
        ranks=rank(traindataformat1[,4])
        mydat=cbind(traindataformat1, ranks)
        org=Heter.gamma2(mydat, 2, length(sigSNP.id), mn
        =c((nB-ngb),(nC-ngc)), 5)

        pv.fold=c(pv.fold,org$pgamma)
}

# all of them are non-significant
# [1] 0.2394575 0.3204927 0.4178413 0.3182170 0.5518691 0.4749392
#  0.1996476 0.2739037 0.8633620 0.3097809
report.sigSNP.table=sig.SNPs[,-3]
row.names(report.sigSNP.table )= SNP.id.block[SNP.num]

write.table( SNP.id.block[SNP.num], file="sig.SNPs.within.block115",
row.names=F)

colnames(report.sigSNP.table ) =c("p.value", "99lower.bound",
"mean.ratio.across.folds")

library(xtable)
xtable(report.sigSNP.table, digits=4)
```

A.2.7 Try with Wilcoxon Test and t-test on Individual SNP

```
############################
#### In order to compare the results for feature selection using
#our proposed method with other methods (i.e.: Some score was
#computed based on t-statistic or Wilconxon test on each indificual gene
#### and the selection was based on the those genes ranked top based
#on this calcuated socre.), we also calcutate the p-values of each SNP
#### using Mann-Whitney test (Extension of Wilcoxon test for
#arbitrary sample sizes) and t test. Different multiple comparison
#criteria are considered, ie.,Family Wise Error Rate with Bonferroni correction
###  False Discovery Rate with Benjamin Hochberg FDR.
####  The results from both sets of tests found no significant SNPs
# with either FDR or FWER control.

dat=read.table("tdat.txt",header=T);
 #Name Physical characteristic data "cha"
  cha=read.csv("test.csv")
  #Extract the column of Duke stage from characteristic data, call it "Duke"
  #One Duke Stage from each pair since each pair have same stages
  Duke=as.character(cha[(1:94)*2-1,7])

  #Convert Duke Stages to numerical values (B=1, C=2). Only use B and C
  NDuke=1*(Duke==" B")+2*(Duke==" C")

  #Get a sequence of positions corresponding to "B" stage, call it Blist
  Blist=seq(length(NDuke))[Duke==" B"]
  #Get a sequence of positions corresponding to "C" stage, call it Clist
  Clist=seq(length(NDuke))[Duke==" C"]
```

```
 n=8
 #Get the length of Blist (i.e. count the number of "B" stages) (=46)
 nB=length(Blist)
 #Get the length of Clist (i.e. count the number of "C" stages)(=37)
 nC=length(Clist)
fold=10;alpha=0.05
pv.fold.FWER=pv.fold.FDR= t.pv.fold.FWER=t.pv.fold.FDR=list()
all.pv.wilcox=NULL; all.pv.t=NULL
 for (group in 1:fold){
           ngb=trunc(nB/fold);ngc=trunc(nC/fold)
       testpo=c(((1:ngb)+ngb*(group-1)),(1:ngc)+ngc*(group-1))
       testposition=c(Blist[testpo[1:ngb]],Clist[testpo[ngb+(1:ngc)]])
       trainposition=c(Blist[-testpo[1:ngb]],Clist[-testpo[ngb+(1:ngc)]])
       testclass=NDuke[testposition]
       trainclass=NDuke[trainposition]
           testX=dat[sigSNP.id,testposition]
       trainX=dat[,trainposition]

  #         myWilcox=function(z) wilcox.test(x=z[trainclass==1], y
  # =z[trainclass==2])$p.value
  #         wilcox.p=apply(trainX, 1, myWilcox)
   #        all.pv.wilcox=cbind( all.pv.wilcox,wilcox.p)
    #       wil.sig.FWER= (1:length(wilcox.p))[wilcox.p<0.05/length(wilcox.p)]
     #      wil.sig.FDR= (1:length(wilcox.p))[wilcox.p[order(wilcox.p)]
     # <(1:length(wilcox.p))/length(wilcox.p)*alpha]
   #   pv.fold.FWER[[group]]= wil.sig.FWER
   #   pv.fold.FDR[[group]]= wil.sig.FDR

   myT2=function(z)  t.test(x=z[trainclass==1], y =z[trainclass==2])$p.value
```

```
          t.p=apply(trainX,1, myT2)
           all.pv.t=cbind( all.pv.t,t.p)
             t.sig.FWER= (1:length(t.p))[t.p<0.05/length(t.p)]
             t.sig.FDR= (1:length(t.p))[t.p[order(t.p)] <(1:length(t.p))/
             length(t.p)*alpha]

         t.pv.fold.FWER[[group]]= t.sig.FWER
         t.pv.fold.FDR[[group]]= t.sig.FDR


}

ave.pv=apply(all.pv.wilcox, 1, mean)
write.table(cbind(1:length(ave.pv),ave.pv)[(order(ave.pv)),],file="wilcoxon.pv")

ave.pv.t=apply(all.pv.t, 1, mean)
write.table(cbind(1:length(ave.pv.t),ave.pv.t)[(order(ave.pv.t)),],file="ttest.pv")
```

A.2.8 Classification with SVM

```
######## Use selected 15 SNPs from bolck 115 to classify the
# Duke's stage with SVM as classifer.
######## Leave-One-Out(LOO) CV is used. The training sample
#size is 82, and the test sample size is 1. The class of test
######## instance is predicted based on the training sample
#of size 82, using SVM.
######## All test data (83 of them) are correctly classifiedm,
#i.e.: all 83 accuracy are 1.
########
```

```
setwd("C:\\Users\\plg519\\Documents\\Academic\\MS Report\\R
Files\\After 115 block\\pv results for deleting 1")

source("functions.s")
#Name the snaps data
#(i.e. a 58494(snaps)*94(paris of patiances) matrix) "dat"
dat=read.table("tdat.txt",header=T)


#Name Physical characteristic data "cha"
cha=read.csv("test.csv")


#Extract the column of Duke stage from characteristic data,
#call it "Duke"
#One Duke Stage from each pair since each pair have same stages
Duke=as.character(cha[(1:94)*2-1,7])


#NDuke=1*(Duke==" A")+2*(Duke==" B")+3*(Duke==" C")+4*(Duke==" D")
#order(NDuke)
#table(NDuke)

#Convert Duke Stages to numerical values (B=1, C=2). Only use B and C

NDuke=1*(Duke==" B")+2*(Duke==" C")

#Get a sequence of positions corresponding to "B" stage, call it Blist
Blist=seq(length(NDuke))[Duke==" B"]
#Get a sequence of positions corresponding to "C" stage, call it Clist
```

```
Clist=seq(length(NDuke))[Duke==" C"]

n=8
#Get the length of Blist (i.e. count the number of "B" stages) (=46)
nB=length(Blist)
#Get the length of Clist (i.e. count the number of "C" stages)(=37)
nC=length(Clist)
#bp=sample(1:nB,nB); cp=sample(1:nC,nC)

alpha=0.05
sigSNP.id=(read.table(  file="sig.SNPs.within.block115", header=T))[,1]

##############################################




svm.test.with.par= function(Yu, testuse, crossv=10){

 library(e1071)
       obj = tune(svm, response~., data = Yu,
              ranges = list(gamma = 2^(-3:1), cost = 2^(2:4)),
               tunecontrol = tune.control(cross=crossv )
              )
       bestGamma = obj$best.parameters[[1]]
       bestC = obj$best.parameters[[2]]

#Build model using SVM
 svm1=svm(response~., data=Yu, cost=bestC, gamma=bestGamma )
```

```
#Compute the predicted class for test data
totest=data.frame(testuse[,-ncol(testuse)])
colnames(totest)=colnames(Yu)[-ncol(testuse)]
 pred <- as.character(predict(svm1, totest) )
 pred

#Construct confusious matrix (table)
 confuTable=table(pred,testuse$response)

#Compute accuracy
 accuracy=sum(diag(confuTable))/(sum(confuTable))
#list(parameter=c(bestGamma,bestC),accuracy=accuracy)
c(bestGamma,bestC,accuracy)
}



fold=nB+nC;
 #pv.all=numeric();
accuracy=NULL
for (group in 1:fold){
    #ngb=trunc(nB/fold);ngc=trunc(nC/fold)
    #testpo=c(((1:ngb)+ngb*(group-1)),(1:ngc)+ngc*(group-1))
    #testposition=c(Blist[testpo[1:ngb]],Clist[testpo[ngb+(1:ngc)]])
    #trainposition=c(Blist[-testpo[1:ngb]],Clist[-testpo[ngb+(1:ngc)]])
      testposition=c(Blist,Clist)[group]
      trainposition=c(Blist,Clist)[-group]
    testclass=NDuke[testposition]
    trainclass=NDuke[trainposition]

    trainuse=data.frame(t(dat[sigSNP.id, trainposition]),
```

```
    response=as.factor(trainclass))
    testuse=data.frame(t(dat[sigSNP.id,testposition]),
    response=as.factor(testclass))

    #testX=data.frame(t(dat[sigSNP.id,testposition]))

                ######################
                #Use SVM
    accuracy=rbind(accuracy,svm.test.with.par(trainuse,testuse))
    }

 write.table(accuracy,file="LOOCV.accuracy.with.parameter.txt",
 row.names=F)
library(xtable)
accuracy2=cbind(accuracy[,-3], ifelse(accuracy[,3]==1, rep(
"Yes",nrow(accuracy)), rep("No",nrow(accuracy)) ) )
row.names(accuracy2)= paste("Patient", c(Blist,Clist))
colnames(accuracy2)=c( "gamma", "cost", "Correctly classified")
xtable(accuracy2)
res=cbind( paste("Patient", c(Blist,Clist)[1:42]), accuracy2[1:42, ],
rbind(cbind(paste("Patient", c(Blist,Clist)[43:83]), accuracy2[43:83,]),
 rep("", 4)))
xtable(res)
```

A.2.9 Classification with KNN

```
######## Use selected 15 SNPs from bolck 115 to classify the Duke's
```

```
#stage with SVM as classifer.
######## Leave-One-Out(LOO) CV is used. The training sample size is 82,
#and the test sample size is 1. The class of test
######## instance is predicted based on the training sample of size 82,
# using SVM.
######## All test data (83 of them) are correctly classifiedm,i.e.:
#all 83 accuracy are 1.
########
setwd("C:\\Users\\plg519\\Documents\\Academic\\MS Report\\
R Files\\After 115 block\\pv results for deleting 1")
source("functions.s")
#Name the snaps data
#(i.e. a 58494(snaps)*94(paris of patiances) matrix) "dat"
dat=read.table("tdat.txt",header=T)


#Name Physical characteristic data "cha"
cha=read.csv("test.csv")


#Extract the column of Duke stage from characteristic data, call it "Duke"
#One Duke Stage from each pair since each pair have same stages
Duke=as.character(cha[(1:94)*2-1,7])


#NDuke=1*(Duke==" A")+2*(Duke==" B")+3*(Duke==" C")+4*(Duke==" D")
#order(NDuke)
#table(NDuke)

#Convert Duke Stages to numerical values (B=1, C=2). Only use B and C
```

```
NDuke=1*(Duke==" B")+2*(Duke==" C")

#Get a sequence of positions corresponding to "B" stage, call it Blist
Blist=seq(length(NDuke))[Duke==" B"]
#Get a sequence of positions corresponding to "C" stage, call it Clist
Clist=seq(length(NDuke))[Duke==" C"]

n=8
#Get the length of Blist (i.e. count the number of "B" stages) (=46)
nB=length(Blist)
#Get the length of Clist (i.e. count the number of "C" stages)(=37)
nC=length(Clist)
#bp=sample(1:nB,nB); cp=sample(1:nC,nC)

alpha=0.05
sigSNP.id=(read.table(  file="sig.SNPs.within.block115", header=T))[,1]

##############################################




svm.test.with.par= function(Yu, testuse, crossv=10){

 library(e1071)
       obj = tune(svm, response~., data = Yu,
              ranges = list(gamma = 2^(-3:1), cost = 2^(2:4)),
               tunecontrol = tune.control(cross=crossv )
              )
```

```
        bestGamma = obj$best.parameters[[1]]
        bestC = obj$best.parameters[[2]]

#Build model using SVM
 svm1=svm(response~., data=Yu, cost=bestC, gamma=bestGamma )

#Compute the predicted class for test data
totest=data.frame(testuse[,-ncol(testuse)])
colnames(totest)=colnames(Yu)[-ncol(testuse)]
 pred <- as.character(predict(svm1, totest) )
 pred

#Construct confusious matrix (table)
 confuTable=table(pred,testuse$response)

#Compute accuracy
 accuracy=sum(diag(confuTable))/(sum(confuTable))
#list(parameter=c(bestGamma,bestC),accuracy=accuracy)
c(bestGamma,bestC,accuracy)
}



fold=nB+nC;
 #pv.all=numeric();
accuracy=NULL
for (group in 1:fold){
    #ngb=trunc(nB/fold);ngc=trunc(nC/fold)
    #testpo=c(((1:ngb)+ngb*(group-1)),(1:ngc)+ngc*(group-1))
    #testposition=c(Blist[testpo[1:ngb]],Clist[testpo[ngb+(1:ngc)]])
    #trainposition=c(Blist[-testpo[1:ngb]],Clist[-testpo[ngb+(1:ngc)]])
```

```
      testposition=c(Blist,Clist)[group]
      trainposition=c(Blist,Clist)[-group]
    testclass=NDuke[testposition]
    trainclass=NDuke[trainposition]

    trainuse=data.frame(t(dat[sigSNP.id, trainposition]),
    response=as.factor(trainclass))
    testuse=data.frame(t(dat[sigSNP.id,testposition]),
    response=as.factor(testclass))

    #testX=data.frame(t(dat[sigSNP.id,testposition]))

                ######################
                #Use SVM
    #accuracy=rbind(accuracy,svm.test.with.par(trainuse,testuse))
       accuracy=rbind(accuracy, myknn(trainuse,testuse))
    }

write.table(accuracy,file="dat1.LOOCV.accuracy.knn.with.parameter.txt",
 row.names=F)

library(xtable)
accuracy2=cbind(accuracy[,-2], ifelse(accuracy[,2]==1, rep("Yes"
,nrow(accuracy)), rep("No",nrow(accuracy)) ) )
row.names(accuracy2)= paste("Patient", c(Blist,Clist))
colnames(accuracy2)=c( "k", "Correctly classified")
xtable(accuracy2)
res=cbind( paste("Patient", c(Blist,Clist)[1:42]), accuracy2[1:42, ],
 rbind(cbind(paste("Patient", c(Blist,Clist)[43:83]), accuracy2[43:83,]
 ), rep("", 3)))
```

```
xtable(res)



myknn=function(Yu, testuse, crossv=10){

 library(e1071)
       obj = tune.knn(Yu[,-ncol(Yu)], Yu[,ncol(Yu)],
            k=1:25,
               tunecontrol = tune.control(cross=crossv )
              )
       bestk = obj$best.parameters



#Compute the predicted class for test data
totest=data.frame(testuse[,-ncol(testuse)])
colnames(totest)=colnames(Yu)[-ncol(testuse)]
#Build model using knn
 knn1=knn(train=Yu[,-ncol(Yu)], test= totest, cl=Yu[,ncol(Yu)], k=bestk )
 pred <- as.character(knn1 )
 pred

#Construct confusious matrix (table)
 confuTable=table(pred,testuse$response)

#Compute accuracy
 accuracy=sum(diag(confuTable))/(sum(confuTable))
#list(parameter=c(bestGamma,bestC),accuracy=accuracy)
unlist(c(bestk,accuracy))
}
```

Appendix B

Java and R code for mRNA expression dataset

B.1 Java Code to Retrieve Characteristics Data for mRNA Dataset

```
import java.io.*; import java.net.*; import java.util.Arrays; public
class JavaGetUrl { public static void main(String[] args) throws
IOException {

   String record = new String();
      String[] URL = new String[290]; //290 is the number of URLs

  URL[0] = "http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM358341";
    int i = 0;
```

```
  //BufferedWriter out = new BufferedWriter(new FileWriter("outfile.csv"));

  //String str = "";

FileWriter writer = new FileWriter("test.csv");
     writer.append("Location, DukesStage, Age_Diag,
                   Gender, DFS_Time, DFS_Cens, AdjXRT, AdjCTX");
     writer.append('\n');

    while (i < 290) {
        String digit = URL[i].substring(56,59);
        //the indices of the last digits of the URL that vary

        int idigit = Integer.parseInt(digit);

        int idigit2 = idigit + 1;

        String sdigit2 = Integer.toString(idigit2);

        URL[i+1] = URL[i].substring(0,56) + sdigit2;
        //combine the fixed part of URL and the last several digits that vary,
        //(0,56) is the indices of the fixed digits


        record = Arrays.toString(ret(URL[i])).replace("[", "").replace("]", "");
        //record is the retrieved features for

        //System.out.println(record);
```

```
        writer.append(record);
        writer.append('\n');

        i++;
        }
         writer.flush();
         writer.close();
}

  @SuppressWarnings("deprecation")
  public static String[] ret(String url) {

        URL u;
        InputStream is = null;
        DataInputStream dis;
        String s;

        int[] idx= new int[16];
        String[] feature = new String[8];

        try{
            u = new URL(url);
            is = u.openStream();
            dis = new DataInputStream(new BufferedInputStream(is));


            while ((s = dis.readLine()) != null){
                if (s.contains("Location:")){
                   //System.out.println(s);
```

```
idx[0] = s.indexOf("Location:") + 10;
idx[1] = s.indexOf("; DukesStage:");

idx[2] = s.indexOf("DukesStage:") + 12;
idx[3] = s.indexOf("; Age_Diag:");

idx[4] = s.indexOf("Age_Diag:") + 10;
idx[5] = s.indexOf("; Gender:");

idx[6] = s.indexOf("Gender: ") + 8;
idx[7] = s.indexOf("; DFS_Time:");

idx[8] = s.indexOf("; DFS_Time:") + 12;
idx[9] = s.indexOf("; DFS_Cens:");

idx[10] = s.indexOf("; DFS_Cens:") + 12;
idx[11] = s.indexOf("; AdjXRT:");

idx[12] = s.indexOf("; AdjXRT:") + 10;
idx[13] = s.indexOf("; AdjCTX:");

idx[14] = s.indexOf("; AdjCTX:") + 10;
idx[15] = s.indexOf("<br></td>");

feature[0] = s.substring(idx[0],idx[1]);
feature[1] = s.substring(idx[2],idx[3]);
feature[2] = s.substring(idx[4],idx[5]);
feature[3] = s.substring(idx[6],idx[7]);
feature[4] = s.substring(idx[8],idx[9]);
feature[5] = s.substring(idx[10],idx[11]);
```

```
            feature[6] = s.substring(idx[12],idx[13]);
            feature[7] = s.substring(idx[14],idx[15]);


          }
        }

    }
  catch (MalformedURLException mue){

    System.out.println("Ouch -a MalformedURLExeption happened.");
    mue.printStackTrace();
    System.exit(1);
  }
  catch (IOException ioe) {

      System.out.println("Oops- an IOException happend");
      ioe.printStackTrace();
      System.exit(1);

   } finally{

       try {
          is.close();
       } catch (IOException ioe) {

          }
   }
return feature;
}
```

```
}
```

B.2 R Code for mRNA expression Dataset

B.2.1 Data Preprocessing

```
#################################
## Data preprocessing: Downloaded CEL data from the nih website, and then used
# R package gcrma to summarize probe sets expression, and converted to csv file.
# Downloaded annotation of human U 133 plus 2.0 array from affymatrix website.
## Based on the "Chromosomal location" column from annotatiton csv file, the sex
# chromosomes were idenfied and excluded. In addition, the remianing probe sets
# were sorted based on the locaion info. obtained from "Alligenment" column from
# annotation file.
```

```
setwd("C:\\Users\\plg519\\Documents\\Academic\\MS Report 2nd Dataset")
# the cel. files are in here.
library(affy)
 library(gcrma)
#Load the .CEL files from my directary.
filenames=paste("GSM358",341:630,".CEL",sep="")
```

```
for(i in 1:29){
try = just.gcrma(filenames=filenames[(i-1)*10+(1:10)])
dati=data.frame( featureNames(try), exprs(try))
write.csv(dati,file=paste("GCRMA", i,".expression.probeset.csv",sep=""),
 row.names=F)
}

dat=NULL
for(i in 1:29){
dat=cbind(dat,as.matrix(read.csv(file=paste("GCRMA", i,
".expression.probeset.csv",sep=""),row.names=1)))
}
write.csv(dat,file="GSE14333.GCRMA.csv")

anno=read.csv(file="HG-U133_Plus_2.na31.annot.csv",skip=24)



location=as.character(anno[,16])
YLOC=grep("chrY",location)
XLOC=grep("chrX",location)

#matchRowID=(match(as.character(dat2[,1]),as.character(anno[,1])))
#(1:54675)[abs((matchRowID-(1:54675)))>0]

SEXLOC=c(XLOC,YLOC)
sexProb=anno[-SEXLOC,c(1,13,16)]
dim(sexProb)
head(sexProb)

ordSexProb=sexProb[order(sexProb[,2]),]
```

```
#Check the number of probe sets with neither Alliganment
#locatoion or Chromoseomal Location, i.e., 831
#temp=ordSexProb[(ordSexProb[,2]=="---")&(ordSexProb[,3]=="---"),]

#Check the number of probe sets with no Alliganment locatoion or
#Chromoseomal Location, i.e., 13414
#temp1=ordSexProb[(ordSexProb[,2]=="---")|(ordSexProb[,3]=="---"),]

##Check the number of probe sets with no info. of
#Alliganment locatoion, i.e., 1075
#temp11=ordSexProb[(ordSexProb[,2]=="---"),]
#dim(temp11)

matchID=match(as.character(ordSexProb[,1]),as.character(dat2[,1]))
head(matchID)

#Extract the probe sets excluding sex-related chomosomes
#from dat2 dataset based on Probe
#set ID from annotation file. These probe sets were ordered
#according to their location on
#chromosomes (info. contained in Alliganment in annotation file)
write.csv(dat2[matchID,],file="orderedDat.csv",row.names=F)
```

B.2.2 Test for Interaction Effect for Each Block

```
#setwd("C:\\Users\\plg519\\Documents\\Academic\\MS Report 2nd
#Dataset\\Results\\Stage0p-valuesForFeatureSelection\\WithOrder")
############################
```

```
## Combine the p-values results for fold 1 from 30 separated
#running reuslts into one file


splitMat=matrix(c(kronecker(1770*(0:9), rep(1,2)), 17700+
kronecker(1770*(0:9), rep(1,2)), kronecker(1770*(0:9), rep(1,2))+
35400,  53158)[-1], nrow=30, byrow=T)

fold1=numeric()

 for(j in 1:nrow(splitMat)){
totalLeft=splitMat[j,1]
file=paste("dat2.pv.for", group,".", totalLeft, ".txt", sep="")
fold1=rbind(fold1,read.table(file))

}

#write.table(fold1, file="dat2.pv.for1WithDecimal.txt",row.name=F)
fold1[,2]=trunc(fold1[,2])

#write.table(fold1, file="dat2.pv.for1.txt",row.name=F,col.name=F)
##############################################

step0=100;alpha=0.05
##############################################
fold=10; pv.all=NULL;
sigblocks.all=vector("list",fold)
BonSigBlocks.all=vector("list",fold)
FDRSigBlocks.all=vector("list",fold)
```

```
for(group in 1:fold){
    pvUnord=read.table(paste("dat2.pv.for", group, ".txt", sep="") )[,2:3]

      uni=pvUnord[match(sort(unique(pvUnord[,1])),pvUnord[,1]),]
      pv=uni[,2]
    #pvOrd=pvUnord[order(pvUnord[,1]),]
    #temp=c(0,177,177*2)+pvOrd[,1]
    #pv=pvOrd[order(temp),2]

    pv.all=cbind(pv.all,pv)
    lengthpv=length(pv)

      #print(c(group,lengthpv))

    FDRSigBlocks=((1:lengthpv)[order(pv)])[(pv[order(pv)] <(1:lengthpv)/
    lengthpv*alpha)  ]       #FDR control
    BonSigBlocks=(1:lengthpv)[pv<alpha/lengthpv] # Bonferroni correction

    sigblocks=(1:lengthpv)[pv<alpha]   # no multiple-comparison adjustment
     # Then use common significant blocks from all 'group's.
            ###################################################
    sigblocks.all[[group]]=sigblocks
    BonSigBlocks.all[[group]]=BonSigBlocks
    FDRSigBlocks.all[[group]]=FDRSigBlocks


}

commonSigBlocks=function(sigblocks.all) {
```

```
    T.orF=logical()
    uniquesigs=unique(unlist(sigblocks.all))
    for (h in 1:length(uniquesigs)){
        T.orF=c(T.orF, all(unlist(lapply(sigblocks.all, function(x)
        uniquesigs[h] %in% x))) )
    }
    sig.in.all=uniquesigs[T.orF]
      sig.in.all
}

commonSigBlocks(sigblocks.all)
commonSigBlocks(BonSigBlocks.all)
commonSigBlocks(FDRSigBlocks.all)









nblock=max(pvUnord[,1])

sig.in.all=commonSigBlocks(BonSigBlocks.all)
########################


source("functions.s")
source("faster.heter.gamma.r")

dat=read.csv("orderedDat.csv",row.names=1)
```

```
cha=read.csv("testCompleteFinal.csv")

#dat=read.csv(file="C:\\Users\\plg519\\Documents\\Academic\\MS Report
2nd Dataset\\orderedDat.csv",row.names=1)
#cha=read.csv("C:\\Users\\plg519\\Documents\\Academic\\MS Report 2nd
Dataset\\testCompleteFinal.csv")

set.seed(10)
randOrd=sample(nrow(cha))
#print(dim(dat))
#print(randOrd)

dat=data.frame(dat[,randOrd])
cha=cha[randOrd,]

Duke=as.character(cha[,3])

NDuke=1*(Duke==" B")+2*(Duke==" C")

Blist=seq(length(NDuke))[Duke==" B"]
Clist=seq(length(NDuke))[Duke==" C"]

nB=length(Blist)
nC=length(Clist)



########################3 do delete one SNP at a time and
#recalculate p-values
group=group0
```

```
#for (group in 1:fold){
#pv.delete1=numeric()
for ( hsig in 1:length(sig.in.all) ){
    step=ifelse(sig.in.all[hsig]<nblock, step0, nrow(dat)-nblock*step0)
    sigSNP.id.block=c( t(kronecker(matrix(1:step,nrow=1),
    matrix(1, nrow=length(sig.in.all[hsig]) ))+(sig.in.all[hsig]-1)*step))

       pv.100=numeric()
       for ( jjj in 1:step){
          sigSNP.id= sigSNP.id.block[-jjj]

        ngb=trunc(nB/fold);ngc=trunc(nC/fold)
        testpo=c(((1:ngb)+ngb*(group-1)),(1:ngc)+ngc*(group-1))
        testposition=c(Blist[testpo[1:ngb]],Clist[testpo[ngb+(1:ngc)]])
        trainposition=c(Blist[-testpo[1:ngb]],Clist[-testpo[ngb+(1:ngc)]])
        testclass=NDuke[testposition]
        trainclass=NDuke[trainposition]
                testX=dat[sigSNP.id,testposition]
        trainX=dat[sigSNP.id,trainposition]
        patID=c(1:(nB-ngb),1:(nC-ngc)  )
        traindata=data.frame(patID,trainclass,t(trainX))

        traindataformat1=dataformat2(traindata)
        ranks=rank(traindataformat1[,4])
        mydat=cbind(traindataformat1, ranks)
        org=Heter.gamma2(mydat, 2, step-1, mn=c((nB-ngb),(nC-ngc)), 5)
        pv.100=c(pv.100,org$pgamma)
         cat(c(group,sig.in.all[hsig], jjj,pv.all[sig.in.all[hsig],group],
                org$pgamma),"\n", file=paste("dat2.delete1.pv.for.group",
                group, ".txt", sep=""),append=T)
```

```
      }
    cat(c(group,sig.in.all[hsig], jjj,pv.all[sig.in.all[hsig],
    group], pv.100),"\n",file="dat2.delete1.pv.txt",append=T)
  #  pv.delete1=rbind(pv.delete1, pv.100)

 }



#}









                ######################
                #Use SVM
                #accuracy=c(accuracy,svm.test(trainuse,testuse))

#   }
  # write.table(pv,file=paste("pv.for.fold", group, ".txt", sep=""))

#} #end of group loop

#write.table(pv.all,file="pv.txt",row.names=F)
```

```
#write.table(accuracy,file="accuracy.txt")
#cat("*significant blocks\n",file="blocks.txt", append=F)
#for(k in 1:fold) {
# cat(paste("fold ", k,sep=""), unlist(sigblocks.all[[k]]), "\n",
#file="blocks.txt", append=T)
#}
```

B.2.3 Calculation of Delete-One p-values and Identification of Significant Probe Sets

```
############################
####  After observing the some of differences of p-values between
#using 100 SNPs and 99 SNPs seem small.  To find a criterion for
#selecting the significant SNPs within block 115.
####  The ratio of p-values between using 100 SNPs and 99 SNPs
#were calcuated. Then the ratios across 10 folds for each SNP
#serves as a random sample, and the t-test was conducted
####  to select those SNPs with the mean ratio significant
#greater than 1. 15 SNPs were selected.

#### After finding the 15 significant SNPs within block 115
#### Excluding these 15 SNPs, the remaining 85 SNPs were
#tested for each of the 10 folds.
#### The resutls show that non of them are significant
####

setwd("C:\\Users\\plg519\\Documents\\Academic\\MS Report
```

```
2nd Dataset\\Results\\State1_Within23_selection")


#nblock
all.g=NULL
reverse.rank=NULL


for (i in 1:10) {
#get all 100 p-values from 10 fold
pi=read.table(paste("dat2.delete1.pv.for.group",i,".txt", sep=""))
#construct the matrix including p-values and corresponding
#ratios with/without one SNP
gi=as.matrix(cbind(pi[,c(1,3:5)], pi[,5]/pi[,4]))
all.g=cbind(all.g,gi)
reverse.rank=cbind(reverse.rank, 101-rank(pi[,5]/pi[,4]) )
}

idx=order(apply(reverse.rank,1,mean))
SNPid=(1:100)[idx]
ratio=round(all.g[idx,5*(1:10)],2)
colnames(ratio)=paste("G", 1:10,"AovB", sep="")
row.names(ratio)=SNPid


###  For each SNP in this block, calculate a 99% confidence
#lower bound to see if the mean is > 1.
###  t-test was used for dataset1, but t-test was not used
# for dataset2 due to the extreme values in fold1
#myT=function(x) unlist( t.test(x, alternative = "greater",
```

```
# mu = 1, conf.level = 0.99)[3:5])
#t.test.result=t(apply(ratio,1, myT))
#myT(ratio[1,])
#t.test.result[,1]<0.01
#sig.SNPs=subset(t.test.result, t.test.result[,1]<0.01)

### Since the ratios from fold 1 seems extremme,
#the sign test is conducted instead of t-test
library(BSDA)
mySign=function(x) c(SIGN.test(x, md = 1, alternative =
 "greater", conf.level = 0.99)[2,2], median(x))
#temp=SIGN.test(ratio[1,], md = 1, alternative = "greater",
#conf.level = 0.99)
#temp[2, 2]
Sign.test.result=t(apply(ratio,1, mySign))
sig.SNPs=Sign.test.result[Sign.test.result[,1]>1,]

boxplot(ratio)
postscript("boxplot_log_ratio_dat2.eps")
boxplot(log(ratio), main="Boxplot of the log(ratios of p-values)" )
dev.off()

SNP.num=as.numeric(row.names(sig.SNPs))

### Exclude these 36 significant SNPs from 115th blcok(has 100
#SNPs) and test to see whether
### the remaining SNPs are still significant


dat=read.csv(file="C:\\Users\\plg519\\Documents\\Academic\\MS
```

```
 Report 2nd Dataset\\orderedDat.csv",row.names=1)
cha=read.csv("C:\\Users\\plg519\\Documents\\Academic\\MS
 Report 2nd Dataset\\testCompleteFinal.csv")

set.seed(10)
randOrd=sample(nrow(cha))
dat=data.frame(dat[,randOrd])
cha=cha[randOrd,]

step0=100
nblock=trunc(nrow(dat)/step0)
#nblock =584
 step=ifelse(pi[1,2]<nblock, step0, nrow(dat)-nblock*step0)
 SNP.id.block=c( t(kronecker(matrix(1:step,nrow=1), matrix(1,
 nrow=length(pi[1,2]) ))+(pi[1,2]-1)*step))
  sigSNP.id=SNP.id.block[-SNP.num]

####### get the training data for above SNP ids and do one test
  #Name Physical characteristic data "cha"
  #Extract the column of Duke stage from characteristic data,
  #call it "Duke"
  #One Duke Stage from each pair since each pair have same stages

Duke=as.character(cha[,3])

NDuke=1*(Duke==" B")+2*(Duke==" C")

Blist=seq(length(NDuke))[Duke==" B"]
Clist=seq(length(NDuke))[Duke==" C"]
```

```
nB=length(Blist)
nC=length(Clist)

   fold=10
  source("C:\\Users\\plg519\\Documents\\Academic\\MS Report 2nd
   Dataset\\functions.s")
  source("C:\\Users\\plg519\\Documents\\Academic\\MS Report 2nd
  Dataset\\faster.heter.gamma.r")
  pv.fold=numeric()
  for (group in 1:fold){
            ngb=trunc(nB/fold);ngc=trunc(nC/fold)
        testpo=c(((1:ngb)+ngb*(group-1)),(1:ngc)+ngc*(group-1))
        testposition=c(Blist[testpo[1:ngb]],Clist[testpo[ngb+(1:ngc)]])
        trainposition=c(Blist[-testpo[1:ngb]],Clist[-testpo[ngb+(1:ngc)]])
        testclass=NDuke[testposition]
        trainclass=NDuke[trainposition]
                testX=dat[sigSNP.id,testposition]
        trainX=dat[sigSNP.id,trainposition]
        patID=c(1:(nB-ngb),1:(nC-ngc)  )
        traindata=data.frame(patID,trainclass,t(trainX))

        traindataformat1=dataformat2(traindata)
        ranks=rank(traindataformat1[,4])
        mydat=cbind(traindataformat1, ranks)
        org=Heter.gamma2(mydat, 2, length(sigSNP.id), mn=c((nB-ngb),
        (nC-ngc)), 5)
        pv.fold=c(pv.fold,org$pgamma)
}

# all of them are non-significant
```

```
# [1] 0.8450296 0.7899294 0.8728943 0.8625813 0.9573931
# 0.7737742 0.8082923 0.7296441 0.7720006 0.8835492
report.sigSNP.table=as.matrix(sig.SNPs)
row.names(report.sigSNP.table )= row.names(dat)[SNP.id.block[SNP.num]]
colnames(report.sigSNP.table )=c("99lower.bound",
"median.ratio.across.folds")

SNP.id.block[SNP.num]
#### The indeces of the final selected probe sets
#[1] 2234 2253 2236 2300 2257 2273 2299 2295 2227
#[10] 2298 2251 2249 2291 2296 2206 2254 2279 2282
#[19] 2286 2260 2278 2268 2213 2290 2215 2218 2276
#[28] 2235 2233 2229 2202 2226 2214 2237 2225 2207


write.csv( cbind(report.sigSNP.table, SNP.id.block[SNP.num]) ,
 file="dat2.sig.SNPs.within.block23.csv")

library(xtable)
xtable(report.sigSNP.table, digits=4)
```

B.2.4 Classification Using SVM

```
######## Use selected 36 SNPs from bolck 23 to classify the
#Duke's stage with SVM as classifer.
######## Leave-One-Out(LOO) CV is used. The training sample
# size is 184, and the test sample size is 1. The class of test
######## instance is predicted based on the training sample
```

```
#of size 184, using SVM.
######## All test data (185 of them) are correctly classified,
#i.e.: all 185 accuracy are 1.
########
setwd("C:\\Users\\plg519\\Documents\\Academic\\MS Report 2nd
 Dataset\\Results\\State1_Within23_selection")


source("C:\\Users\\plg519\\Documents\\Academic\\MS Report 2nd
Dataset\\functions.s")
source("C:\\Users\\plg519\\Documents\\Academic\\MS Report 2nd
Dataset\\faster.heter.gamma.r")

#Name the snaps data (i.e. a 58494(snaps)*94(paris of patiances)
# matrix) "dat"
dat=read.csv(file="C:\\Users\\plg519\\Documents\\Academic\\MS
Report 2nd Dataset\\orderedDat.csv",row.names=1)


#Name Physical characteristic data "cha"
cha=read.csv("C:\\Users\\plg519\\Documents\\Academic\\MS Report
 2nd Dataset\\testCompleteFinal.csv")


set.seed(10)
randOrd=sample(nrow(cha))
dat=data.frame(dat[,randOrd])
cha=cha[randOrd,]
```

```
#Extract the column of Duke stage from characteristic data, call it "Duke"
#One Duke Stage from each pair since each pair have same stages
Duke=as.character(cha[,3])

NDuke=1*(Duke==" B")+2*(Duke==" C")

Blist=seq(length(NDuke))[Duke==" B"]
Clist=seq(length(NDuke))[Duke==" C"]

nB=length(Blist)
nC=length(Clist)



alpha=0.05
sigSNP.id=(read.csv("C:\\Users\\plg519\\Documents\\Academic\\MS
Report 2nd Dataset\\Results\\State1_Within23_selection\\
dat2.sig.SNPs.within.block23.csv"))[,4]

svm.test.with.par= function(Yu, testuse, crossv=10){

 library(e1071)
       obj = tune(svm, response~., data = Yu,
              ranges = list(gamma = 2^(-3:1), cost = 2^(2:4)),
               tunecontrol = tune.control(cross=crossv )
              )
       bestGamma = obj$best.parameters[[1]]
       bestC = obj$best.parameters[[2]]

#Build model using SVM
 svm1=svm(response~., data=Yu, cost=bestC, gamma=bestGamma )
```

```
#Compute the predicted class for test data
totest=data.frame(testuse[,-ncol(testuse)])
colnames(totest)=colnames(Yu)[-ncol(testuse)]
 pred <- as.character(predict(svm1, totest) )
 pred

#Construct confusious matrix (table)
 confuTable=table(pred,testuse$response)

#Compute accuracy
 accuracy=sum(diag(confuTable))/(sum(confuTable))
#list(parameter=c(bestGamma,bestC),accuracy=accuracy)
c(bestGamma,bestC,accuracy)
}



##############################################
fold=nB+nC;
 #pv.all=numeric();
accuracy=NULL
for (group in 1:fold){



    #ngb=trunc(nB/fold);ngc=trunc(nC/fold)
    #testpo=c(((1:ngb)+ngb*(group-1)),(1:ngc)+ngc*(group-1))
    #testposition=c(Blist[testpo[1:ngb]],Clist[testpo[ngb+(1:ngc)]])
    #trainposition=c(Blist[-testpo[1:ngb]],Clist[-testpo[ngb+(1:ngc)]])
      testposition=c(Blist,Clist)[group]
      trainposition=c(Blist,Clist)[-group]
```

```
    testclass=NDuke[testposition]
    trainclass=NDuke[trainposition]

    trainuse=data.frame(t(dat[sigSNP.id, trainposition]),
    response=as.factor(trainclass))
    testuse=data.frame(t(dat[sigSNP.id,testposition]),
    response=as.factor(testclass))

    #testX=data.frame(t(dat[sigSNP.id,testposition]))

                ######################
                #Use SVM
    accuracy=rbind(accuracy,svm.test.with.par(trainuse,testuse))

    }
 write.table(accuracy,file="dat2.LOOCV.accuracy.with.para.estimate.txt",
 row.names=F)

library(xtable)
accuracy2=cbind(accuracy[,-3], ifelse(accuracy[,3]==1, rep("Yes",
nrow(accuracy)), rep("No",nrow(accuracy)) ) )
row.names(accuracy2)= paste("Patient", c(Blist,Clist))
colnames(accuracy2)=c( "gamma", "cost", "Correctly classified")
xtable(accuracy2)
res=cbind( paste("Patient", c(Blist,Clist)[1:42]), accuracy2[1:42, ],
rbind(cbind(paste("Patient", c(Blist,Clist)[43:83]), accuracy2[43:83,]),
rep("", 4)))
xtable(res)

mat.accuracy=matrix(accuracy, ncol=3, byrow=T)
```

B.2.5 Classification with KNN

```
######## Use selected 36 SNPs from bolck 23 to classify the
#Duke's stage with KNN as classifer.
######## Leave-One-Out(LOO) CV is used. The training sample size
#is 184, and the test sample size is 1. The class of test
######## instance is predicted based on the training sample of
#size 184, using KNN.
######## All test data (185 of them) are correctly classified,
#i.e.: all 185 accuracy are 1.
########
setwd("C:\\Users\\plg519\\Documents\\Academic\\MS Report 2nd
 Dataset\\Results\\State1_Within23_selection")


source("C:\\Users\\plg519\\Documents\\Academic\\MS Report
 2nd Dataset\\functions.s")
source("C:\\Users\\plg519\\Documents\\Academic\\MS Report
2nd Dataset\\faster.heter.gamma.r")

#Name the snaps data (i.e.
#a 58494(snaps)*94(paris of patiances) matrix) "dat"
dat=read.csv(file="C:\\Users\\plg519\\Documents\\Academic\\
MS Report 2nd Dataset\\orderedDat.csv",row.names=1)


#Name Physical characteristic data "cha"
```

```
cha=read.csv("C:\\Users\\plg519\\Documents\\Academic\\
MS Report 2nd Dataset\\testCompleteFinal.csv")


set.seed(10)
randOrd=sample(nrow(cha))
dat=data.frame(dat[,randOrd])
cha=cha[randOrd,]


#Extract the column of Duke stage from characteristic data, call it "Duke"
#One Duke Stage from each pair since each pair have same stages
Duke=as.character(cha[,3])

NDuke=1*(Duke==" B")+2*(Duke==" C")

Blist=seq(length(NDuke))[Duke==" B"]
Clist=seq(length(NDuke))[Duke==" C"]

nB=length(Blist)
nC=length(Clist)


alpha=0.05
sigSNP.id=(read.csv("C:\\Users\\plg519\\Documents\\Academic\\MS
Report 2nd Dataset\\Results\\State1_Within23_selection\\
dat2.sig.SNPs.within.block23.csv"))[,4]

##############################################
fold=nB+nC;
```

```
 #pv.all=numeric();
accuracy=NULL
for (group in 1:fold){



    #ngb=trunc(nB/fold);ngc=trunc(nC/fold)
    #testpo=c(((1:ngb)+ngb*(group-1)),(1:ngc)+ngc*(group-1))
    #testposition=c(Blist[testpo[1:ngb]],Clist[testpo[ngb+(1:ngc)]])
    #trainposition=c(Blist[-testpo[1:ngb]],Clist[-testpo[ngb+(1:ngc)]])
      testposition=c(Blist,Clist)[group]
      trainposition=c(Blist,Clist)[-group]
    testclass=NDuke[testposition]
    trainclass=NDuke[trainposition]

    trainuse=data.frame(t(dat[sigSNP.id, trainposition]),
    response=as.factor(trainclass))
    testuse=data.frame(t(dat[sigSNP.id,testposition]),
    response=as.factor(testclass))

    #testX=data.frame(t(dat[sigSNP.id,testposition]))

                ######################
                #Use SVM
    #accuracy=c(accuracy,svm.test(trainuse,testuse))
      accuracy=rbind(accuracy, myknn(trainuse,testuse))


    }
 write.table(accuracy,file="dat2.knn.LOOCV.accuracy.txt", row.names=F)

library(xtable)
```

```
accuracy2=cbind(accuracy[,-2], ifelse(accuracy[,2]==1,
rep("Yes",nrow(accuracy)), rep("No",nrow(accuracy)) ) )
row.names(accuracy2)= paste("Patient", c(Blist,Clist))
colnames(accuracy2)=c( "k", "Correctly classified")
xtable(accuracy2)
res=cbind( paste("Patient", c(Blist,Clist)[1:92]),
accuracy2[1:92, ], rbind(cbind(paste("Patient", c(Blist,Clist)[93:185]),
 accuracy2[93:185,]), rep("", 2)))
xtable(res)



myknn=function(Yu, testuse, crossv=10){

 library(e1071)
       obj = tune.knn(Yu[,-ncol(Yu)], Yu[,ncol(Yu)],
            k=1:25,
               tunecontrol = tune.control(cross=crossv )
              )
       bestk = obj$best.parameters


#Compute the predicted class for test data
totest=data.frame(testuse[,-ncol(testuse)])
colnames(totest)=colnames(Yu)[-ncol(testuse)]
#Build model using knn
 knn1=knn(train=Yu[,-ncol(Yu)], test= totest, cl=Yu[,ncol(Yu)], k=bestk )


 pred <- as.character(knn1 )
```

```
 pred

#Construct confusious matrix (table)
 confuTable=table(pred,testuse$response)

#Compute accuracy
 accuracy=sum(diag(confuTable))/(sum(confuTable))
#list(parameter=c(bestGamma,bestC),accuracy=accuracy)
unlist(c(bestk,accuracy))
}
```