

Statistical Methods for Analysis of Genomic Data with Applications in Oncology

by

Michelle Saul

A Dissertation Presented in Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy

Approved July 2021 by the  
Graduate Supervisory Committee:

Valentin Dinu, Chair  
Li Liu  
Junwen Wang

ARIZONA STATE UNIVERSITY

August 2021

## ABSTRACT

This dissertation presents three novel algorithms with real-world applications to genomic oncology. While the methodologies presented here were all developed to overcome various challenges associated with the adoption of high throughput genomic data in clinical oncology, they can be used in other domains as well.

First, a network informed feature ranking algorithm is presented, which shows a significant increase in ability to select true predictive features from simulated data sets when compared to other state of the art graphical feature ranking methods. The methodology also shows an increased ability to predict pathological complete response to preoperative chemotherapy from genomic sequencing data of breast cancer patients utilizing domain knowledge from protein-protein interaction networks.

Second, an algorithm that overcomes population biases inherent in the use of a human reference genome developed primarily from European populations is presented to classify microsatellite instability (MSI) status from next-generation-sequencing (NGS) data. The methodology significantly increases the accuracy of MSI status prediction in African and African American ancestries.

Finally, a single variable model is presented to capture the bimodality inherent in genomic data stemming from heterogeneous diseases. This model shows improvements over other parametric models in the measurements of receiver-operator characteristic (ROC) curves for bimodal data. The model is used to estimate ROC curves for heterogeneous biomarkers in a dataset containing breast cancer and cancer-free specimen.

## ACKNOWLEDGEMENTS

I would like to acknowledge my coworkers at Caris Life Sciences without whom I would not have had the opportunity or knowledge to complete this dissertation. I would also like to acknowledge my good friend Mari Firago, who believed in me even when I did not, and my husband Justin Saul for the moral support and sustenance.

# TABLE OF CONTENTS

	Page
LIST OF TABLES .....	v
LIST OF FIGURES .....	vi
CHAPTER	
1 INTRODUCTION.....	1
1.1 Goal .....	1
1.2 Family Rank .....	1
1.3 Population Bias in MSI.....	2
1.4 Bimixt .....	3
2 FAMILY RANK: A GRAPHICAL KNOWLEDGE INFORMED FEATURE RANKING ALGORITHM.....	6
2.1 Introduction.....	6
2.2 Background .....	7
2.3 Methods .....	10
2.4 Results .....	17
2.5 Discussion .....	32
3 POPULATION BIAS IN SOMATIC MEASUREMENTS OF MICROSATELLITE INSTABILITY.....	35
3.1 Introduction.....	35
3.2 Background .....	35
3.3 Methods .....	42
3.4 Results .....	49
3.5 Discussion .....	58

CHAPTER	Page
4	BIMIXT: MAXIMUM LIKELIHOOD ESTIMATION OF RECEIVER OPERATING CHARACTERISTIC (ROC) CURVES FROM MULTIMODAL, NON-GAUSSIAN DATA.....60
4.1	Introduction.....60
4.2	Background .....61
4.3	Methods .....64
4.4	Results .....70
4.5	Discussion.....73
5	CONCLUSION AND FUTURE WORK .....75
5.1	Family Rank .....75
5.2	Population Bias in MSI.....76
5.3	Bimixt .....77
	REFERENCES.....79
APPENDIX	
A	FAMILY RANK WORKED EXAMPLE .....84
B	FAMILY RANK R PACKAGE DOCUMENTATION.....91
C	BIMIXT R PACKAGE DOCUMENTATION .....106

## LIST OF TABLES

Table	Page
Table 2.1 Parameters Assessed by Cross Validation .....	16
Table 2.2 Ranks of True Predictors by Ranking Method.....	20
Table 2.3 Cross Validation Classification Performance on Simulated Data .....	21
Table 2.4 Independent Validation Performance on Simulated Data .....	23
Table 2.5 Cross Validation Classification Performance on Breast Cancer Data .....	24
Table 2.6 Independent Validation Performance on Breast Cancer Data .....	26
Table 2.7 Selected Features by Ranking Method .....	27
Table 2.8 STRINGdb Network Statistics of Selected Genes on Breast Cancer Data.....	30
Table 2.9 STRINGdb Functional Enrichment of Selected Genes on Breast Cancer Data.....	31
Table 3.1 Cohort Demographics by FA MSI Status .....	50
Table 3.2 Original NGS Model Performance on Flagged Cohort Samples. ....	53
Table 3.3 Performance Metrics Across NGS Model Methods and Cohorts .....	53
Table 3.4 Updated NGS Model Performance on Flagged Cohort Samples.....	55
Table 3.5 Performance of NGS Model Methods by Cancer Type.....	56
Table 4.1 Four Component Model Parameters.....	65

## LIST OF FIGURES

Figure	Page
Figure 2-1 Density Plot of Simulated Noise Features .....	13
Figure 2-2 Rank of True Predictors by Scoring Method. ....	18
Figure 2-3 Rank of True Predictors by Scoring Method and Damping Parameter. ....	19
Figure 2-4 10-Fold CV AUCs for Simulated Data. ....	21
Figure 2-5 10-Fold CV AUCs for Simulated Data by Scoring Method and Classifier.....	22
Figure 2-6 10-Fold CV AUCs for Breast Cancer Data.....	24
Figure 2-7 10-Fold CV AUCs for Breast Cancer Data by Scoring Method and Classifier.....	25
Figure 2-8 Venn Diagram of Ranking Method Gene Selections on Breast Cancer Data.....	28
Figure 2-9 STRINGdb Networks of Selected Gene Sets on Breast Cancer Data .....	30
Figure 3-1 Example Microsatellite.....	36
Figure 3-2 Immune Checkpoint Inhibition of T-Cell Activation.....	38
Figure 3-3 Anti-PD-1 Antibody Aided T-Cell Activation .....	39
Figure 3-4 Diagram of NGS MSI Model Development Process.....	45
Figure 3-5 Sample Population Allele Frequencies by Cohort.....	52
Figure 3-6 Depiction of Final Model Selected on Training Data .....	54
Figure 3-7 Number of Loci Included in Model Versus Model Score.....	55
Figure 4-1 Four Component Density Plot.....	65
Figure 4-2 Special Case of Four component Density Plot.....	66
Figure 4-3 Comparison of ROC Curve Estimates for Non-Gaussian, Bimodal Data .....	71
Figure 4-4 Effects of Sample Size on Area Between ROC Curves.....	72

Figure	Page
Figure 4-5 Histogram of P-Values for Bimixt Fit of Proteomic Data .....	73



## CHAPTER 1

### INTRODUCTION

#### 1.1 Goal

The goal of this research is to motivate, develop, and test analytical tools useful for analysis of molecular oncology data. Recurring challenges in analyses of molecular oncology data include finding predictive biomarkers, accounting for tumor heterogeneity, and working with data sets with small sample sizes and large feature spaces. The following chapters each aim to address such challenges.

#### 1.2 Family Rank

When designing prediction models built with many features and relatively small sample sizes, feature selection methods can often overfit training data, leading to the selection of irrelevant features. Distinguishing between irrelevant features and features that are true predictors of a response variable can be difficult because, by chance, sets of irrelevant features may describe the response variable as equally well as sets of relevant features. The larger the ratio of features to sample size, the higher the probability irrelevant features will look relevant by chance. This presents a common challenge in many molecular oncology studies as there are often small sample sizes (e.g. cohorts of patients with specific subtypes of cancer) and large feature sets (e.g. tens of thousands of genes or proteins).

One way to potentially mitigate overfitting is to incorporate domain knowledge into the feature selection algorithm. Using such knowledge to weight features can force an algorithm to choose a relevant feature over an irrelevant one, even when empirically the two features are apparently equal at predicting the response variable. Moreover, domain knowledge that can be represented graphically may help select interacting features without having to test all pairwise interactions, which can be computationally expensive. For example, two features represented by connected nodes in a graph may be more likely to have a true interaction effect than two unconnected features.

Graphical domain knowledge is also applicable to many molecular oncology studies where protein-protein networks can affect the underlying biology of the data. Additionally, many publicly available databases have well annotated protein-protein networks. For example, the 'Search Tool for the Retrieval of Interacting Genes/Proteins' database (STRINGdb) contains graphical representation of millions of known and predicted protein-protein interactions ([Szklarczyk et al., 2015](#)).

Ranking algorithms such as Personalized Page Rank ([Page, Brin, Motwani, & Winograd, 1999](#)) and Gene Rank ([Morrison, Breitling, Higham, & Gilbert, 2005](#)) are two algorithms that can be used to weight features utilizing graphical knowledge. Both algorithms are based on the original page rank algorithm in which random walks are used to determine the probability (or rank) of nodes in a graph. A random walk begins on a randomly selected node, and moves from node to node along the edges of the graph. The more connected a node is, the more likely it is visited during the random walk. If edge weights are given as part of the graph structure, the probability that a step is taken along an edge is weighted by this factor. At each node there is also a probability, referred to as the damping factor, that the random walk along the current path will end, and a new starting node will be chosen at random. The probability that a node is visited as the random walks progress to convergence is that node's page rank.

In personalized page rank, the probability of landing on a node when abandoning a random walk is given by a unique probability for each node as opposed to a uniform probability across all nodes. This allows empirical feature scores to be incorporated into the personalization of the page rank algorithm by assigning the node probabilities based on these scores. In gene rank, the probability of each node is scaled by the empirical feature score.

Chapter 2 will present a novel feature ranking algorithm called 'Family Rank' which utilizes graphical domain knowledge to weight feature scores computed from empirical data. Family rank will then be compared to page rank and gene rank on simulated and real-world oncology data sets.

### 1.3 Population Bias in MSI

Microsatellite instability (MSI) is a prognostic biomarker utilized by clinicians to guide cancer treatment. In particular, immunotherapies have been approved by the United States Food and Drug Administration's (FDA) for treatment of tumors with high levels of MSI (MSI-H) ([Lemery, Keegan, & Pazdur, 2017](#)).

Historically, MSI has been detected in cancerous tumor tissue samples by performing fragment analysis (FA) on a panel of five representative genomic markers ([Boland et al., 1998](#)). More recently, next-generation sequencing (NGS) has been used to analyze thousands of microsatellite loci to detect MSI. NGS-based tests have been shown to improve the robustness and sensitivity of MSI detection ([Vanderwalde, Spetzler, Xiao, Gatalica, & Marshall, 2018](#)).

NGS MSI techniques typically rely on bioinformatics pipelines that align microsatellite locations in tumor DNA to a reference genome. Large numbers of microsatellite locations that do not match the reference genome is evidence of MSI. However, because the initial reference genome was created from a limited number of individuals with a heavy bias towards populations of European ancestry ([E pluribus unum, 2010](#); [Sherman et al., 2019](#)), this method of microsatellite alignment can be prone to population biases. Furthermore, findings from the 1000 Genomes Project indicate that variants, including microsatellites, can be specific to ancestral lines, and individuals of African ancestry have more normal germline variation relative to other ancestral lines ([Fan et al., 2019](#); [Genomes Project Consortium, 2010](#)).

In chapter 3, the hypothesis that this natural variation decreases the specificity of NGS MSI detection in patients of African/African American ancestry is tested, and an NGS-based diagnostic test aimed at minimizing the hypothesized bias is trained on a data set of 6,140 tumor specimen.

#### 1.4 Bimixt

Disease heterogeneity refers to the existence of varying genetic signatures observed across patients with a single disease. While a disease may present similarly across patients, the variation in underlying genomic mutational patterns can give rise to differing responses to therapies and differing medical prognoses.

Over the past several decades, molecular sequencing has revealed genetic subtypes of many heterogeneous diseases. A canonical example of disease heterogeneity is breast cancer which is commonly classified based on the presence or absence of three biomarkers: estrogen receptor (ER), progesterone receptor (PR), and the HER2 gene. ER and PR are hormone receptors (HR). Hormone receptor status has been linked to patient prognoses with ER+/PR+ patients showing significantly decreased mortality rates compared to patients with only one HR mutation, or no HR mutations ([Dunnwald, Rossing, & Li, 2007](#)). Furthermore, meta-analyses of data from 20 clinical trials has shown that response to the hormone therapy drug tamoxifen can significantly decrease mortality in patients with ER+ tumors, but has little to no effect in patients that are ER- ([Group, 2011](#)). Finally, targeted therapies that are engineered to target the HER2 gene have shown significant decreases in mortality of HER2+ breast cancer patients ([Slamon et al., 2001](#)).

While many disease subtypes have been classified, subtyping remains an active field of research, particularly within oncology. Detection of biomarkers associated with cancer status may indicate which genes play crucial roles in the molecular pathways associated with specific cancer subtypes and may be good candidates for targeted therapies.

Early phase biomarker discovery studies that aim at detecting potentially actionable cancer mutations often revolve around assaying as many markers as possible and trying to identify relevant markers from the larger population that can distinguish between subjects from two different categories. For example, a biomarker discovery study of a proteomic screening may look at the expression levels of thousands of proteins to try to find the proteins with the greatest ability to distinguish individuals with a disease (cases) from individuals without the disease (controls). A single protein's ability to distinguish cases from controls is determined by calculating sensitivity (true positive rate) and specificity (true negative rate) at various thresholds for the expression level. However, due to disease heterogeneity, finding such markers may be difficult as they may only be mutated in a small sub-sample of any given cohort. Thus, heterogeneity should be accounted for when assessing individual biomarker candidates.

Chapter 4 will present a novel method called 'bimixt' for fitting a mixture model to numeric measurements from heterogeneous populations. The chapter will cover estimation of the model, application of the model to real biomarker data, and utility of the model for assessing individual biomarker candidates.

## CHAPTER 2

### FAMILY RANK: A GRAPHICAL KNOWLEDGE INFORMED FEATURE RANKING ALGORITHM<sup>1</sup>

#### 2.1 Introduction

When designing prediction models built with many features and relatively small sample sizes, feature selection methods can often overfit training data, leading to the selection of irrelevant features. Distinguishing between irrelevant features and features that are true predictors of a response variable can be difficult because, by chance, sets of irrelevant features may describe the response variable as equally well as sets of relevant features. The larger the ratio of features to sample size, the higher the probability irrelevant features will look relevant by chance. One way to potentially mitigate this impact is to incorporate domain knowledge into the feature selection algorithm. Using such knowledge to weight features can force an algorithm to choose a relevant feature over an irrelevant one, even when empirically the two features are apparently equal at predicting the response variable. Moreover, domain knowledge that can be represented graphically may help select interacting features without having to test all pairwise interactions, which can be computationally expensive. For example, two features represented by connected nodes in a graph may be more likely to have a true interaction effect than two unconnected features.

In this chapter, a feature ranking algorithm called 'Family Rank' is presented which utilizes graphical domain knowledge to weight feature scores computed from empirical data (henceforth referred to as 'empirical feature scores'). The algorithm looks at each feature as a starting point to grow a family of features. Families are generated by iteratively selecting features that maximize a weighted score calculated from empirical feature scores and interaction scores (edge weights) with features previously added to the family. The final ranking for a feature is determined by summing a feature's family-weighted scores across all families in which it appears.

---

<sup>1</sup>The work in this chapter has been published in ([Saul & Dinu, 2021](#)): Saul, Michelle, and Valentin Dinu. "Family Rank: A graphical domain knowledge informed feature ranking algorithm." *Bioinformatics* (2021).

A simulated data set is used to demonstrate a scenario in which the family rank algorithm outperforms other state-of-the-art graph based ranking algorithms. An example from oncology is then used to explore a real-world application of family rank. The oncology domain is used as an illustrative example because it is common to have small sample sizes (e.g. cohorts of patients in clinical trials with specific subtypes of cancer), large feature sets (e.g. tens of thousands of genes or proteins), and graphical domain knowledge (e.g. protein-protein interactions).

## 2.2 Background

### 2.2.1 Empirical Feature Ranking

Empirical scores refer to any statistical method used to generate a score that measures the ability of a feature to predict an outcome based on empirical data. Ranks based on empirical feature scores without incorporating domain knowledge are used both as parameter inputs to the graphical ranking algorithms and as a baseline for comparison among ranking methods (referred to as 'Empirical Rank' method). In this chapter, the following empirical scoring methods are used to generate ranks:

1. Area under the receiver-operator characteristic (ROC) curve (AUC) ([Robin et al., 2011](#))
2. Absolute difference in group means ( $\Delta$  mean)
3. Absolute difference in group medians ( $\Delta$  med)
4. Earth mover's distance (EMD) ([Nabavi, Schmolze, Maitituoheti, Malladi, & Beck, 2016](#))

Features are ranked based on the empirical scores by ordering from highest to lowest score. Higher scores correspond to lower ranks. Empirical ranking methods used in this chapter correspond to statistical methods for scoring a feature's ability to classify categorical outcome variables, since that is the response type for both simulated data sets and oncology data sets analyzed. However, different empirical feature ranking methods can be employed for numerical features (for example Pearson's correlation) and plugged into the graphical ranking methods in the same manner as scores generated for categorical outcome data.

### 2.2.2 Graphical Ranking Algorithms

Ranking algorithms, such as Personalized Page Rank ([Page et al., 1999](#)) and Gene Rank ([Morrison et al., 2005](#)) may be used to weight empirical feature scores. Both algorithms are based on the original page rank algorithm in which random walks are used to determine the probability (or rank) of nodes in a graph. A random walk begins on a randomly selected node, and moves from node to node along the edges of the graph. The more connected a node is, the more likely it is visited during the random walk. If edge weights are given as part of the graph structure, the probability that a step is taken along an edge is weighted by this factor. At each node there is also a probability, referred to as the damping factor, that the random walk along the current path will end, and a new starting node will be chosen at random. The probability that a node is visited as the random walks progress to convergence is that node's page rank.

In personalized page rank, the probability of landing on a node when abandoning a random walk is given by a unique probability for each node as opposed to a uniform probability across all nodes. This allows empirical feature scores to be incorporated into the personalization of the page rank algorithm by assigning the node probabilities based on these scores. In gene rank, the probability of each node is scaled by the empirical feature score.

This chapter presents a novel ranking algorithm, referred to as 'Family Rank', that incorporates empirical feature scores and graphical domain knowledge. The algorithm looks at each feature as a starting point to grow a family of features. Families are generated by iteratively selecting features that maximize a weighted score calculated from empirical feature scores and interaction scores (edge weights) with features previously added to the family. The final ranking for a feature is determined by summing a feature's family-weighted scores across all families in which it appears.

### 2.2.3 Protein-Protein Networks

Graphical knowledge is required input for all graphical ranking algorithms. The graphical domain knowledge for oncology applications presented in this chapter was extracted from the 'Search Tool for the Retrieval of Interacting Genes/Proteins' database (STRINGdb), which



contains graphical representation of known and predicted protein-protein interactions ([Szklarczyk et al., 2015](#)). STRINGdb contains millions of interaction scores suggesting evidence of functional links between pairs of proteins.

Interaction scores are calculated by combining confidence scores from seven different sources of evidence as follows:

$$S = 1 - \prod_i (1 - S_i)$$

Where  $S_i$  is the confidence score for the  $i^{th}$  evidence type and an independence among evidence sources is assumed ([Mering et al., 2003](#)).

Confidence scores for the evidence types range between 0 and 1, and can be categorized as low confidence (<0.4), medium confidence (0.4 to 0.7), and high confidence (>0.7).

Three of the sources of evidence of functional links between proteins come from genomic context produced de novo in STRING and include ([Huynen, Snel, von Mering, & Bork, 2003](#); [Von Mering et al., 2005](#)):

1. **Neighborhood in the Genome:** Groups of genes that are frequently observed in each other's genomic neighborhood across different species
2. **Gene Fusions:** Genes that are sometimes fused into single open reading frames
3. **Cooccurrence Across Genomes:** Gene families whose occurrence patterns across genomes show similarities

The other four sources of evidence of functional links between proteins are imported from other databases ([Huynen et al., 2003](#); [Von Mering et al., 2005](#)) and include:

4. **Co-Expression:** Proteins whose genes are observed to be correlated in expression, across a large number of experiments
5. **Experimental/Biochemical Data:** Co-purification, co-crystallization, Yeast2Hybrid, Genetic Interactions, etc. as imported from primary sources

6. **Association in Curated Databases:** Known metabolic pathways, protein complexes, signal transduction pathways, etc. from curated databases
7. **Co-Mentioned in PubMed Abstracts:** Automated, unsupervised text mining searching for proteins that are frequently mentioned together

Genomic context evidence is based on systematic comparisons of genomes across multiple species. Functionally interacting proteins tend to be associated with each other within a genome. Therefore, genomes are searched for gene pairs with more evolutionary patterns in common than expected by chance.

Performance of predictions from imported evidence are benchmarked against a common reference set of gold standard associations. The Kyoto Encyclopedia of Genes and Genomes (KEGG) database is used as the gold standard.

## 2.3 Methods

### 2.3.1 Family Rank Algorithm

The family rank algorithm takes as input a vector of empirical scores ( $\vec{s} = \{s_1, \dots, s_n\}$ ) corresponding to a set of features ( $\vec{f} = \{f_1, \dots, f_n\}$ ) and a graph object ( $G$ ) in which nodes correspond to individual features and edge weights indicate strengths of interactions between pairs of features, denoted as  $I(f_i, f_j)$ .

The first step of the algorithm is normalization. If any empirical feature score is not between 0 and 1,  $\vec{s}$  is normalized. Likewise, if any edge weight in  $G$  is not between 0 and 1, the set of all edge weights is normalized. Normalization consists of shifting all values by -1 times the minimum value if any value is less than 0, and dividing all values by the maximum value if any values are greater than 1 after shifting.

The next step of the algorithm is to generate families of features. A family is generated for each feature in  $\vec{f}$  by iteratively updating and maximizing a weighted score vector  $\vec{w}_{i,j} = \{w_{i,j_1}, \dots, w_{i,j_n}\}$  where  $i$  indicates the iteration and  $j$  indicates the family was initiated by feature

$f_j$ . At iteration  $i$  for the family initiated by feature  $f_j$ , the weighted score for feature,  $f_k$  (e.g. the  $k^{\text{th}}$  element of  $\vec{w}_{i,j}$ ) is defined as:

$$w_{i,j_k} = \begin{cases} s_k, & i = 1, k = j \\ 0, & i = 1, k \neq j \\ 0, & i > 1, k = m \\ (1 - d) * s_k + d * I(f_k, f_m), & i = 2, k \neq m \\ (1 - d) * w_{i-1,j_k} + d * I(f_k, f_m), & i > 2, k \neq m \end{cases}$$

where

$$f_m = \arg \max_{f_k \in \vec{f}} \vec{w}_{i-1,j}$$

In the equation above,  $d$  is a user-defined parameter (referred to as the damping parameter for consistency with page rank terminology) between 0 and 1 that determines how much weight to give to the interaction score, and  $f_m$  is equal to the feature  $f_k$  that maximizes  $\vec{w}_{i-1,j}$ .

At each iteration, the feature that maximizes  $\vec{w}_{i,j}$  is selected (i.e. added to the family), and the weight for that feature is set to 0 for the next iteration. For the first iteration,  $\vec{w}_{i,j}$  is equal to the empirical score for the initiating feature and 0 for all other features.

For the following iterations, the weighted score for feature  $f_k$  is updated to the weighted average of the score for feature  $f_k$  from the previous iteration and the interaction score between  $f_k$  and the feature selected in the previous iteration.

Features are added to the family iteratively until at least one of two stopping criteria is met: a feature that has already been added to the family is selected again and/or the maximum weighted score is less than a predefined tolerance level (e.g. 10E-6).

Selected features are stored in an  $n \times n$  feature matrix denoted as  $\Lambda$  and the corresponding score ( $w_{i,j_k}$ ) for the selected feature is stored in an  $n \times n$  score matrix denoted as  $\Omega$ . Columns of the matrices represent families, and are populated by selected features for each family until the stopping criteria is met, as outlined above. Given a weighted score vector for the family initiated by feature  $f_j$  at iteration  $i$ , the  $i, j^{\text{th}}$  element of each matrix is defined as:

$$\Omega_{i,j} = \max(\vec{w}_{i,j})$$

$$\Lambda_{i,j} = \arg \max_{f_k \in \vec{f}} \vec{w}_{i,j}$$

While the maximum number of features that can be added to a family is  $n$  (i.e. all features are selected), fewer features may be added, in which case not all rows of the matrices  $\Lambda$  and  $\Omega$  will be completely filled.

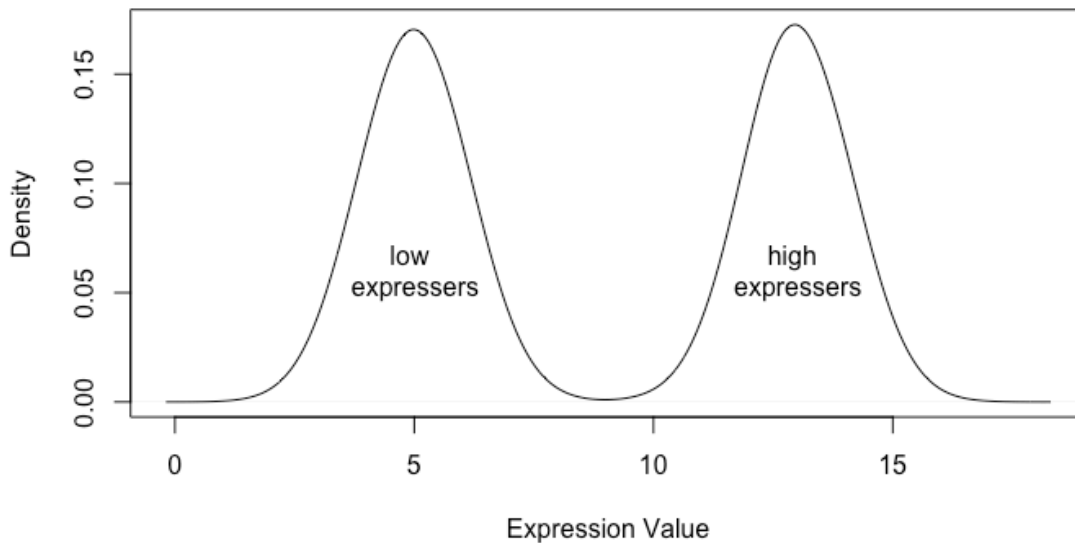
The final step of the algorithm is to generate a single score for each feature. The final score for a feature is defined as the sum of the weighted scores for all families in which that feature appears:

$$score_{f_k} = \sum (\Omega_{i,j} | \Lambda_{i,j} = f_k)$$

In other words, the final score for a feature  $f_k$  is the sum of the scores in  $\Omega$  corresponding to the locations in which  $f_k$  appears in  $\Lambda$ . An implementation of the algorithm can be found in the 'FamilyRank' package available on CRAN [<https://www.rdocumentation.org/packages/FamilyRank/versions/1.0>].

### 2.3.2 Data Simulation

To assess how family rank performs on complex data, simulations were used to emulate gene expression data in oncology. Features were simulated from bimodal Gaussian distributions. A two-component distribution was used to imitate high and low expressers of genes. Lower components were simulated from a  $N(\mu = 5, \sigma = 1)$  distribution and upper components were simulated from a  $N(\mu = 13, \sigma = 1)$  distribution. Features simulated as noise had a 50% probability of coming from either component. The density plot is shown in Figure 1.



*Figure 2-1 Density Plot of Simulated Noise Features*

Simulated responses were balanced, with half the samples labeled 'positive' and the other half labeled 'negative'. The response was perfectly defined by a set of 15 features (e.g. 'true predictors'). Positive samples were simulated to fall into one of 3 subtypes, each defined by 5 different features. Subtypes were used to imitate tumor heterogeneity and reflect the fact that tumorigenesis resulting in a single cancer type can be initiated by different gene sets.

Subtype 1 was defined as having features 1 through 3 and at least one of either features 4 or 5 simulated from the upper component of the bimodal Gaussian distribution. Subtype 2 was defined as having all of features 6 through 10 simulated from the upper component. Subtype 3 was defined as having features 11 through 14 simulated from the upper component and feature 15 simulated from the lower component. Simulations were performed so that all response subtypes were mutually exclusive. That is, no positive sample fit the criteria for more than one subtype. The logic defining subtypes was used to imitate the fact that both over-expressed and under-expressed genes can result in protein levels that are either too high or too low for cells to function properly. The fact that multiple features were used to define a subtype was used to imitate the fact that a low level of one protein can be counter-balanced by a higher level of a

different but similar protein, so that mutations in more than one gene may be required to have a functional impact on a cellular level.

Negative samples were simulated such that they did not fit the criteria of any of the positive subtypes. To ensure they did not fit subtype 1 criteria, at least one of features 1 through 3 and/or both features 4 and 5 were simulated from the lower component of the Gaussian distribution. To ensure they did not fit subtype 2 criteria, at least one of features 6 through 10 was simulated from the lower component. To ensure they did not fit subtype 3 criteria, at least one of features 11 through 14 was simulated from the lower component and/or feature 15 was simulated from the upper component.

Graphical domain knowledge for the subtypes was simulated such that each cluster of 5 features defining a subtype was fully connected by including all possible pairs of subtype features, and assigning them edge weights of 1. One million additional interactions were simulated among random pairs of all features to introduce noise into the graphical domain knowledge. Edge weights for the random interactions were sampled uniformly along the interval from 0 to 1. Duplicated interaction pairs were removed, so that final simulated domain knowledge graphs had slightly fewer than 1 million total interactions.

Ranking methods were directly compared on simulated datasets of varying sample sizes. Each sample had simulated values for 10,000 features, where features 1 through 15 were true predictors and the other 9,985 were noise. Code for generating the simulated data sets can be found in the 'FamilyRank' package available on CRAN [<https://www.rdocumentation.org/packages/FamilyRank/versions/1.0>].

### 2.3.3 Breast Cancer Data

Ranking methods were applied to breast cancer gene-expression data obtained from GEO (GSE16716). Data were collected as part of the Microarray Quality Control (MAQC)-II project ([Consortium, 2010](#)) and have previously been described ([Popovici et al., 2010](#)). Briefly, the data consist of 22,284 genes assayed across 230 breast cancer specimen. Breast cancer patients were given preoperative chemotherapy and were classified as either responders (no

residual invasive cancer was detected in breast or lymph nodes post therapy) or non-responders (residual invasive cancer was detected post therapy). As in the original analysis, the first 130 cases are used for training and the next 100 are used as an independent validation set.

Domain knowledge was extracted from the STRING database ([Mering et al., 2003](#)), which contains graphical representation of known and predicted protein-protein interactions. Interaction scores between two proteins are assigned a value between 0 and 1 based on evidence of functional links from multiple sources. Details on interaction score calculations and sources of evidence have been previously described ([Huynen et al., 2003](#); [Szklarczyk et al., 2015](#); [Von Mering et al., 2005](#)). The interaction scores used in analysis were downloaded from STRING version 11.0 for the homo sapiens species identifier 9606 (<https://string-db.org/cgi/download>).

Genes from the microarray expression data obtained from GEO were matched to the STRING network data by gene symbol. Genes not found in the expression array were removed from the domain knowledge. Remaining genes used for analysis included a total of 7,342,791 unique protein-protein interactions.

#### 2.3.4 Cross Validation

Cross validation was performed on both simulated data and the breast cancer data set obtained from the MAQC study. Fifty iterations of 10-fold cross validation were performed on both simulated and real-world data sets. All calculations were performed in R ([R Core Team, 2019](#)). Training folds were used to build models across multiple parameters including 4 types of empirical scoring methods, 4 ranking methods (3 using domain knowledge, and one using ranks from empirical scores to compare as a baseline), 3 damping factors, 9 feature selection sizes, and 3 classifiers. The parameters evaluated are listed in Table 2.1.

Table 2.1 Parameters Assessed by Cross Validation

Empirical Scoring Methods	Ranking Methods	Damping Factors	Feature Selection Size	Classifiers
Area under the receiver-operator characteristic (ROC) curve (AUC)	Empirical Rank	0.15	10 – 50 by 5	KNN Random Forest SVM
Absolute difference in group means ( $\Delta$ mean)	Family Rank	0.5		
Absolute difference in group medians ( $\Delta$ med)	Gene Rank	0.85		
Earth mover's distance (EMD)	Page Rank			

Within each training fold, a total of 1,296 prediction models were built, one for each combination of parameters. First, 4 sets of empirical scores were generated on training data. AUCs were calculated using the 'pROC' package ([Robin et al., 2011](#)) and EMD was calculated using the 'EMDomics' package ([Nabavi et al., 2016](#)).

Next, for each of the 4 sets of empirical scores, ranks were generated for the top 1,000 features. The top 1,000 features associated with each empirical scoring method were used for ranking as opposed to all 10,000 features in order to reduce computation time. The page rank algorithm was performed using functions provided in the 'igraph' package ([Csardi & Nepusz, 2006](#)), the gene rank algorithm was performed using R code adapted from the MATLAB code provided in the original publication ([Morrison et al., 2005](#)), and the family rank algorithm was performed using the 'FamilyRank' package ([Saul, 2021](#)). Family rank, gene rank, and page rank were run 3 times per empirical score set, one for each damping parameter assessed. Although empirical rank does not incorporate damping factors, to ensure equal number of classifiers were built per ranking method, 3 sets of the empirical rank were generated per empirical score set as well. Thus, a total of 48 ranked score sets were generated (12 per ranking algorithm).

Finally, a total of 27 classifiers were built for each of the 48 ranked score sets. For each ranked score set, 3 different classification methods were used to build classifiers incorporating 10



to 50 (sequenced by 5) of the top ranked features. Random forest classification was performed using functions provided by the 'randomForest' package ([Andy Liaw & Wiener, 2002](#)), SVM was performed using functions provided in the 'e1071' package ([Dimitriadou, Hornik, Leisch, Meyer, & Weingessel, 2008](#)), and KNN was performed using functions provided in the 'class' package ([Venables & Ripley, 2002](#)).

Additional parameters for classifiers were not optimized due to computation time, but were instead set to defaults found in the literature. For KNN, the default cluster size was set to 11, as was done in the MAQC analysis ([Popovici et al., 2010](#)). For RF, the default settings from the randomForest function in the 'randomForest' R package were used, including a tree size of 1,000 and a feature sample size at each tree split equal to the floor of the square root of the number of features. For the SVM classifier, the default settings from the svm function in the 'e1071' R package were used, including a radial kernel, a gamma value equal to 1 divided by the number of features, and a constant of regulation parameter equal to 1.

For each training fold, class predictions were made for the 1,296 training models on testing data from the held-out fold. Binary predictions were aggregated across the 10 folds to calculate confusion matrices and associated statistics. Class probabilities were aggregated across the 10 folds to calculate AUCs. All statistics were then averaged over 50 iterations.

## 2.4 Results

### 2.4.1 Simulated Data

Data sets were simulated for sample sizes ranging from 50 to 500 to compare the ranks of true predictors (1 through 15) among ranking algorithms. For each sample size, 10 data sets were simulated, and ranks were generated for each ranking method evaluated at damping parameters ranging from 0.15 to 0.85 and empirical scoring methods 1 through 4 (as discussed in the methods section). The number of true predictors ranked in the top 15 features were averaged over the 10 simulations. Results corresponding to the best performing damping parameter per ranking method are shown in Figure 2-2 and results stratified by damping parameter and empirical scoring method are shown in Figure 2-3.

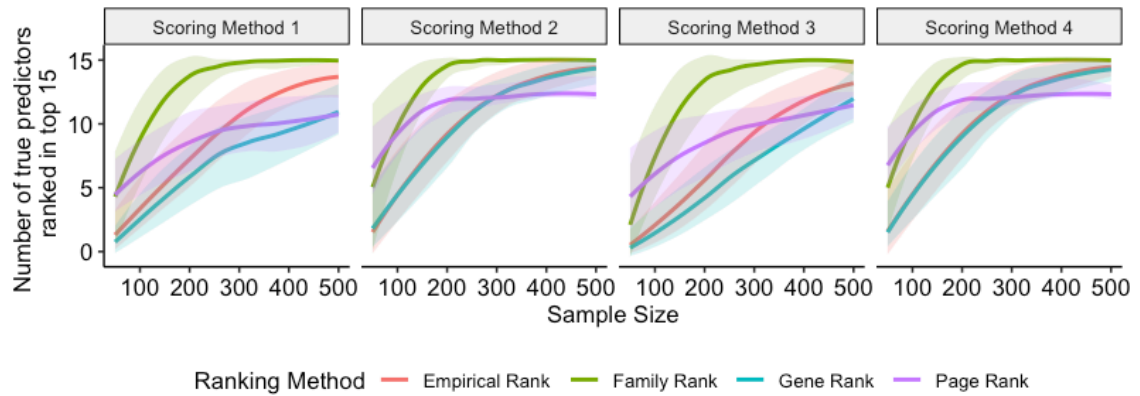
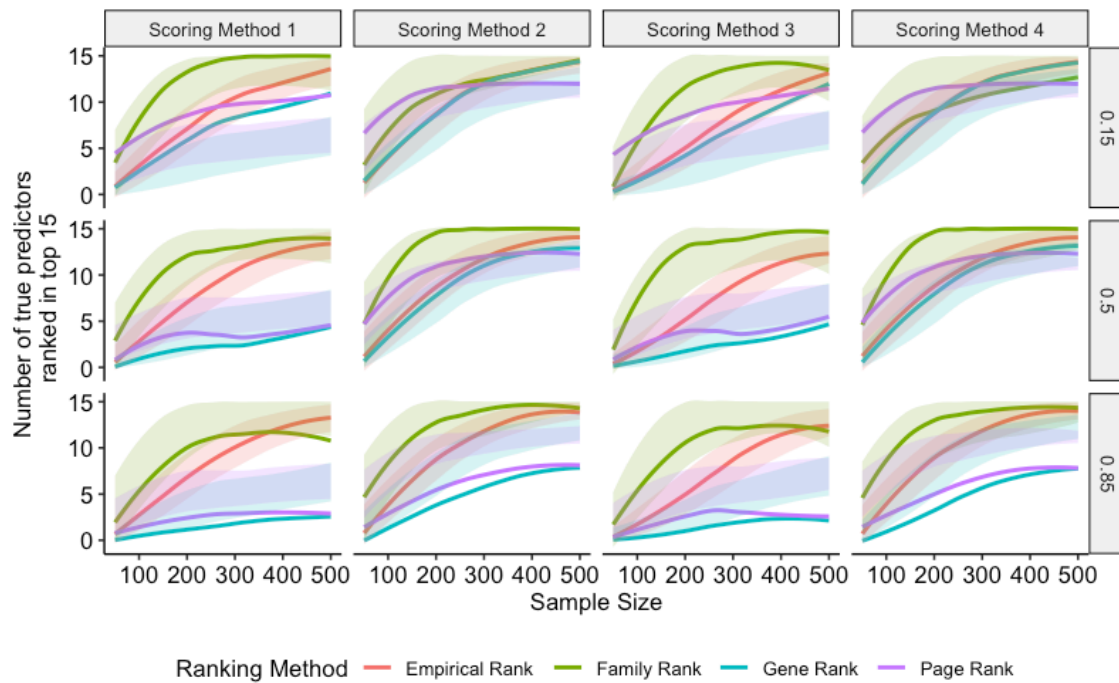


Figure 2-2 Rank of True Predictors by Scoring Method.

Ranking method results corresponding to best performing damping parameter. Lines indicate the number of true predictors ranked in the top 15 averaged over 10 simulations. Ribbons depict the maximum and minimum number of true predictors in the top 15 across simulations. Scoring methods analyzed include AUC (1), difference in means (2), difference in medians (3), and EMD (4).



*Figure 2-3 Rank of True Predictors by Scoring Method and Damping Parameter.*

Figure shows ranking method results faceted by scoring method and by damping parameters. Lines indicate the number of true predictors ranked in the top 15 averaged over 10 simulations. Ribbons depict the maximum and minimum number of true predictors in the top 15 across simulations. Scoring methods analyzed include AUC (1), difference in means ( $\Delta$  mean) (2), difference in medians (3), and EMD (4).

To compare the raw ranks of the true predictors among ranking methods, a data set of size 200 was simulated. Features were ranked by each ranking method at damping parameters ranging from 0.15 to 0.85 and across all 4 scoring methods. The results for the scoring method and damping parameters that yielded the best (lowest) average rank for true predictors for each ranking method is shown in Table 2.2.

Table 2.2 Ranks of True Predictors by Ranking Method

True Predictor	Family Rank	Page Rank	Gene Rank	Empirical Rank
1	8	1	306	327
2	13	8	4	4
3	7	3	7	10
4	>1000	>1000	>1000	>1000
5	6	45	8	5
6	1	2	10	12
7	4	15	9	8
8	5	9	6	6
9	3	4	249	271
10	2	13	468	528
11	10	5	11	9
12	14	7	1	1
13	12	18	2	2
14	9	6	3	3
15	11	244	13	11

**Table 2.2:** Ranks of true predictors were generated on a simulated data set with a sample size of 200. The damping parameters and empirical scoring methods used for generating the ranks were determined by selecting the parameters that minimized the average rank of features 1 through 15.

In addition to the ranks of the true predictors, the ability of selected features to classify the response variable was assessed via 50 replicates of 10-fold cross-validation (CV). For each replicate, a unique data set with a sample size of 200 was simulated. Classification models were built on training folds for multiple combinations of parameters and classifiers. Results for parameters achieving the highest average cross-validated AUCs for each ranking method are shown in Table 2.3.

Table 2.3 Cross Validation Classification Performance on Simulated Data

Ranking Method	Parameters <sup>1</sup>				Average CV Performance (SD) <sup>2</sup>									
	Classifier	Scoring Method	Damping Factor	# Predictors	TP	TN	FP	FN	AUC	Accuracy (%)	Sensitivity (%)	Specificity (%)	PPV (%)	NPV (%)
Family Rank	SVM	2	0.5	15	91.1 (3.5)	89.9 (2.7)	10.1 (2.7)	8.9 (3.5)	0.97 (0.02)	90.5 (2.8)	91.1 (3.5)	89.9 (2.7)	90.1 (2.6)	91 (3.3)
Page Rank	SVM	4	0.15	15	87.3 (3.6)	85.8 (3.8)	14.2 (3.8)	12.7 (3.6)	0.94 (0.02)	86.5 (3.3)	87.3 (3.6)	85.8 (3.8)	86.1 (3.5)	87.1 (3.5)
Gene Rank	SVM	4	0.15	10	73.6 (6)	74.3 (5.8)	25.7 (5.8)	26.4 (6)	0.81 (0.06)	74 (5.4)	73.6 (6)	74.3 (5.8)	74.2 (5.5)	73.9 (5.5)
Empirical Rank	SVM	4	N/A	10	73.8 (6.1)	74.8 (5.4)	25.2 (5.4)	26.2 (6.1)	0.82 (0.06)	74.3 (5.3)	73.8 (6.1)	74.8 (5.4)	74.5 (5.3)	74.1 (5.5)

<sup>1</sup>Parameters that resulted in the highest average AUCs for each ranking method are displayed.

<sup>2</sup>Statistics are averaged over 50 replicates of 10-fold CV. For each replicate, probabilities and predictions on samples in held-out folds were used to calculate CV statistics.

AUCs for the optimal performing parameters for each ranking method are shown in

Figure 2-4, and AUCs faceted by scoring method and classifier are shown in Figure 2-5.

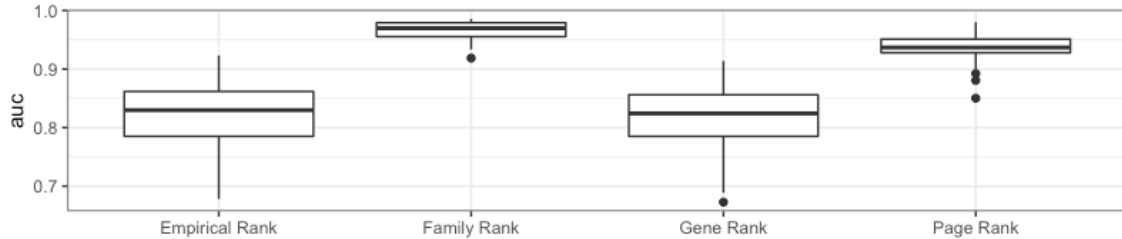


Figure 2-4 10-Fold CV AUCs for Simulated Data.

Ranking method CV results corresponding to best performing parameters. AUCs were calculated for 50 replicates of 10-fold CV. For each replicate, class probabilities were calculated for samples in held-out folds. Held-out probabilities from each fold were then aggregated and used to calculate a single AUC.

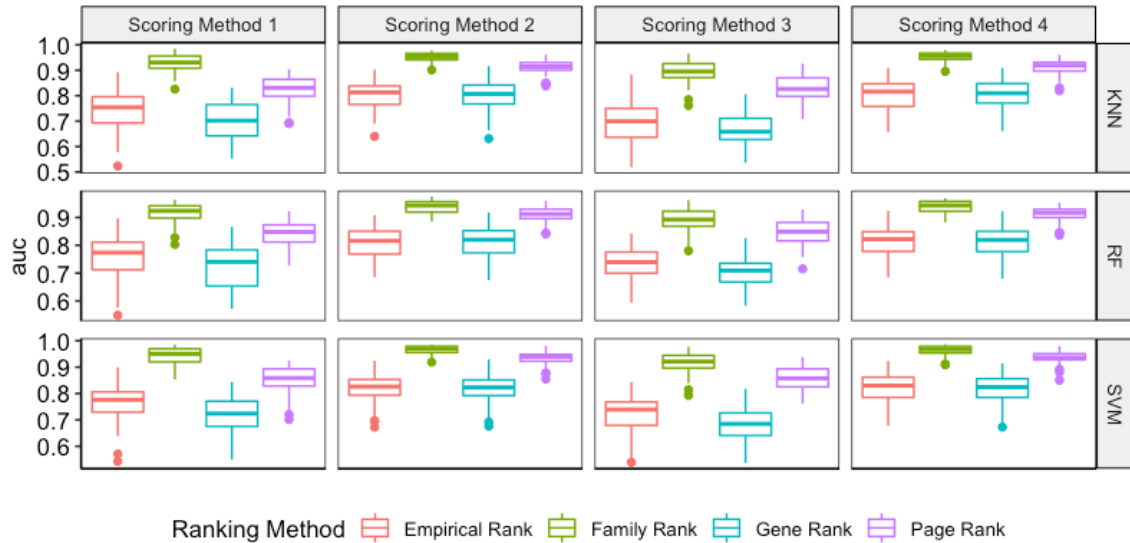


Figure 2-5 10-Fold CV AUCs for Simulated Data by Scoring Method and Classifier.

Figures show ranking method CV results stratified by empirical scoring methods and classifiers. AUCs were calculated for 50 replicates of 10-fold CV. For each replicate, class probabilities were calculated for samples in held-out folds. Held-out probabilities from each fold were then aggregated and used to calculate a single AUC. Scoring methods analyzed include AUC (1), difference in means (2), difference in medians (3), and EMD (4).

Independent testing was also performed on simulated data. Optimal parameters found during cross validation were used to build prediction models on a simulated data set with 75 'positives' and 75 'negatives'. Another data set containing 25 'positive' and 25 'negative' samples was simulated for independent testing. Results are shown in Table 2.4.

Table 2.4 Independent Validation Performance on Simulated Data

Ranking Method	Parameters <sup>1</sup>				Performance									
	Classifier	Scoring Method	Damping Factor	# Predictors	TP	TN	FP	FN	AUC	Accuracy (%)	Sensitivity (%)	Specificity (%)	PPV (%)	NPV (%)
Family Rank	SVM	2	0.5	15	23	22	3	2	0.96	90	92	88	88	92
Page Rank	SVM	4	0.15	15	22	22	3	3	0.95	88	88	88	88	88
Gene Rank	SVM	4	0.15	10	19	22	3	6	0.89	82	76	88	86	79
Empirical Rank	SVM	4	N/A	10	20	22	3	5	0.89	84	80	88	87	81

<sup>1</sup>Optimal parameters from training cross validation were used to build models on the entire training set. These models were then evaluated on the independent testing set.

#### 2.4.2 Breast Cancer Data

Ranking methods were applied to breast cancer gene-expression data obtained from the Microarray Quality Control (MAQC)-II project to predict response to chemotherapy. The original publication assessed cross-validation results on a data set consisting of 33 responders and 97 non-responders. This set is used here to assess cross-validation results as well. Cross validation is performed using the same methods as the simulated data. Results for parameters achieving the highest average cross-validated AUCs for each ranking method are shown in Table 2.5.

Table 2.5 Cross Validation Classification Performance on Breast Cancer Data

Ranking Method	Parameters <sup>1</sup>				Average CV Performance (SD) <sup>2</sup>									
	Classifier	Scoring Method	Damping Factor	# Predictors	TP	TN	FP	FN	AUC	Accuracy (%)	Sensitivity (%)	Specificity (%)	PPV (%)	NPV (%)
Family Rank	RF	2	0.5	40	14.9 (1.5)	88.1 (1.3)	8.9 (1.3)	18.1 (1.5)	0.84 (0.01)	79.2 (1.6)	45.2 (4.4)	90.8 (1.3)	62.6 (4.3)	83 (1.2)
Page Rank	RF	1	0.85	25	14.6 (1.6)	88 (1.4)	9 (1.4)	18.4 (1.6)	0.83 (0.01)	78.9 (1.8)	44.2 (4.8)	90.7 (1.4)	61.8 (4.9)	82.7 (1.3)
Gene Rank	KNN	1	0.15	25	17.8 (2.1)	81.8 (2.2)	15.2 (2.2)	15.2 (2.1)	0.8 (0.02)	76.7 (2.3)	54.1 (6.5)	84.4 (2.3)	54.1 (4.6)	84.4 (1.9)
Empirical Rank	KNN	1	N/A	40	16.2 (2)	81.6 (2.2)	15.4 (2.2)	16.8 (2)	0.81 (0.02)	75.3 (2.5)	49.1 (6.2)	84.2 (2.2)	51.4 (5.4)	83 (1.8)

<sup>1</sup>Parameters that resulted in the highest average AUCs for each ranking method are displayed.

<sup>2</sup>Statistics are averaged over 50 replicates of 10-fold CV. For each replicate, probabilities and predictions on samples in held-out folds were used to calculate CV statistics.

AUCs for the optimal performing parameters for each ranking method are shown in

Figure 2-6. AUCs faceted by scoring method and classifier are shown in Figure 2-7.

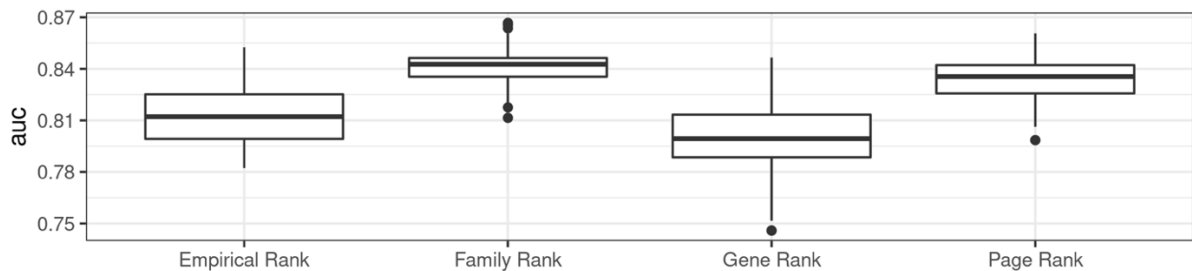
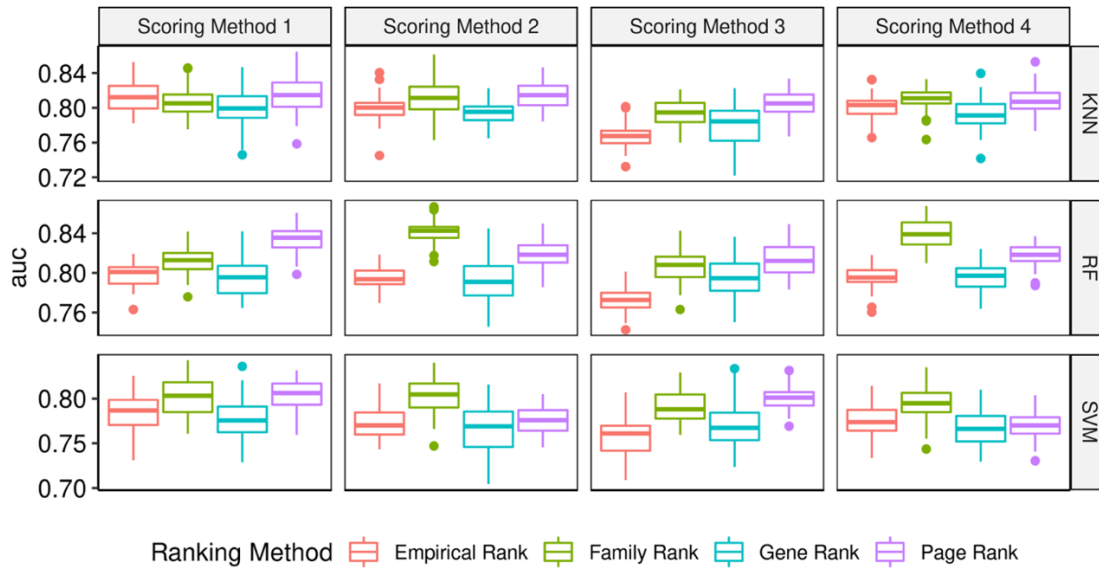


Figure 2-6 10-Fold CV AUCs for Breast Cancer Data.

Ranking method CV results corresponding to best performing parameters. AUCs were calculated for 50 replicates of 10-fold CV. For each replicate, class probabilities were calculated for samples in held-out folds. Held-out probabilities from each fold were then aggregated and used to calculate a single AUC.





*Figure 2-7 10-Fold CV AUCs for Breast Cancer Data by Scoring Method and Classifier.*

Figures show ranking method CV results stratified by empirical scoring methods and classifiers. AUCs were calculated for 50 replicates of 10-fold CV. For each replicate, class probabilities were calculated for samples in held-out folds. Held-out probabilities from each fold were then aggregated and used to calculate a single AUC. Scoring methods analyzed include AUC (1), difference in means (2), difference in medians (3), and EMD (4).

The original publication assessed optimal results on an independent data set consisting of 15 responders and 85 non-responders. This set is used here to assess independent test results using the optimal parameters from CV. Results on independent data are shown in Table 2.6.

Table 2.6 Independent Validation Performance on Breast Cancer Data

Ranking Method	Parameters <sup>1</sup>				Performance									
	Classifier	Scoring Method	Damping Factor	# Predictors	TP	TN	FP	FN	AUC	Accuracy (%)	Sensitivity (%)	Specificity (%)	PPV (%)	NPV (%)
Family Rank	SVM	2	0.5	40	23	22	3	2	0.96	90	92	88	88	92
Page Rank	SVM	4	0.15	25	22	22	3	3	0.95	88	88	88	88	88
Gene Rank	SVM	4	0.15	25	19	22	3	6	0.89	82	76	88	86	79
Empirical Rank	SVM	4	N/A	40	20	22	3	5	0.89	84	80	88	87	81

<sup>1</sup>Optimal parameters from training cross validation were used to build models on the entire training set. These models were then evaluated on the independent testing set.

Features chosen by each ranking method using optimal parameters from CV results are shown in Table 2.7. Overlap of selected genes by ranking methods is shown in Figure 2-8

Table 2.7 Selected Features by Ranking Method

Family Rank		Page Rank	Gene Rank	Empirical Rank	
ADCY9	ESR1.3	AR.1	ACSM1	AGR2	IGF1R.1
AGR2	FOXA1	CCNA2.1	ASPM	AMFR.1	IGFBP4
CCND1.1	FYN.2	CCNB2	BTG3	ARL3	MAPT
CCNE1	IGF1R.1	CCND1.1	CDKN2A.1	BTG3	MAPT.1
CDC20	IRS1	CDC20	CHMP2A	BTG3.1	MAPT.3
CDCA8	ITPR1.1	CDKN2A	COMP	BUB1	MCM3
CENPE	KIF2C	ENO1.2	COX7C.1	CA12	MCM5
CENPF.1	KIT	ESR1.3	CYP2B7P	CA12.3	MCM5.1
COL10A1	MAD2L1	EZH2	IFT46	CA12.4	MELK
COL1A2	MYB	GLI3	IKBKB.1	CA12.5	METRNL
COL9A3	NDC80	IGF1R.1	KPNA2	CTSV	MLPH
CXCL1	NPY1R	KIT	LCMT2	DAPK1	MYO5C
CXCL10	PGR	MAD2L1	MAN2B2	E2F3.1	NKAIN1
CXCL11.1	PLCB4.1	MCM3	MYOF	ESR1	PADI2
CXCL12.1	PLCG2	MCM5.1	NKAIN1	ESR1.8	SLC7A8
CXCL13	PLK1 PTGER3. 3	MCM7.1	PLPBP.1	GAMT	SLC7A8.1
CXCL3		MYB	PRSS23	GATA3.1	TBC1D9
CXCL5.1	SYK	MYBL2	RAB11FIP1	GATA3.2	TTK
CXCL8	TFF1	NDC80	RABEP1.1	GFRA1	VAV3.1
CXCR4.2	XBP1	NOTCH1	RARRES1. 1	IFT46	ZNF688
		PGR	RFWD3		
		PLK1	RRM2.1		
		RAD51.1	SKP1		
		SKP1	UGDH		
		VEGFA.1	X215304_at		

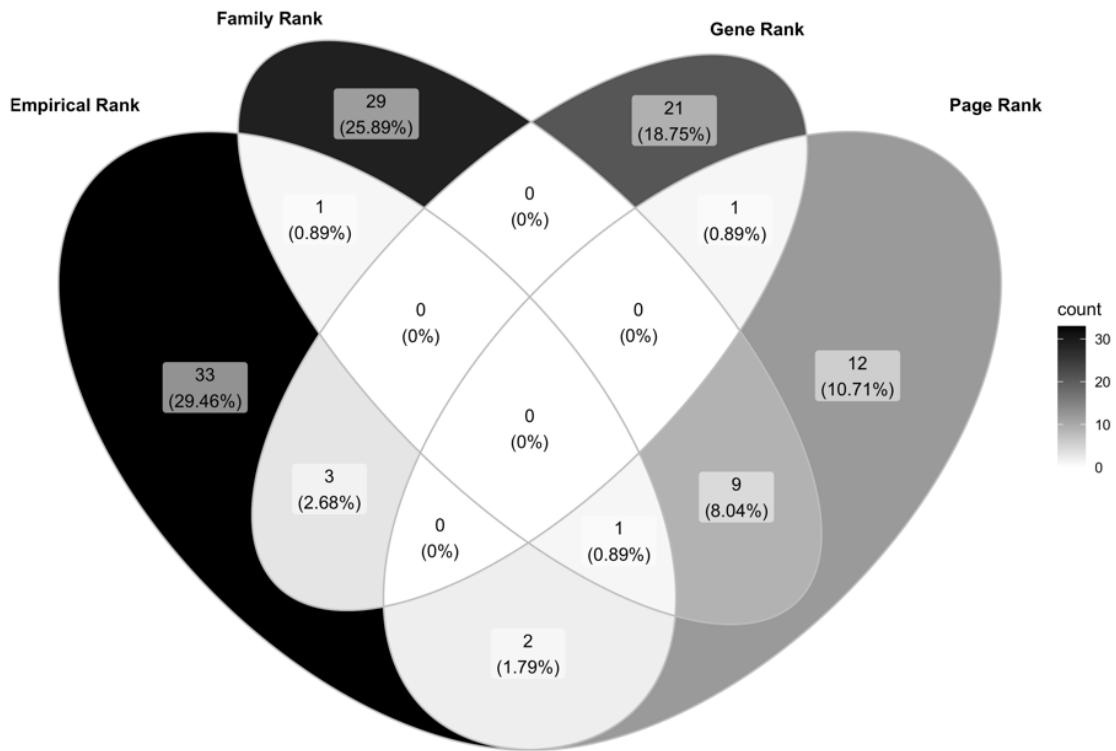
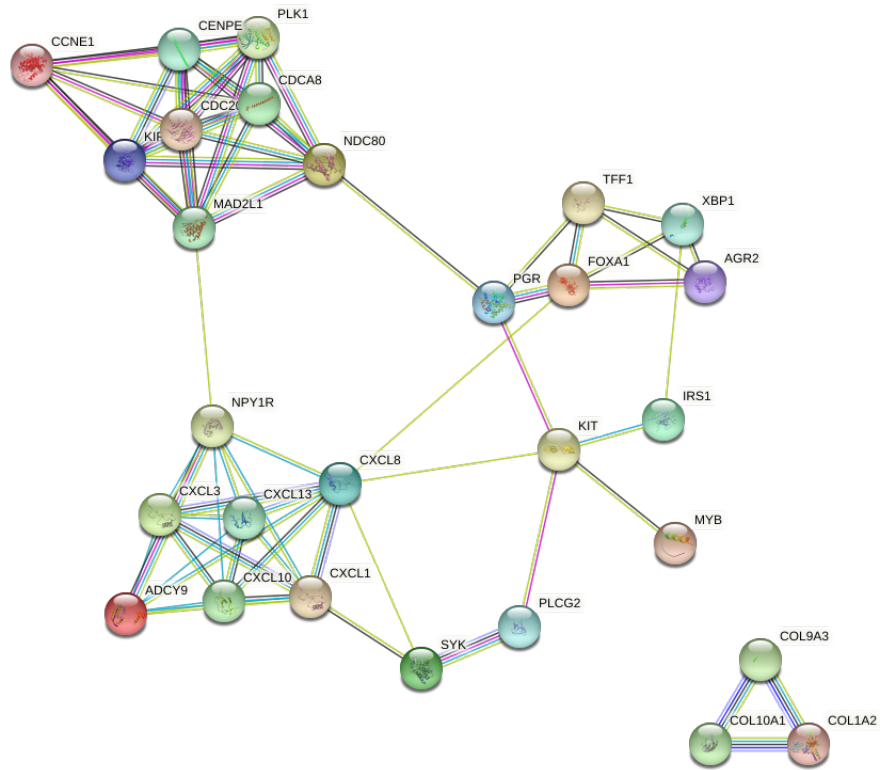


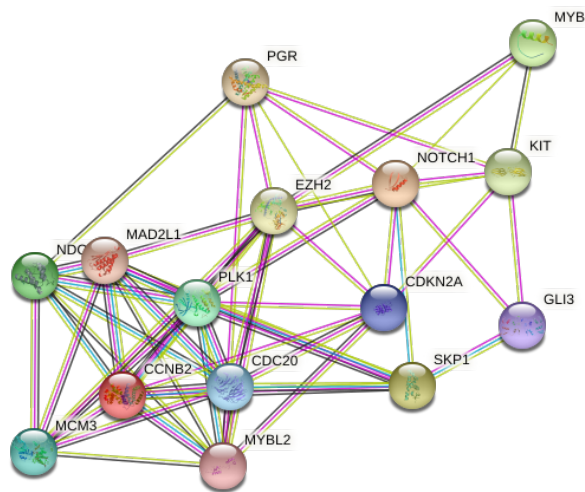
Figure 2-8 Venn Diagram of Ranking Method Gene Selections on Breast Cancer Data

Genes selected by each method were analyzed using STRINGdb ([string-db.org](http://string-db.org)). The networks for gene sets selected by each ranking method are shown in Figure 2-9. Network analysis summaries are shown in Table 2.8 and functional enrichment analysis summaries are shown in Table 2.9.

### A. Family Rank



### B. Page Rank



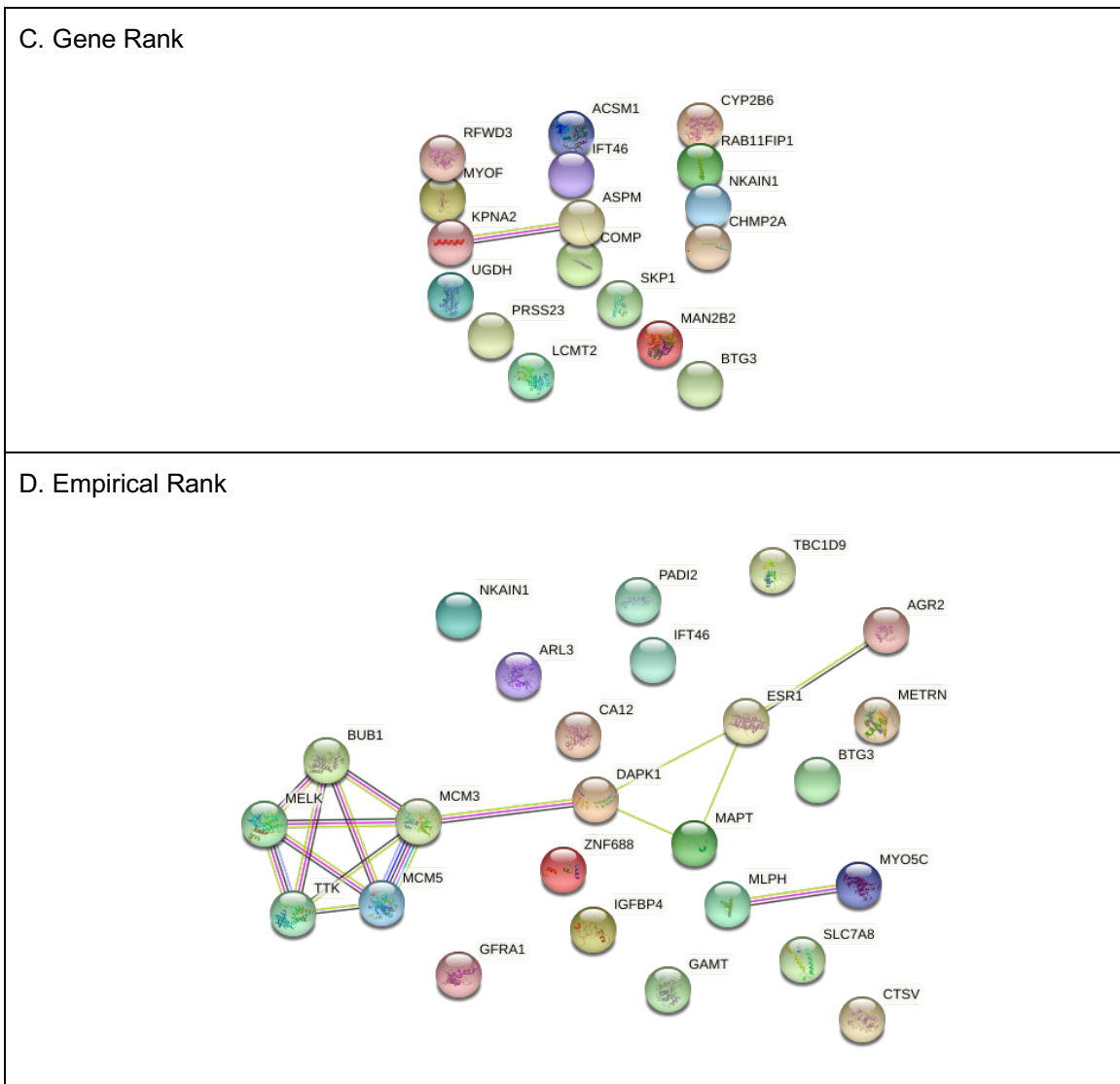


Figure 2-9 STRINGdb Networks of Selected Gene Sets on Breast Cancer Data

From top to bottom, the figure shows the networks of selected genes on the breast cancer data set for Family Rank (A), Page Rank (B), Gene Rank (C), and Empirical Rank (D).

Table 2.8 STRINGdb Network Statistics of Selected Genes on Breast Cancer Data

Ranking Method	Number of Nodes	Number of Edges	Average Node Degree	Expected number of Edges	PPI Enrichment P-value
Family Rank	28	71	5.07	26	2.43e13
Page Rank	15	55	7.33	13	<1e-16
Gene Rank	17	1	0.118	1	0.699
Empirical Rank	25	16	1.28	4	8.71e-6

Table 2.9 STRINGdb Functional Enrichment of Selected Genes on Breast Cancer Data

Biological Process (Gene Ontology)				
Ranking Method <sup>1</sup>	Description	Count in Network	Strength <sup>2</sup>	False Discovery Rate
Family Rank	attachment of mitotic spindle microtubules to kinetochore	3 of 10	2.32	3.69E-05
	positive regulation of mitotic cell cycle spindle assembly checkpoint	2 of 7	2.3	0.0011
	epithelial cell maturation	3 of 16	2.12	8.25E-05
	regulation of T cell chemotaxis	2 of 11	2.1	0.002
	positive regulation of B cell differentiation	2 of 13	2.03	0.0024
Page Rank	positive regulation of mitotic cell cycle spindle assembly checkpoint	2 of 7	2.57	0.00063
	negative regulation of ubiquitin protein ligase activity	2 of 10	2.42	0.00089
	regulation of ubiquitin protein ligase activity	4 of 23	2.36	9.73E-07
	mitotic nuclear envelope disassembly	2 of 12	2.34	0.0012
	positive regulation of ubiquitin protein ligase activity	2 of 13	2.3	0.0012
Empirical Rank	cytoskeleton-dependent intracellular transport	4 of 151	1.32	0.0457
	mitotic cell cycle process	6 of 564	0.92	0.0457
KEGG Pathways				
Ranking Method <sup>1</sup>	Description	Count in Network	Strength <sup>2</sup>	False Discovery Rate
Family Rank	Regulation of lipolysis in adipocytes	3 of 53	1.6	0.00072
	Legionellosis	3 of 54	1.59	0.00072
	Oocyte meiosis	6 of 116	1.56	2.18E-06
	Epithelial cell signaling in Helicobacter pylori infection	3 of 66	1.5	0.0012
	IL-17 signaling pathway	4 of 92	1.48	0.00018
Page Rank	Cell cycle	7 of 123	1.87	1.43E-10
	Oocyte meiosis	6 of 116	1.83	5.93E-09
	Progesterone-mediated oocyte maturation	4 of 94	1.74	9.26E-06
	p53 signaling pathway	2 of 68	1.58	0.0068
	HTLV-I infection	6 of 250	1.5	3.42E-07
Empirical Rank	DNA replication	2 of 36	1.64	0.0123
	Cell cycle	4 of 123	1.41	0.00044

<sup>1</sup>No significant functional enrichment was detected in Gene Rank.  
<sup>2</sup>Where more than 5 functional enrichments were found, only the 5 strongest are shown.

## 2.5 Discussion

Several feature-ranking methods were compared using extensive simulation and real-world data to address whether using prior or domain knowledge in conjunction with empirical feature scores can reduce the number of samples necessary to detect true predictors and/or whether better predictions can be made.

Simulations suggest that in scenarios with similar complexity where pertinent domain knowledge is available, family rank can significantly decrease the number of samples required to find true predictors. Figure 2-2 shows that family rank took 2 to 3-fold fewer samples compared to other ranking methods to find all relevant features of the simulated data. It is important to note, however, that the complexity of the simulated data may not accurately reflect the complexity of real world data. For example, in the simulated data, 15 features perfectly defined the response variable, but in the breast cancer data analysis, the fewest number of features selected for classifiers was 25. Therefore, simulations with more predictive features may be worth exploring. Additionally, Figure 2-9 A suggests that family rank identified 4 potentially relevant clusters (e.g. families), whereas the simulated data only utilized 3 subtypes. In the simulated data, all subtypes were also defined by 5 features, whereas the number of features in the families in Figure 2-9 A vary by cluster. Therefore, simulated data with varying numbers of true predictors per subtypes, and more numbers of subtypes are worth analyzing as well. Additionally, there was no connectivity between the true predictors and noise predictors in the simulated data set. The simulated domain knowledge was therefore probably overly optimistic and easier to pick out of a data set than the real world networks which had interactions between true predictors and noise predictors. Finally, the edge weights in the simulated data were all 1 between the true predictors of each subtype, which was also overly optimistic. Therefore, more connected graphs, and lower edge weights between true predictors should be considered in future analyses

Table 2.2 shows that with a sample size of 200, the empirical method alone ranked 11 out of the 15 true predictors in the top 15. Features 1, 4, 9, and 10 were not in the top 15. Family



rank was able to bump 3 of these features (1, 9, and 10) up to the top 15. Page rank was also able to bump these 3 feature up into the top 15. However, page rank also bumped the true feature 15 out of the top 15. Meanwhile, the difference in feature rankings of the true predictors between the empirical method and gene rank was very modest, and no additional true predictors moved into the top 15 with gene rank.

Both simulated and real-world data showed modest increases in cross-validated performance measures as depicted in figures 2-2 and 2-3. It also showed modest increase in independent test data performance as shown in tables 2.4 and 2.7. While the increases in performance are modest, if true patterns are discovered, the results can have additional benefits, such as highlighting potential areas for further study in fields where there are huge numbers of features. Additionally, even modest improvements in performance can have real world consequences. For example, in precision medicine, even small improvements can translate to many more patients receiving appropriate therapies over time. Moreover, analyses of gene features by pathways may reveal genetic associations between genes on an expression level that are novel.

Ranking methods did not select many overlapping genes in the gene sets selected on the breast cancer data. Figure 2-8 shows that the largest overlap was between Family Rank and Page Rank and included genes CCND1.1, CDC20, ESR1.3, KIT, MAD2L1, MYB, NDC80, PGR, and PLK1. These genes belong to the oocyte meiosis, progesterone-mediated oocyte maturation, cell cycle, and breast cancer pathways ([Kanehisa, 2002](#)). The relationship between cell cycles and chemotherapy is well-documented in the literature ([Lind, 2008](#)), thus providing a biological impetus for these genes to predict response to chemotherapies.

To understand why there was little overlap between ranking methods, STRINGdb enrichment analyses were performed on the gene sets selected by ranking methods. Page rank had the highest enrichment value ( $p < 1e-16$ ). This suggests that page rank had the most closely connected genes selected, which makes sense as page rank emphasizes the connectivity between nodes when selecting features. This is also visually apparent in figure 2-9 B which

shows many edges between all selected features. Family rank had the second highest enrichment value ( $p = 2.43e-13$ ). Additionally, it is apparent from figure 2-9 A that the family rank algorithm selected more distinctive families of networks. This is shown by the several edges between clusters of genes, but relatively few edges between the clusters. This is expected as Family Rank ranks each family independently. That is, the connectivity of one cluster (or family) does not affect the ranks of another family. This is ideal if researchers are looking for heterogeneous subtypes where many features may be related within a single subtype, but a different subtype maybe driven by a different cluster (or family). Empirical Rank had the 2<sup>nd</sup> least enrichment ( $p = 8.71e-6$ ). This shows that the assay technologies used did empirically pull out some enriched pathways, but that there were several unconnected genes that also had high empirical scores. Finally, gene rank was the least enriched set, and in fact had no significant enrichment detected ( $p = 0.699$ ). Gene Rank is the only method that analyzes the domain knowledge completely independently of the empirical data results. As opposed to page rank and family rank, which both analyze the empirical and domain knowledge together, gene rank does a sequential analysis where the domain knowledge is ranked, the empirical data scores are ranked, and the two are averaged. The results suggest that this approach results in less enriched pathways.

While this research has demonstrated that ranking methods utilizing domain knowledge have the potential to benefit prediction models and decrease the number of samples required for successful feature-selection, the methods must be used cautiously. There are many scenarios where ranking methods can hurt overall predictions. This could be the case if the domain knowledge contains information that is not relevant to the outcome being predicted. In this case, the ranks of irrelevant features may be increased and classification performance may suffer. In order to successfully utilize a ranking method, graphical domain knowledge, empirically measured variables, and measured outcomes must all complement each other.

## CHAPTER 3

### POPULATION BIAS IN SOMATIC MEASUREMENTS OF MICROSATELLITE INSTABILITY<sup>2</sup>

#### 3.1 Introduction

Microsatellite instability (MSI) is a key secondary effect of a defective DNA mismatch repair mechanism resulting in incorrectly replicated microsatellites in many malignant tumors. Historically, MSI has been detected in cancerous tumor tissue samples by performing fragment analysis (FA) on a panel of five representative genomic markers. More recently, next-generation sequencing (NGS) has been used to analyze thousands of microsatellite loci to detect MSI. NGS-based tests have been shown to improve the robustness and sensitivity of MSI detection. However, they can be prone to population biases if NGS results are aligned to a single reference genome instead of patient-matched normal tissue. In this chapter, an NGS-based diagnostic test that utilized 7,317 microsatellite loci to determine MSI status is shown to have an increased rate of false positives detected in patients of African ancestry as compared to an FA-based MSI diagnostic test. This bias was then minimized by training a modified calling model on NGS data that utilized 2,011 microsatellite loci. With these adjustments 100% (95% CI: 89.1% to 100%) of African ancestry patients in an independent validation test were called correctly using the updated model. This poses not only a significant technical improvement but also has an important clinical impact on directing immune checkpoint inhibitor therapy.

#### 3.2 Background

##### 3.2.1 Mismatch Repair

DNA mutations in which base pairs are inserted or deleted from a strand of DNA can occur from errors during DNA replication or recombination, from damage caused by natural disruptions such as byproducts of metabolic processes, or from damage caused by environmental disruptions such as radiation or ultraviolet (UV) light. Mismatch repair (MMR) refers to the natural

---

<sup>2</sup>The work in this chapter is published in ([Saul et al., 2020](#)): Saul, Michelle, et al. "Population bias in somatic measurement of microsatellite instability status." *Cancer medicine* 9.17 (2020): 6452-6460.

molecular process for repairing these types of DNA mutations. In a healthy genome, MMR proteins will detect these mutations and recruit DNA enzymes to correct them.

Josef Jiricny's 2006 review of mismatch-repair systems ([Jiricny, 2006](#)) provides a detailed outline of how a proficient mismatch-repair (pMMR) system functions. Briefly, the complexes involved in MMR initiate a chain reaction that includes four major events:

1. Detection of the mismatched base pair by protein complexes MutS $\alpha$  or MutS $\beta$
2. Degradation of the mismatched nucleotide and nearby nucleotides by the EXO1 enzyme
3. Synthesis of new base pairs by the polymerase Pol  $\delta$
4. Ligation of the two corrected DNA strands by DNA ligase

### 3.2.2 Microsatellite Instability

Microsatellites are tandem repeats of short DNA sequences. Typically, sequences range from 1 to 6 base-pairs and are repeated 5 to 50 times. An example of a microsatellite is shown in Figure 3-1.



*Figure 3-1 Example Microsatellite*

Figure shows an example microsatellite with a dimer (2 sequence repeat) repeated 5 times. Figure was created with BioRender.com.

Due to their repeated structures, microsatellites are prone to replication errors caused by DNA slippage, a phenomenon in which DNA polymerase dissociates from the template strand and re-anneals at the incorrect location ([Jiricny, 2006](#)). Normally, mutations occurring in microsatellites are corrected by the MMR system. When the MMR system is defective, the

mutations accumulate and result in microsatellite instability (MSI). MSI is a genomic condition characterized by genome-wide accumulation of insertion/deletion mutations in microsatellites.

In healthy genomes, mismatch repair (MMR) proteins are responsible for recognizing and correcting these DNA mutations. However, mutations in MMR genes can lead to deficient MMR (dMMR) systems, preventing the correction of insertion/deletion mutations and resulting in MSI. Tumors with high levels of MSI (MSI-H tumors) resulting from a dMMR system accumulate far more somatic mutations (e.g. non-inherited mutations) than tumors with pMMR systems. MSI-H tumors can harbor 10 to 20 times more somatic mutations than microsatellite stable tumors ([Lee, Murphy, Le, & Diaz Jr, 2016](#)).

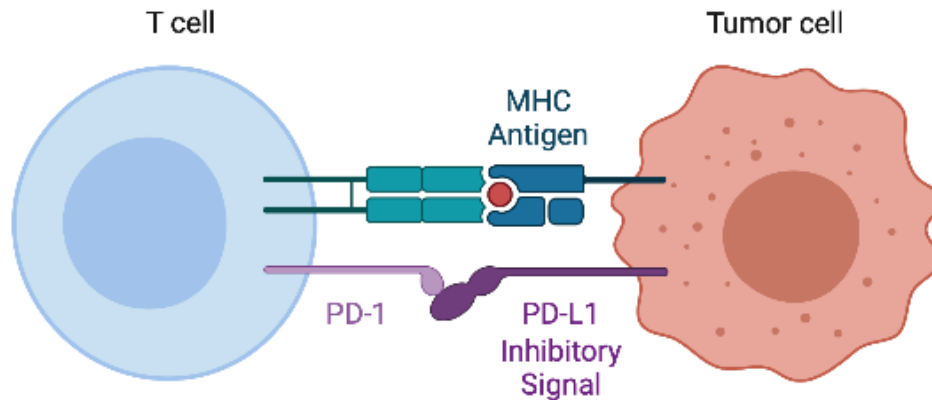
Somatic mutations can cause unintended translation of peptides. Neoantigens are peptides located on the surface of cancer cells that form due to the somatic mutations in the tumor DNA. The large somatic mutational burden of MSI-H tumors can result in a large number of tumor-specific neoantigens ([Marcus, Lemery, Keegan, & Pazdur, 2019](#)). Because neoantigens are only expressed by tumor cells, they can be targeted by cancer therapeutics without posing a threat to normal cells. Furthermore, because they present on the surface of tumor cells, they are easier targets for therapeutics than intra-cellular targets.

### 3.2.3 Immune Checkpoint Inhibitors

Immune checkpoints regulate the balance between activating immune cells and suppressor immune cells during an immune response. If too many activating immune cells are present, the immune system produces an overstimulated response that can damage healthy cells and tissues and cause autoimmune diseases. When too many suppressor immune cells are present, immune response is under stimulated allowing disease-causing agents to proliferate ([Pardoll, 2012](#)).

Antigen-presenting tumor cells signal to the immune system that a tumor cell is foreign, which would normally initiate an immune system response. However, tumor cells exploit immune checkpoint pathways to evade destruction by the immune system by upregulating inhibitory molecules that signal to the immune system to suppress response. For example, an antigen-

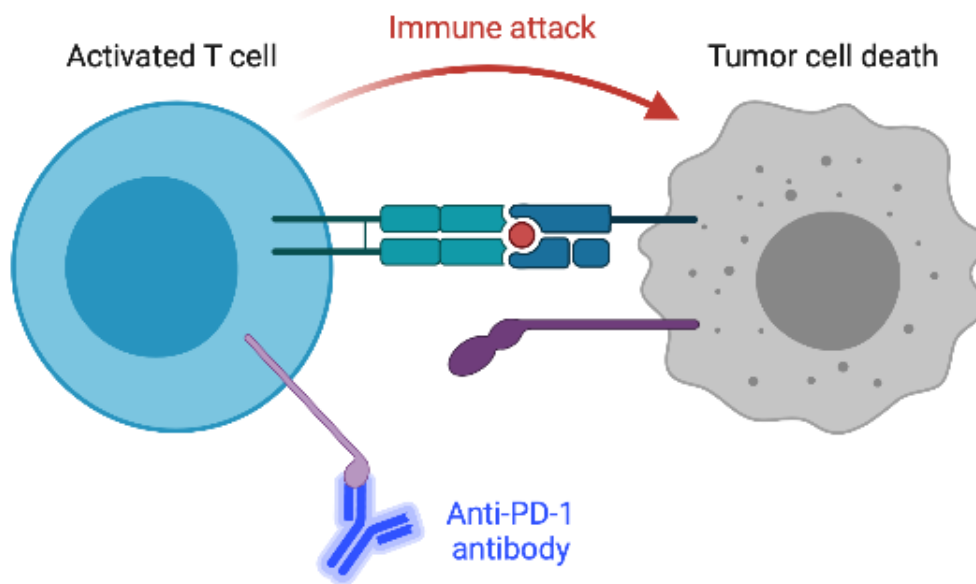
presenting tumor cell will uptake blockade molecules such as PD-L1 so that when the antigen attracts an immune cell, the blockade molecule will inhibit any response. There are many inhibitory and activating molecules that can affect the immune response. A simplified illustration of immune checkpoint inhibition of a t-cell by PD-L1 is shown in Figure 3-2.



*Figure 3-2 Immune Checkpoint Inhibition of T-Cell Activation*

MHC tumor antigen attracts T-cell, while PD-L1 molecule inhibits a T-cell response. Figure was created with BioRender.com.

Cancer therapeutics known as immune checkpoint inhibitors act on molecules within the immune checkpoint signaling pathway to disrupt the inhibitory signals of tumor cells. For example, they may stimulate anti-PD-1 antibodies, which inhibit the T-cell checkpoint, and render the tumor cells inhibitory molecule ineffective, allowing the immune response to activate. A simplified illustration of the activation of an immune response in the presence of anti-PD-1 antibodies is shown in Figure 3-3.



*Figure 3-3 Anti-PD-1 Antibody Aided T-Cell Activation*

Anti-PD-1 antibody blocks immune checkpoint, allowing for an immune system response. Figure was created with BioRender.com.

Cancer tumors with high levels of MSI (MSI-H) can express an increased number of neoantigen peptides that signal a cell is cancerous, increasing the likelihood of response to immunotherapy treatments ([Marcus et al., 2019](#)).

Clinical trials for the immune checkpoint inhibitor pembrolizumab have shown that MSI-H status correlates with clinical response, agnostic of primary cancer site, leading to the first pan-cancer drug approval by United States Food and Drug Administration's (FDA) for use of pembrolizumab in MSI-H or dMMR cancer patients ([Lemery et al., 2017](#)).

### 3.2.4 Fragment Analysis

While no FDA approved companion diagnostic for MSI assessment is currently indicated to direct the use of pembrolizumab or other immune checkpoint inhibitors, the most common historical method for MSI classification in tumor specimens has been a polymerase chain reaction (PCR)-based fragment analysis (FA) assay. FA classifies MSI status as high (MSI-H), low (MSI-L), or stable (MSS) by assessing DNA fragment length variation between tumor and patient-

matched normal tissue at five genomic loci (BAT25, BAT26, D2S123, D5S346, and D17S250), according to a 1997 National Cancer Institute (NCI) consensus meeting ([Boland et al., 1998](#)).

FA results utilized in the research presented in this chapter were obtained via a commercially available fluorescent multiplex PCR-based method, MSI Analysis System (Promega Life Sciences). The system utilizes comparative analysis between enriched tumor tissue sample and nontumor (normal) tissue. Prior to molecular testing, tumor and matched normal tissue were collected by harvesting targeted tissue using manual microdissection techniques. Allelic profiles were generated for BAT-25, BAT-26, MONO-27, NR-21, and NR-24 and compared between tumor and normal samples by a board-certified clinical molecular geneticist at Caris Life Sciences, a college of American Pathologists (CAP) / Clinical Laboratory Improvement Amendments (CLIA) approved laboratory in Phoenix, AZ.

### 3.2.5 Immunohistochemistry

A dMMR mechanism, which can lead to MSI, is frequently associated with loss of MMR protein expression. Immunohistochemical (IHC) evaluation for the expression levels of four MMR proteins, MLH1, MSH2, MSH6, and PMS2, is commonly used to determine MMR status. Lack of expression in any one of these proteins is indicative of a dMMR system, while, in most cases, detection of expression across all four proteins is indicative of a proficient MMR (pMMR) system.

Because dMMR status is strongly correlated with MSI-H status and a pMMR status is strongly correlated with MSS status, detection of MMR status by IHC is used as an alternative method of determining patient eligibility for immune checkpoint inhibitors ([Le et al., 2017](#); [Le et al., 2015](#); [McConechy et al., 2015](#)).

IHC analysis of mismatch repair proteins MSH6, MSH2, MLH1, and PMS2 utilized in this research was performed on full slides of formalin-fixed paraffin-embedded (FFPE) tumor specimens using automated staining techniques on the Benchmark XT (Ventana) utilizing antibody clones MLH1 (M1), MSH2 (G219-1129), MSH6 (44), and PMS2 (EPR3947). MMR status was determined to be deficient if staining indicated a complete loss of protein in any one of the four biomarkers in tumor cells and proficient if staining was present for all four proteins in



tumor. All analyses were performed at the Caris Life Sciences' CAP/CLIA certified laboratory in Phoenix, AZ.

### 3.2.6 Next Generation Sequencing

Recently, next-generation sequencing (NGS) assays that assess thousands of microsatellite loci have been developed for MSI detection in solid tumor tissues. These methods have been shown to improve robustness and sensitivity of MSI detection ([Vanderwalde et al., 2018](#)). Additionally, NGS-based methods do not require patient-matched normal tissue since NGS data can be aligned to a reference genome to detect mutations. This eliminates the need for collection and processing of normal tissue, which increases clinical access to MSI screening, as normal tissue is often not available from routine diagnostic biopsies. Additionally, NGS evaluation of MSI allows for screening to be performed as part of a panel of NGS tests performed on a single assay. This is ideal for targeted therapy because it allows for a broad analysis of potential biomarkers to be performed as part of a single assay, rather than having to perform additional biopsies for multiple individual tests.

Vanderwalde, et al ([Vanderwalde et al., 2018](#)) developed an NGS assay that aligns tumor genomic sequences to the human reference genome version hg19 ([Kent et al., 2002](#)) to assess variations between tumor and the reference genome at 7,317 microsatellite loci. Microsatellite loci were identified by scanning short tandem repeats of DNA sequences from NGS panel target regions found in the hg19 reference genome. All loci with genomic sequences equal to or longer than 5 repeats of monomers (N=6,960), 5 repeats of dimers (N=47), 4 repeats of trimers (N=228), 3 repeats of tetramers (N=57), or 3 repeats of pentamers (N=25) were included in the MSI calling model.

While aligning NGS results to 7,317 microsatellite loci on the reference genome allows for analysis of a large chunk of the microsatellite environment, the annotations of those loci are subject to population-bias due to the fact that the hg19 reference genome used for loci detection was derived from the initial human reference genome, which was created from a limited number

of individuals, with a representation heavily biased toward populations of European ancestry ([E pluribus unum, 2010](#); [Sherman et al., 2019](#)).

Findings from the 1000 Genomes Project indicate that variants can be specific to ancestral lines, and, in particular, individuals of African ancestry have more normal germline variation relative to other ancestral lines ([Fan et al., 2019](#); [Genomes Project Consortium, 2010](#)). This natural variation can occur throughout the genome, including in areas classified as microsatellites. While ancestral-specific germline variants are benign, they can be falsely classified as microsatellite insertion/deletion mutations if they are detected by an NGS assay in which the alignment process does not account for ancestral germline variation. Thus, alignment of NGS data to a primarily European-derived reference genome can lead to an increased false detection of microsatellite mutations and thus a higher false positive rate of MSI detection in patients with these variants, and in particular, within patients of African ancestry.

NGS data utilized in the research presented in this chapter were collected on genomic DNA isolated from FFPE samples using the NextSeq platform (Illumina, Inc). Prior to molecular testing, tumor enrichment was achieved by harvesting targeted tissue using manual microdissection techniques. A custom-designed SureSelect XT assay was used to capture 592 whole-gene targets (Agilent Technologies). All variants were detected with >99% confidence based on allele frequency and amplicon coverage, with an average sequencing depth of 750x and an analytic sensitivity of 5%. Sequencing alignment was compared with the reference genome hg19 from the UCSC Genome Browser database. All sequencing was performed at the Caris Life Sciences' CAP/CLIA certified laboratory in Phoenix, AZ.

### 3.3 Methods

#### 3.3.1 Study Cohorts

All analyses presented in this chapter were performed on de-identified, retrospective data. As such, this research was covered under international review board (IRB) Exemption, reviewed and determined by the Western Institutional Review Board (WIRB). Specimen analyzed in this study were collected from FFPE tissue samples from solid tumor biopsies performed

across multiple cancer types. All specimen submitted over a 4-year time period to Caris Life Science for genetic profiling as part of routine clinical care that had valid NGS results were considered for inclusion. A total of 6,262 specimen were enrolled in the study.

#### 3.3.1.1 Historical Cohort

All specimen that had undergone MSI testing by both FA and NGS (using the Vanderwalde method) were enrolled in a historical cohort comprising a random sampling representative of the entire population of specimen undergoing MSI analyses at Caris Life Sciences. A total of 6,198 retrospective samples were enrolled in this cohort, including 452 MSI-H, 65 MSI-L, and 5,681 MSS by FA.

#### 3.3.1.2 Flagged Cohort

Specimen having undergone MMR testing by IHC and MSI testing by NGS (using the Vanderwalde method) that had discordant results indicating an MSI-H or MSI-L status by NGS but pMMR by IHC were reviewed by a board certified clinical molecular geneticist and a board-certified pathologist for sequence variants associated with African populations in gnomAD ([Karczewski & Francioli, 2017](#)). Discordant samples with variants associated with African ancestry were flagged as having potential population-biased NGS MSI results. All flagged samples with sufficient remnant material for testing by FA were then enrolled in the flagged cohort and subsequently tested for MSI status by FA. Over a 6-month time period, a total of 64 samples were enrolled in the flagged cohort and tested by FA.

#### 3.3.1.3 Training and Validation Cohorts

The 6,262 available samples from the flagged and historical cohorts were divided into training and independent validation cohorts. The independent validation cohort included 122 samples: 90 randomly selected historical samples equally distributed across FA MSI results (30 FA MSI-H; 30 FA MSI-L; 30 FA MSS) and 32 randomly selected flagged samples. The remaining 6,140 samples were included in the training cohort. The 32 flagged samples included in training were used to assess how the model would perform on the flagged samples in the independent validation.

### 3.3.2 Ancestry and NGS Bias Assessment

Genetic signatures of samples were summarized utilizing population frequency data from gnomAD. To summarize the genetic signature of a sample, all alleles detected for that sample with reference single nucleotide polymorphism (SNP) IDs (RSIDs) in gnomAD were identified. Population frequencies of each allele were collected from gnomAD via Bioconductor ([Lek et al., 2016](#)) for the 7 populations represented in gnomAD (African/African-American, Latino/Admixed American, Ashkenazi Jewish, East Asian, Finnish, Non-Finnish European, and Other). An allele was then defined as supporting the population with the highest frequency of that allele. In the event of ties (e.g. an allele had equivalent frequencies in two or more populations), the allele was defined as supporting all populations with the highest frequencies. For each sample, the number of alleles detected that supported each of the seven populations were totaled and the percentages of alleles supporting each population were calculated. Allele frequencies of samples in the flagged cohort were used to confirm that the flagged cohort contained samples with genetic signatures consistent with a primarily African/African American ancestry. Allele frequencies of samples in the historical cohort were used to establish that the historical cohort was representative of all genetic ancestries.

After establishing ancestry by allele frequencies, potential bias of NGS based MSI results was assessed on the flagged cohort. Since all flagged cohort samples had discordant results when measured by IHC and NGS, FA results were used to assess bias. If no bias was present in the NGS call, it was hypothesized that FA results would be randomly distributed between MSI (high or low) and MSS. If MSI results by NGS were biased towards a higher rate of false positives in samples with African/African-American ancestry, it was hypothesized that FA results in the flagged cohort would agree more often with IHC MMR results than with NGS MSI results. A one-sided exact test was performed to test the hypothesis that, in a cohort with discordant NGS MSI and IHC MMR results, the probability of FA results that agreed with IHC would be greater than 50%.

### 3.3.3 Model Development

To mitigate effects of NGS bias on MSI calls, a new NGS calling model was built incorporating a feature selection step to select a subset of the original 7,317 MSI loci with the goal of eliminating any loci that could potentially be detected as false mutations in African/African-American genomes. An overview of the model development process is shown in Figure 3-4.

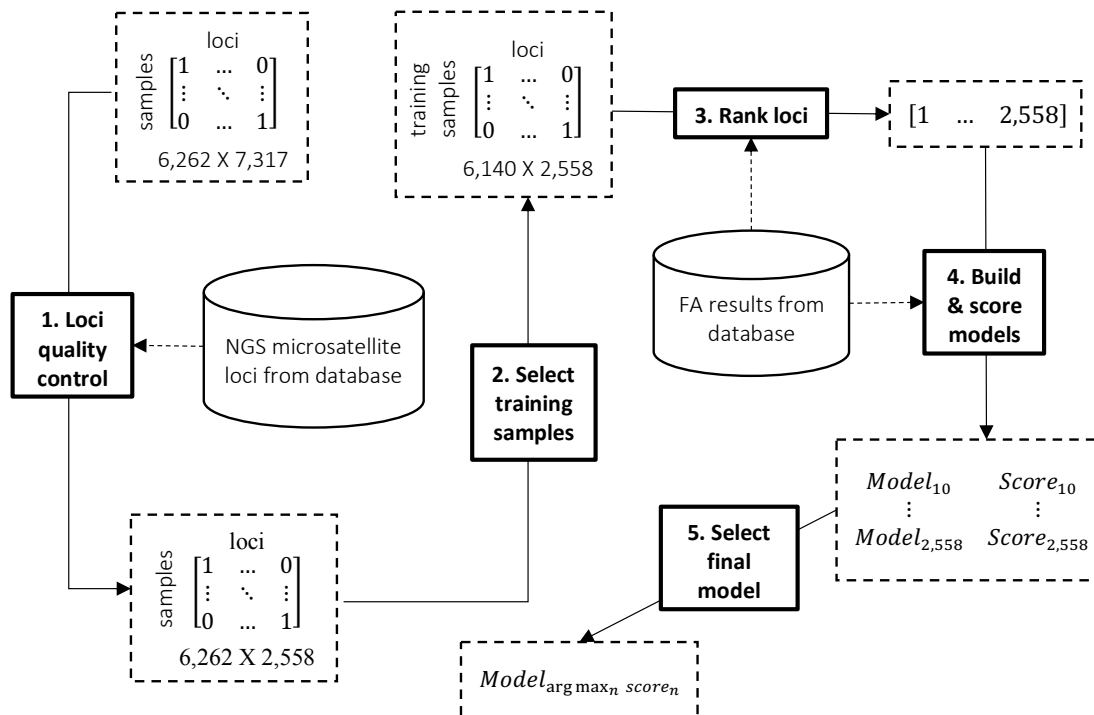


Figure 3-4 Diagram of NGS MSI Model Development Process

### 3.3.3.1 Loci Quality Control

The quality of the original 7,317 loci was assessed on all samples prior to model training. To be included in training, a locus had to meet the following minimum quality requirements:

1. Have a variant detected in at least one of the 6,262 study samples.
2. Show repeatability when measured on replicates of the same sample.
3. Have an average sequencing coverage depth of at least 200x.

A total of 4,759 loci were excluded during preprocessing. Of these, 4,588 failed criterion 1, 10 failed criterion 2, and 161 failed criterion 3. The remaining 2,558 loci were included in subsequent analyses.

### 3.3.3.2 Loci Ranking

The 2558 loci that met minimum quality requirements were then ranked based on their ability to predict FA MSI status in the training samples. An informative locus was expected to have a higher number of variants detected in the FA MSI-H samples than in the FA MSS samples. Analysis of variance (ANOVA) P-values were computed to measure how informative a locus was by comparing the difference in the mean number of variants detected between the FA MSI-H and FA MSS groups. If the mean of the FA MSS samples was higher than the mean of the FA MSI-H samples, the P-value for that locus was set to 1. Loci were ranked from lowest to highest P-value.

### 3.3.3.3 Model Building

The NGS MSI calling model classifies MSI status of a sample by totaling the number of variants detected across all microsatellite loci and comparing the total to predetermined thresholds. Totals greater than the upper threshold are classified as MSI-H; totals less than the lower threshold are classified as MSS; and totals within the inclusive range of the two thresholds are classified as MSI equivocal. A total of 2549 calling models were built using between 10 and 2558 loci included sequentially by ranking.

Thresholds for each model were calibrated on training data to optimize the separation between FA MSS and FA MSI-H results. Samples with FA MSI-L results were excluded from thresholding due to the fact that FA is subject to interpretation and different interpreters may classify FA MSI-L calls as MSI-H or MSS.

The first threshold was chosen to optimize the sum of sensitivity and specificity, and was calculated using methods provided in the 'pROC' package ([Robin et al., 2011](#)) for R version 3.5.2 ([R Core Team, 2019](#)). The second threshold was chosen to ensure a maximum 3% false negative (FN) rate in the training population. It was calculated as the minimum of:

1. The maximum number of variants detected in FA MSS samples and
2. The third percentile of variants detected in the FA MSI-H samples.

If the first threshold already met the 3% criterion, the second threshold had the effect of limiting the false positive (FP) calls.

#### 3.3.3.4 Model Scoring

To score NGS models, NGS results were benchmarked against FA results. Results indicating eligibility for treatment by immune checkpoint inhibitors (FA MSI-H and NGS MSIH) were considered positive, while results indicating ineligibility for immune checkpoint inhibitor treatment (FA MSI-L, FA MSS, and NGS MSS) were considered negative. Model scores were calculated as true positive (TP) + true negative (TN) – false positive (FP) – false negative (FN). NGS equivocal calls did not contribute to model scores because operations protocol in a clinical setting for an NGS equivocal result often dictated that the sample be retested by FA to definitively determine MSI status. The number of loci corresponding to the maximum model score was selected as the optimal number of loci to include in the final model.

#### 3.3.4 Model Validation

Performance of the modified model was assessed initially via cross-validation, to ensure the model was not overfitting the training data, and subsequently on the independent validation cohort, to confirm population bias was eliminated and overall performance was not hindered.

##### 3.3.4.1 Performance Measures

Cross-validation and independent validation performances of the modified NGS-based calling model were assessed by calculating sensitivity, specificity, positive predictive value (PPV), and negative predictive value (NPV) benchmarked against FA MSI results. For these calculations, results that indicated eligibility for treatment by immune checkpoint inhibitors (FA MSI-H and NGS MSI-H) were considered positive, while results that were not eligible for treatment by immune checkpoint inhibitors (FA MSI-L, FA MSS, and NGS MSS) were considered negative.

Again, NGS MSI-equivocal results were excluded from analysis as the clinical protocol for NGS MSI-equivocal results often dictates that samples be tested with FA. Percent equivocal values were reported for reference along with performance measures. Due to the fact that the

historical validation cohort was enriched for FA MSI-L and FA MSS samples, performance measures on this cohort were evaluated on the enriched population as well as a prevalence adjusted population. Prevalence values for MSI-H, MSI-L, and MSS used for adjustment were estimated from the FA MSI results in the training cohort.

#### 3.3.4.2 Cross Validation

Ten iterations of threefold cross-validation were performed on the training data to estimate model utility on unknown cases. Data were randomized to folds preserving distributions of FA MSI calls using the 'Caret' package ([Kuhn, 2015](#)) in R version 3.5.2. Training folds were used to rank loci and build and score models. The optimal model was then applied to the testing fold and performance measures were computed. Threefold CV was repeated for 10 different randomization seeds, and performance measures were averaged over these 10 runs.

#### 3.3.4.3 Independent Validation

Independent validation was assessed on a cohort of 30 MSI-H, 30 MSI-L, and 30 MSS tumor specimen, evaluated by FA. Performance measures were calculated on the independent validation cohort containing equal prevalence of each FA result.

Additionally, performance measures were calculated on the independent validation cohort adjusted for prevalence of FA results. Historical data suggest the true prevalence of FA results in the intended use population are 7.29% (452/6198) MSI-H, 1.05% (65/6198) MSI-L, and 91.66% (5681/6198) MSS. To adjust for this, results in the confusion matrix corresponding to MSI-H, MSI-L, and MSS were multiplied by 7.29%, 1.05%, and 91.66% respectively before calculation of performance measures were performed.

#### 3.3.5 Reference genome version comparison

The original NGS MSI calling model used human reference genome hg19 for alignment, and is the reference genome version utilized for this study. However, a newer version of the human reference genome, hg38, has since been released. Effects of aligning to this newer version were assessed on the independent validation cohort. MSI loci variants were recomputed



for all 7,317 microsatellite loci after alignment to hg38 and compared to the number of variants detected under hg19 alignment.

## 3.4 Results

### 3.4.1 Demographics

Gender and cancer types were compared across cohorts stratified by FA MSI status using chi-square test. No significant differences were found. Age was compared across cohorts stratified by FA MSI status using analysis of variance (ANOVA). The flagged cohorts had slightly lower average ages than the historical cohorts, which supports observations of lower ages of cancer in African-American populations.<sup>18</sup> Available demographics by FA MSI status for historical training, historical validation, flagged training, and flagged validation cohorts are shown in Table 3.1.

Table 3.1 Cohort Demographics by FA MSI Status

	Historical Training	Flagged Training	Historical Validation	Flagged Validation	P
<b>FA MSI-H</b>	<b>N = 422</b>	<b>N = 0</b>	<b>N = 30</b>	<b>N = 0</b>	
Age					0.61
Mean	65	---	65	---	
Median	66	---	70	---	
Range	22-90	---	25-90	---	
Gender					0.93
Female	334 (79.1%)	---	23 (76.7%)	---	
Male	88 (20.9%)	---	7 (23.3%)	---	
Cancer Type					0.88
Breast	2 (0.5%)	---	0 (0%)	---	
Gastrointestinal	185 (43.8%)	---	12 (40%)	---	
Genitourinary	3 (0.7%)	---	0 (0%)	---	
Gynecologic	220 (52.1%)	---	18 (60%)	---	
Male Genital Tract	1 (0.2%)	---	0 (0%)	---	
Other	7 (1.7%)	---	0 (0%)	---	
Thoracic	4 (0.9%)	---	0 (0%)	---	
<b>FA MSI-L</b>	<b>N = 35</b>	<b>N = 0</b>	<b>N = 30</b>	<b>N = 0</b>	
Age					0.94
Mean	61	---	64	---	
Median	65	---	64	---	
Range	31-79	---	44-82	---	
Gender					0.73
Female	22 (62.9%)	---	21 (70%)	---	
Male	13 (37.1%)	---	9 (30%)	---	
Cancer Type					0.4
Gastrointestinal	18 (51.4%)	---	11 (36.7%)	---	
Gynecologic	14 (40%)	---	17 (56.7%)	---	
Other	2 (5.7%)	---	1 (3.3%)	---	
Sarcoma	1 (2.9%)	---	0 (0%)	---	
Skin	0 (0%)	---	1 (3.3%)	---	
<b>FA MSS</b>	<b>N = 5,651</b>	<b>N = 32</b>	<b>N = 30</b>	<b>N = 32</b>	
Age					0.02
Mean	66	62	65	60	
Median	67	65	66	60	
Range	7-90	38-83	42-93	28-82	
Gender					0.69
Female	3,539 (62.6%)	23 (71.9%)	20 (66.7%)	21 (65.6%)	
Male	2,112 (37.4%)	9 (28.1%)	10 (33.3%)	11 (34.4%)	
Cancer Type					0.27
Breast	175 (3.1%)	0 (0%)	0 (0%)	2 (6.2%)	
Gastrointestinal	2621 (46.4%)	15 (46.9%)	14 (46.7%)	9 (28.1%)	
Genitourinary	173 (3.1%)	2 (6.2%)	1 (3.3%)	3 (9.4%)	
Gynecologic	1640 (29%)	8 (25%)	8 (26.7%)	10 (31.2%)	
Lymphoma	4 (0.1%)	0 (0%)	0 (0%)	0 (0%)	
Male Genital Tract	1 (0%)	0 (0%)	0 (0%)	0 (0%)	
Neuroendocrine	77 (1.4%)	2 (6.2%)	0 (0%)	1 (3.1%)	
Other	169 (3%)	2 (6.2%)	3 (10%)	2 (6.2%)	
Brain	20 (0.4%)	0 (0%)	0 (0%)	0 (0%)	
Sarcoma	62 (1.1%)	0 (0%)	0 (0%)	1 (3.1%)	
Skin	125 (2.2%)	0 (0%)	1 (3.3%)	0 (0%)	
Thoracic	584 (10.3%)	3 (9.4%)	3 (10%)	4 (12.5%)	

Ancestry was summarized as described in the methods section. On average, 43% of the alleles for flagged cohort samples supported African/African-American population, with a minimum of 38%, while the historical cohort averaged 20%, suggesting the flagged cohort samples had genetic signatures more consistent with African/African-American ancestry than the average sample from the historical cohort. A comparison of the distribution of alleles by population between the historical cohort and the flagged cohort can be shown in Figure 3-5.

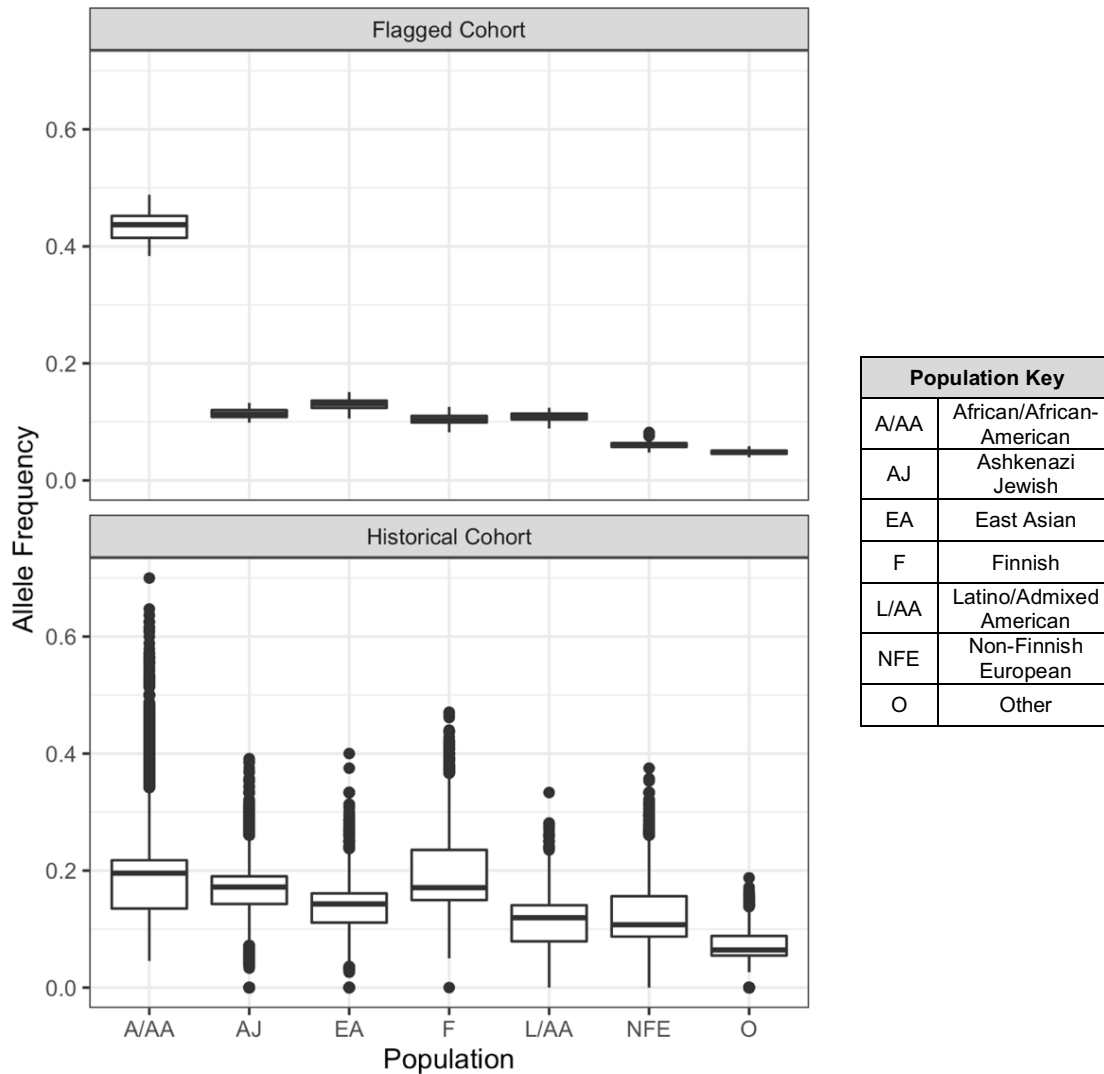


Figure 3-5 Sample Population Allele Frequencies by Cohort

Samples in the flagged cohort had significantly more African/African-American allele frequencies than any other population. By comparison, allele frequencies for samples in the historical cohort were more evenly distributed.

### 3.4.2 Ancestry and NGS Bias Results

All 64 samples in the flagged cohort had FA MSS results. The observed probability that the FA MSI results were 100% (95% CI: 95%, 100%) and was statistically significantly greater than the probability expected by chance ( $P \ll .001$ ), suggesting a strong bias in the NGS results. The performance measures of the NGS results on the flagged cohort are presented in Table 3.2.

Table 3.2 Original NGS Model Performance on Flagged Cohort Samples.

	FA Result	Original NGS Model Results (N)			Specificity % (95% CI*)	Equivocal % (95% CI*)
		MSI-H	Equivocal	MSS		
Flagged Cohort	MSS	16	48	0	0 (0.0, 20.6)	75 (62.6, 85.0)

\*Confidence intervals (CI) calculated by Clopper-Pearson method.

### 3.4.3 Validation Results

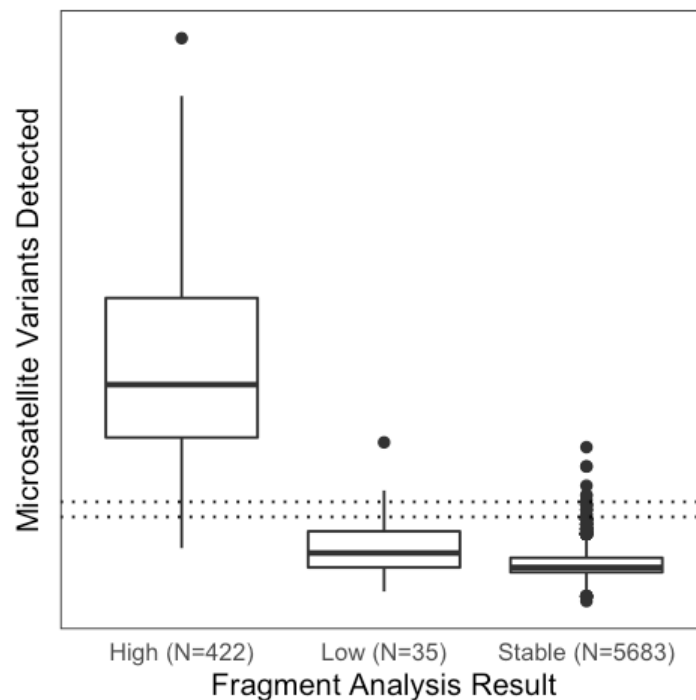
Cross-validation and independent validation results are summarized in Table 3.3.

Table 3.3 Performance Metrics Across NGS Model Methods and Cohorts

NGS MSI Method	Cohort	FA Result	NGS Results (N)			Sensitivity % (95% CI*)	Specificity % (95% CI*)	PPV % (95% CI*)	NPV % (95% CI*)	Equivocal % (95% CI*)
			MSI-H	Equiv.	MSS					
Original Method	Historical Training	MSI-H	399	7	16	96.1	99.5	93.2	99.7	1.0
		MSI-L	3	5	27	(93.8, 97.8)	(99.3, 99.7)	(90.4, 95.4)	(99.5, 99.8)	(0.8, 1.3)
MSS		26	52	5573						
	Flagged Training	MSS	9	23	0	---	0.0 (0, 33.6)	---	---	71.9 (53.3, 86.3)
Modified Method Averaged over 10 X 3-fold CV		Historical Training	MSI-H	405	7.9	9.1	97.8	99.7	96.4	99.8
	MSI-L		3.7	2.1	29.2	(95.9, 99.0)	(99.6, 99.9)	(94.2, 98.0)	(99.7, 99.9)	(0.2, 0.5)
	MSS		11.3	8.8	5630.9					
	Flagged Training	MSS	0.2	1.1	30.7	---	99.4 (88.8, 100)	---	---	3.4 (0.1, 16.2)
Original Method	Historical Validation	MSI-H	28	1	1	96.6	86.0	77.8	98.0	4.4
		MSI-L	8	3	19	(82.2, 99.9)	(74.2, 93.7)	(60.8, 89.9)	(89.4, 99.9)	(1.2, 11)
		MSS	0	0	30					
	Historical Validation (Adjusted)	MSI-H	5.77	0.21	0.21	96.5	99.8	97.6	99.7	0.3
		MSI-L	0.14	0.05	0.32	(54.1, 100)	(95.7, 100)	(54.1, 100)	(95.7, 100)	(0.0, 4.0)
		MSS	0	0	83.3					
Flagged Validation	MSS	7	25	0	---	0.0 (0.0, 41)	---	---	78.1 (60.0, 90.7)	
Final Modified Method	Historical Validation	MSI-H	29	1	0	100	84.7	76.3	100	2.2
		MSI-L	9	1	20	(88.1, 100)	(73.0, 92.8)	(59.8, 88.6)	(92.9, 100)	(0.3, 7.8)
		MSS	0	0	30					
	Historical Validation (Adjusted)	MSI-H	5.98	0.21	0	100	99.8	97.6	100	0.3
		MSI-L	0.15	0.02	0.34	(54.1, 100)	(95.7, 100)	(54.1, 100)	(95.7, 100)	(0, 4.0)
		MSS	0	0	83.3					
Flagged Validation	MSS	0	0	32	---	100 (89.1, 100)	---	---	0.0 (0.0, 10.9)	

\*Confidence intervals (CI) calculated by Clopper-Pearson method. All decimal values were rounded to the nearest whole number for CI calculations.

Most notably, results showed a drastic improvement of specificity in the flagged cohorts. Specificity on the flagged cohort went from 0% with the original model to 94% in the cross-validated models within the training set, and from 0% with the original model to 100% in the final modified model during validation. Additionally, PPV had a moderate increase from 93.2% in the original model to 96.4% in the cross-validated models. Because cross-validation results were satisfactory, the methodology was used on the full set of training data to create a final modified NGS MSI calling model. The 2558 loci that passed quality control were included in the analysis. The optimal model score corresponded to the model containing 2011 loci, eliminating a total of 547 uninformative microsatellite loci. Additional figures illustrating model selection are provided in the Figure 3-6 and Figure 3-7.



*Figure 3-6 Depiction of Final Model Selected on Training Data*

Thresholds shown as horizontal dotted lines.

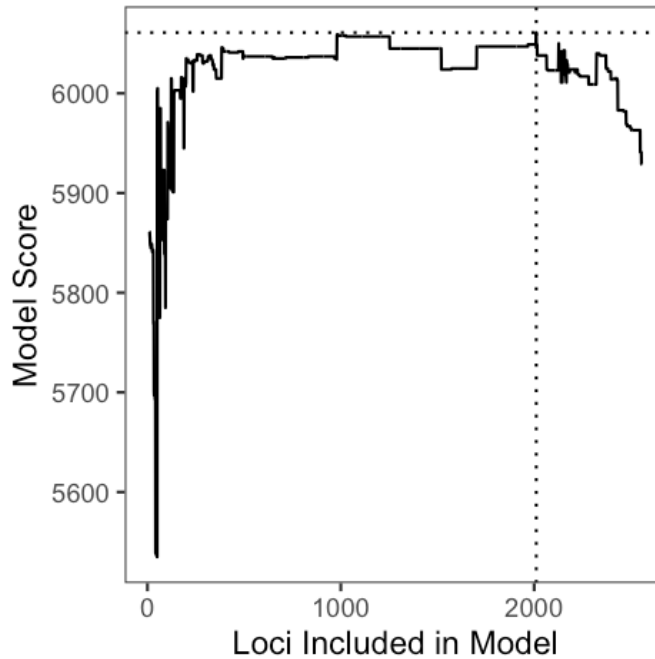


Figure 3-7 Number of Loci Included in Model Versus Model Score

The dotted line indicates the maximum score at 2011 Loci. Model score depicts the number of training samples correctly classified out of 6,140 samples.

Final model results on the Flagged cohort are shown in Table 3.4.

Table 3.4 Updated NGS Model Performance on Flagged Cohort Samples.

	FA Result	NGS Model Results (N)			Specificity % (95% CI*)	Equivocal % (95% CI*)
		MSI-H	Equivocal	MSS		
Flagged Cohort	MSS	0	0	64	100 (94.4, 100)	0 (0, 5.6)

\*Confidence intervals (CI) calculated by Clopper-Pearson method.

Results of the final model compared to the original model are shown by cancer type in Table 3.5.

Table 3.5 Performance of NGS Model Methods by Cancer Type

Cancer Type	NGS MSI Method	FA Result	NGS Results (N)			Sensitivity % (95% CI*)	Specificity % (95% CI*)	PPV % (95% CI*)	NPV % (95% CI*)	Equivocal % (95% CI*)
			MSI-H	Equiv.	MSS					
Breast	Original	MSI-H	2	0	0	100	98.8	50 (6.8, 93.2)	100 (97.9, 100)	2.8 (0.9, 6.4)
		MSI-L	0	0	0	(15.8, 100)	(95.9, 99.9)			
		MSS	2	5	170					
	Modified	MSI-H	2	0	0	100	99.4	66.7	100	0.6 (0, 3.1)
		MSI-L	0	0	0	(15.8, 100)	(96.9, 100)	(9.4, 99.2)	(97.9, 100)	
		MSS	1	1	175					
Gastro-intestinal	Original	MSI-H	191	0	6	97	99.4	91.8	99.8	1.5
		MSI-L	0	3	26	(93.5, 98.9)	(99, 99.6)	(87.2, 95.2)	(99.5, 99.9)	(1.1, 2)
		MSS	17	39	2603					
	Modified	MSI-H	192	2	3	98.5	100	100	99.9	0.2
		MSI-L	0	2	27	(95.6, 99.7)	(99.9, 100)	(98.1, 100)	(99.7, 100)	(0.1, 0.4)
		MSS	0	1	2658					
Genito-urinary	Original	MSI-H	3	0	0	100	98.3	50	100	2.7
		MSI-L	0	0	0	(29.2, 100)	(95, 99.6)	(11.8, 88.2)	(97.9, 100)	(0.9, 6.3)
		MSS	3	5	171					
	Modified	MSI-H	3	0	0	100	100	100	100	0 (0, 2)
		MSI-L	0	0	0	(29.2, 100)	(98, 100)	(29.2, 100)	(98, 100)	
		MSS	0	0	179					
Gynecologic	Original	MSI-H	221	8	9	96.1	99.1	93.6	99.4	1.9
		MSI-L	11	5	15	(92.7, 98.2)	(98.5, 99.5)	(89.7, 96.4)	(99, 99.7)	(1.3, 2.6)
		MSS	15	24	1627					
	Modified	MSI-H	230	6	2	99.1	99.7	97.9	99.9	0.7
		MSI-L	12	2	17	(96.9, 99.9)	(99.3, 99.9)	(95.1, 99.3)	(99.6, 100)	(0.4, 1.2)
		MSS	5	6	1655					
Lymphoma	Original	MSI-H	0	0	0	---	100	---	100	0 (0, 60.2)
		MSI-L	0	0	0		(39.8, 100)		(39.8, 100)	
		MSS	0	0	4					
	Modified	MSI-H	0	0	0	---	100	---	100	0 (0, 60.2)
		MSI-L	0	0	0		(39.8, 100)		(39.8, 100)	
		MSS	0	0	4					
Male Genital Tract Malignancy	Original	MSI-H	0	0	1	0 (0, 97.5)	100	---	50 (1.3, 98.7)	0 (0, 84.2)
		MSI-L	0	0	0		(2.5, 100)			
		MSS	0	0	1					
	Modified	MSI-H	0	0	1	0 (0, 97.5)	100	---	50 (1.3, 98.7)	0 (0, 84.2)
		MSI-L	0	0	0		(2.5, 100)			
		MSS	0	0	1					
Neuro-endocrine tumors	Original	MSI-H	0	0	0	---	98.7	0 (0, 97.5)	100 (95.2, 100)	5 (1.4, 12.3)
		MSI-L	0	0	0		(92.9, 100)			
		MSS	1	4	75					
	Modified	MSI-H	0	0	0	---	100	---	100 (95.4, 100)	1.2 (0, 6.8)
		MSI-L	0	0	0		(95.4, 100)			
		MSS	0	1	79					



None of These Apply	Original	MSI-H	6	0	1	85.7	100	100	99.4	3.2
		MSI-L	0	0	3	(42.1,	(97.9,	(54.1,	(96.8,	(1.2,
		MSS	0	6	170	99.6)	100)	100)	100)	6.9)
	Modified	MSI-H	7	0	0	100	100	100	100	0 (0,
		MSI-L	0	0	3	(59,	(97.9,	(59,	(97.9,	2)
		MSS	0	0	176	100)	100)	100)	100)	
Primary Brain Tumors	Original	MSI-H	0	0	0	---	100	---	100	10
		MSI-L	0	0	0	---	(81.5,	---	(81.5,	(1.2,
		MSS	0	2	18	---	100)	---	100)	31.7)
	Modified	MSI-H	0	0	0	---	95	0 (0,	100	0 (0,
		MSI-L	0	0	0	---	(75.1,	97.5)	(82.4,	16.8)
		MSS	1	0	19	---	99.9)	---	100)	
Sarcoma	Original	MSI-H	0	0	0	---	100	---	100	3.1
		MSI-L	0	0	1	---	(94.1,	---	(94.1,	(0.4,
		MSS	0	2	61	---	100)	---	100)	10.8)
	Modified	MSI-H	0	0	0	---	100	---	100	0 (0,
		MSI-L	0	0	1	---	(94.3,	---	(94.3,	5.6)
		MSS	0	0	63	---	100)	---	100)	
Skin Cancer	Original	MSI-H	0	0	0	---	100	---	100	0.8
		MSI-L	0	0	1	---	(97.1,	---	(97.1,	(0,
		MSS	0	1	125	---	100)	---	100)	4.3)
	Modified	MSI-H	0	0	0	---	100	---	100	0.8
		MSI-L	0	0	1	---	(97.1,	---	(97.1,	(0,
		MSS	0	1	125	---	100)	---	100)	4.3)
Thoracic	Original	MSI-H	4	0	0	100	99.3	50	100	2 (1,
		MSI-L	0	0	0	(39.8,	(98.2,	(15.7,	(99.4,	3.5)
		MSS	4	12	578	100)	99.8)	84.3)	100)	
	Modified	MSI-H	4	0	0	100	100	100	100	0.2
		MSI-L	0	0	0	(39.8,	(99.4,	(39.8,	(99.4,	(0,
		MSS	0	1	593	100)	100)	100)	100)	0.9)

\*Confidence intervals (CI) calculated by Clopper-Pearson method. All decimal values were rounded to the nearest whole number for CI calculations.

#### 3.4.4 Reference Genome Version Comparison Results

Overall 892,674 variants were assessed (7,317 loci across 122 samples). Of these, only two variants were discordant between the hg19 and hg38 alignments. One microsatellite variant was detected under hg19 alignment, but not detected under hg38 alignment due to different alignments on the reverse strand in hg38. The other microsatellite variant was detected under hg38 alignment but not under hg19 alignment, due to low variant allele frequency. The discrepant variant calls were detected in two samples. However, discrepancies did not affect the final MSI status determination. Both samples were classified as MSI-H under both the hg19 and hg38 alignments. Therefore, there was 100% concordance of NGS MSI calls between the genome

versions. This result suggests that all conclusions drawn in the analyses of hg19-aligned samples are valid in samples aligned to hg38.

### 3.5 Discussion

In this study, a bias was observed in NGS MSI test results within a cohort of patients with presumed African ancestry. Our goal was to refine the selection of microsatellite loci used to determine MSI status and improve the specificity of the test for patients prone to the observed bias. We accomplished this by training a new model using a large database of previously exhibited a false positive NGS-based MSI test result compared to FA or IHC.

Initial refinement required a quality assessment of the 7317 original loci, which eliminated over half the loci. Next, a model-building process eliminated noninformative loci resulting in a final set of 2011 loci that enhanced overall performance of the calling model. In particular, the updated model increased specificity in flagged cases by 100% in an independent validation, and increased overall PPV by approximately 3% in a cross-validation assessment. Overall specificity was not affected due to the large number of MSS samples in the general population. However, the 3% increase in PPV seen in cross-validation results suggests roughly 3% of MSI-H samples in the general population were prone to the bias observed in this study.

While the research here used optimal model scores to select the final model built on 2011 loci, it is worth noting that Figure 3-7 shows that several other loci counts may also be reasonable selections. In some instances, it is ideal to maximize the number of loci selected. For instance, if the assay baits will change in future versions of the NGS assay, some loci may drop out due to low quality. In particular, if baits are eliminated in future assays, the depth of loci captured by those baits may not be large enough to accurately determine mutational status, and therefore the loci will have to drop out of the calling model. Therefore, having more loci makes the model more robust to lost loci due to low depth if baits are eliminated. On the other hand, having fewer loci in the calling model can streamline the analysis, and reduce noise due to additional loci that may not be as reliable. For example, Figure 3-7 shows that model performance actually dropped before the optimal performance seen at 2011 loci. Therefore, selecting fewer loci, perhaps by

selecting the number of loci that maximized performance, without seeing any loci that significantly reduced performance, may be more robust to noise in the model.

Performance of MSI diagnostics are important to direct the use of immunotherapies due to the immunological response induced in patients whose tumors produce large numbers of neoantigens. This immunological landscape is not present in MSS patients, so false positive results may lead to patients receiving a pharmacological agent unlikely to have clinical benefit.

While the performance of FA MSI diagnostics have been well established, NGS-based methods for MSI assessment offer several advantages: they can extend the availability of MSI diagnostics to patients whose tumor biopsies do not have sufficient normal tissue for MSI assessment by FA; they can potentially provide a more accurate assessment of genomic signatures due to the large number of microsatellite loci surveyed; and they can be assessed as part of comprehensive genetic profiling panels, resulting in conservation of tissue and reduced time-to-results, allowing clinicians to select appropriate therapies more quickly.

The work presented here suggests that NGS-based tests that are affected by population biases can significantly benefit from training models using data from unbiased methods. Although this work focused on MSI, similar approaches could likely be utilized to minimize biases in NGS-based tests for other biomarkers as well.

## CHAPTER 4

### BIMIXT: MAXIMUM LIKELIHOOD ESTIMATION OF RECEIVER OPERATING CHARACTERISTIC (ROC) CURVES FROM MULTIMODAL, NON-GAUSSIAN DATA<sup>3</sup>

#### 4.1 Introduction

Disease heterogeneity refers to the existence of varying genetic signatures observed across patients with a single disease. While a disease may present similarly across patients, the variation in underlying genomic mutational patterns can give rise to differing responses to therapies and differing medical prognoses.

Over the past several decades, molecular sequencing has revealed genetic subtypes of many heterogeneous diseases. A canonical example of disease heterogeneity is breast cancer which is commonly classified based on the presence or absence of three biomarkers: estrogen receptor (ER), progesterone receptor (PR), and the HER2 gene. ER and PR are hormone receptors (HR). Hormone receptor status has been linked to patient prognoses with ER+/PR+ patients showing significantly decreased mortality rates compared to patients with only one HR mutation, or no HR mutations ([Dunnwald et al., 2007](#)). Furthermore, meta-analyses of data from 20 clinical trials has shown that response to the hormone therapy drug tamoxifen can significantly decrease mortality in patients with ER+ tumors, but has little to no effect in patients that are ER- ([Group, 2011](#)). Finally, targeted therapies that are engineered to target the HER2 gene have shown significant decreases in mortality of HER2+ breast cancer patients ([Slamon et al., 2001](#)).

While many disease subtypes have been classified, subtyping remains an active field of research, particularly within oncology. Detection of biomarkers associated with cancer status may indicate which genes play crucial roles in the molecular pathways associated with specific cancer subtypes and may be good candidates for targeted therapies.

---

<sup>3</sup> The algorithm developed in this chapter is freely available from ([Winerip, Wallstrom, & LaBaer, 2015](#)): Michelle Winerip, Garrick Wallstrom and Joshua LaBaer (2015). Bimixt: Estimates Mixture Models for Case-Control Data. R package version 1.0. <https://CRAN.R-project.org/package=bimixt>

Early phase biomarker discovery studies that aim at detecting potentially actionable cancer mutations often revolve around assaying as many markers as possible and trying to identify relevant markers from the larger population that can distinguish between subjects from two different categories. For example, a biomarker discovery study of a proteomic screening may look at the expression levels of thousands of proteins to try to find the proteins with the greatest ability to distinguish individuals with a disease (cases) from individuals without the disease (controls). A single protein's ability to distinguish cases from controls is determined by calculating sensitivity (true positive rate) and specificity (true negative rate) at various thresholds for the expression level. However, due to disease heterogeneity, finding such markers may be difficult as they may only be mutated in a small sub-sample of any given cohort. Thus, heterogeneity should be accounted for when assessing individual biomarker candidates.

This chapter presents a novel method called 'bimixt' for fitting a mixture model to numeric measurements from heterogeneous populations. It covers estimation of the model, application of the model to real biomarker data, and utility of the model for assessing individual biomarker candidates, specifically in moderately small case/control studies (e.g. 30 – 100 subjects). The methods presented are made freely available via the open-source 'bimixt' R package ([Winerip et al., 2015](#)).

## 4.2 Background

In general, a good biomarker will have a threshold that yields high sensitivity and high specificity. Biomarkers with good distinguishing abilities can be incorporated into clinical decision support as part of clinical trial endpoints ([Strimbu & Tavel, 2010](#)), diagnostic test panels, or as less costly or less invasive preliminary tests to help guide physicians to further testing prior to diagnosis. However, for some diseases, the presence of latent heterogeneity presents significant challenges for assessing biomarker quality.

Latent disease heterogeneity refers to diseases that are made up of unknown, biologically differing subtypes. Patients with different subtypes may exhibit different prognoses and different responses to clinical treatments. Molecular studies are continually revealing new

subtypes of diseases, and it is possible that even known disease subtypes are themselves made of several more unknown subtypes ([Wallstrom, Anderson, & LaBaer, 2013](#)).

One challenge of latent disease heterogeneity is that a biomarker that classifies individuals of one subtype well may not look like a good classifier for the entire population. For instance, if a biomarker correctly classifies 100% of subjects with one subtype, but that subtype is only present in 20% of the cases, the sensitivity of that biomarker will be roughly 20%. That is, the sensitivity of that biomarker will be limited by the prevalence of that subtype in the population ([Anderson & LaBaer, 2005](#)).

Compounding this challenge is the fact that many biomarker discovery studies often have small sample sizes, making it difficult to estimate the population sensitivity of a biomarker from the sample. Because the sensitivity of a good biomarker may not be much higher than the sensitivity of a poor biomarker, and because biomarker studies are often small, it is important to have a method that can accurately estimate sensitivity in small sample studies of biomarkers for heterogeneous diseases.

We can get a more accurate value of sensitivity from a small sample taken from a heterogeneous population if we have a suitable technique for generating a smooth receiver operator characteristic (ROC) curve. A ROC curve is a plot of sensitivity or true positive rate (TPR) versus 100% - specificity or false positive rate (FPR) for each possible threshold of the biomarker. For a review of ROC curves see Greene & Swets, 1966 ([Green & Swets, 1966](#)).

We can construct an empirical ROC curve from sensitivity and specificity values of the observed data. However, particularly when sample sizes are small, slight perturbations in the specificity of the empirical curve can result in large changes in sensitivity or vice versa. Smooth ROC curves have the advantage of filling in the gaps of empirical ROC curves to allow for continuous estimation of sensitivity and specificity. Thus, if the smooth curve is generated using appropriate assumptions for the data, the smooth curve can give more accurate estimates for sensitivity and specificity.

ROC analysis has been used to aid medical decision making since the 1950s when it was used to assess the performance of an automated Pap smear reader in distinguishing cancerous from non-cancerous cytology smears ([Zweig & Campbell, 1993](#)).

Many popular smoothing techniques exist for ROC analysis. Those implemented in the R package pROC ([Robin et al., 2011](#)) include a parametric binormal method and a non-parametric kernel density method.

The parametric method implemented in pROC is a modified binormal method in which a linear association is assumed between the normal quantile function values of sensitivity and specificity. This assumes that both cases and controls are normally distributed after a monotone transformation. In pROC the parameters of the linear function are estimated using regression whereas in the traditional binormal method proposed by Metz, they are estimated using maximum likelihood estimates ([Metz, Herman, & Shen, 1998](#)).

The non-parametric method implemented in pROC is derived from kernel density estimates of TPR and FPR ([Zou, Hall, & Shapiro, 1997](#)). Based on histograms of case and control data. However, when sample sizes are small, histograms may not be representative of the true population. Furthermore, histograms present the danger of overfitting the sample data.

More recently, Gaussian mixture models have been used to model smooth ROC curves. In one approach, the expectation-maximization (EM) algorithm ([Dempster, Laird, & Rubin, 1977](#)) is used to find Gaussian mixture parameters for case and control data separately. Monte Carlo simulations are used to generate ROC curves from the distribution parameters. The Monte-Carlo ROC curves are then averaged together to create a non-closed form of a smooth ROC curve ([Cheam & McNicholas, 2016](#)).

Zou and Hall, 2000 propose a Box-Cox transformation of data and then modeling the transformed case data and transformed control data as single Gaussian distributions ([Zou & Hall, 2000](#)).

In this paper, a parametric smoothing technique called Bimixt is introduced that implements both mixture modelling and transformation by modelling Box-Cox transformed case-

control data as Gaussian mixtures. This approach allows us to model heterogeneous biomarker data that is not necessarily Gaussian using a parametric approach to better understand subpopulations and to deal with small sample sizes.

### 4.3 Methods

A novel method for modeling case-control data for individual biomarkers is developed. In this approach, Box-Cox transformation ([Box & Cox, 1964](#)) is applied to the data and the transformed data is then modeled as a Gaussian mixture with at most four components. Maximum likelihood estimates of the transformation parameters and Gaussian component parameters are developed.

#### 4.3.1 Four Component Model

The Box-Cox transformation is a power transformation that has a normalizing effect on data. We will denote the Box-Cox transformation of random variable  $x$  by transformation parameter  $\lambda$  as  $bc(x|\lambda)$ . The transformed variable is defined as follows:

$$bc(x|\lambda) = \begin{cases} \frac{x^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \ln(\lambda), & \lambda = 0 \end{cases}$$

A transformed random variable  $x$  from the four component (4c) model is distributed as the following Gaussian mixture:

$$bc(x|\lambda) \sim \begin{cases} N(\mu_{ctrl_1}, \sigma_{ctrl_1}, \mu_{ctrl_2}, \sigma_{ctrl_2}, \pi_{ctrl}), & x \text{ is a control} \\ N(\mu_{cs_1}, \sigma_{cs_1}, \mu_{cs_2}, \sigma_{cs_2}, \pi_{cs}), & x \text{ is a case} \end{cases}$$

An example four component probability density plot is shown in Figure 4-1. It consists of lower and upper control components and lower and upper case components. In terms of a biomarker the upper components reflect subjects that have high expression levels of the marker while the lower components capture subjects that have low expression of the marker.



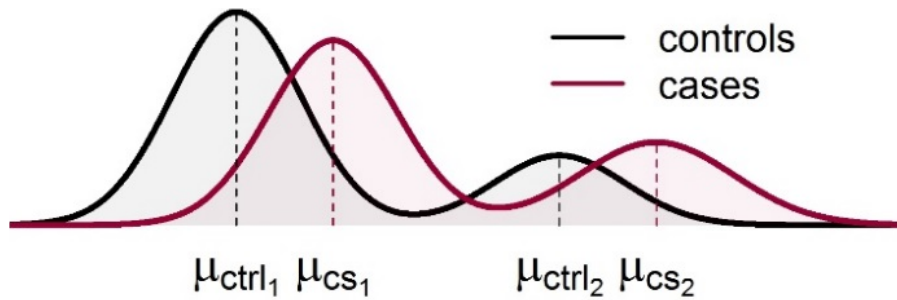


Figure 4-1 Four Component Density Plot

An example probability density function of a four-component model.

The 4c model assumes both case and control data are heterogeneous. Of the heterogeneity models that we implement, the 4c model allows the most flexibility and requires the most parameters. Estimation of the four-component model requires a total of 11 free parameters listed in Table 4.1.

**Table 4.1 Four Component Model Parameters**

Parameter	Description
$\mu_{ctrl_1}$	Mean of lower control component
$\mu_{ctrl_2}$	Mean of upper control component
$\mu_{cs_1}$	Mean of lower case component
$\mu_{cs_2}$	Mean of upper case component
$\sigma_{ctrl_1}$	Standard deviation of lower control component
$\sigma_{ctrl_2}$	Standard deviation of upper control component
$\sigma_{cs_1}$	Standard deviation of lower case component
$\sigma_{cs_2}$	Standard deviation of upper case component
$\pi_{ctrl}$	Proportion of controls in lower component
$\pi_{cs}$	Proportion of cases in lower component
$\lambda$	Box-Cox transformation parameter

#### 4.3.2 Special Cases of Four Component Model

Different heterogeneity assumptions can be reflected by using various restrictions on the 4c model parameters. For example, the classic binormal model assumes no heterogeneity is present in the cases or controls. The Box-Cox transformed binormal model is the special case of the 4c model where  $\mu_{ctrl_1} = \mu_{ctrl_2}$ ,  $\mu_{cs_1} = \mu_{cs_2}$ ,  $\sigma_{ctrl_1} = \sigma_{ctrl_2}$ , and  $\sigma_{cs_1} = \sigma_{cs_2}$ .

Another special case is the case where the lower case and control components are the same, but the upper components are allowed to vary. In this case,  $\mu_{ctrl_1} = \mu_{cs_1}$  and  $\sigma_{ctrl_1} = \sigma_{cs_1}$ . An example of this scenario is shown in the density plot in Figure 4-2.

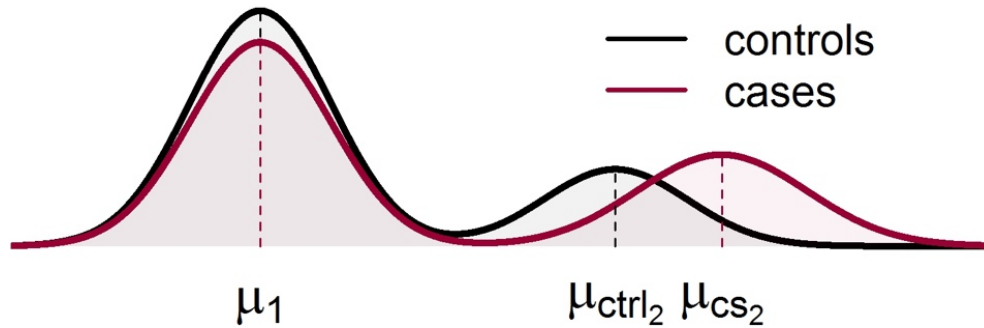


Figure 4-2 Special Case of Four component Density Plot

An example probability density function of a special case of the four component model where the lower cases and lower controls are constrained to the same component.

Special cases can be estimated with fewer parameters, and thus can be more efficient if they are reflective of the data. To determine if a special case is a better fit than the more general 4c model, a likelihood ratio test can be performed to compare the maximum likelihood estimates of the two competing models. This can also give insight into the properties of the biomarker. For example, if the binormal model is a better fit than the 4c by the likelihood ratio test for a particular biomarker, the researcher can be confident the biomarker does not exhibit heterogeneity in the cohort under consideration.

#### 4.3.3 Maximum Likelihood Model Estimation

Model estimation consists of a nested optimization of the Box-Cox parameter and the two-component Gaussian mixture model parameters.

##### 4.3.3.1 Four Component Model Estimation

The Box-Cox parameter  $\lambda$  is estimated using an optimization technique proposed by Nelder and Mead ([Nelder & Mead, 1965](#)) and implemented in the stats R Package ([R Core Team,](#)

[2019](#)). During each iteration of the search for the optimal  $\lambda$ , the data are transformed using the current  $\lambda$  value. Mixture model parameters are then estimated for the transformed data using a finite mixture modelling application of the EM algorithm.

The EM algorithm finds the maximum likelihood estimates for the mixture model parameters by treating the proportion parameters ( $\pi_{ctrl}$  and  $\pi_{cs}$ ) as missing data. Initial estimates for the parameters are obtained using k-means clustering ([Hartigan & Wong, 1979](#)) implemented in the stats R package. The log likelihood value of the fit from the EM algorithm is computed with an adjustment for the additional estimation of  $\lambda$ . The final parameters are chosen to maximize this adjusted log likelihood value.

#### 4.3.3.2 Special Case Model Estimation

For the binormal model proportion estimates are not necessary and so the maximum likelihood estimates can be computed directly from the means and standard deviations of the transformed data. For other special cases, constraints are applied as required. For example, if the model requires  $\mu_{ctrl_1} = \mu_{cs_1}$  and  $\sigma_{ctrl_1} = \sigma_{cs_1}$  (as in the Figure 4-2 scenario) then the case and control data are combined before EM estimates are obtained.

#### 4.3.4 Area Under the Curve Derivation

A closed form function of the area under the curve is derived for the parametric four-component model using moment generating functions.

Let  $X \sim N(0, 1)$  and  $Y \sim N(\mu, \sigma^2)$  be random variables describing continuous test results, where  $X$  denotes a random variable corresponding to a negative test result and  $Y$  denotes a random variable corresponding to a positive test result. It has been shown ([Pepe, 2003](#)) that:

$$AUC = \Pr [y > x] = \Pr [y - x > 0]$$

Thus, the AUC is equal to the difference between the probability density functions of  $Y$  and  $X$ . The derivation of this value for the four component curve utilizing moment generating functions to describe the probability density functions follows:

## AUC of the Quadrinormal ROC Curve

Let  $x$  be random variable with probability  $p_1$  of being normally distributed with mean  $\mu_1$  and variance  $\sigma_1^2$  and probability  $1 - p_1$  of being normally distributed with mean  $\mu_2$  and variance  $\sigma_2^2$ . Then the probability density function of  $x$  is given by:

$$N(x; \mu_1, \sigma_1^2, \mu_2, \sigma_2^2, p_1) = p_1 \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + (1-p_1) \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}$$

Furthermore, the moment generating function is given by:

$$\begin{aligned} M_x(t) &= \int_{-\infty}^{\infty} e^{xt} \left( p_1 \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + (1-p_1) \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} \right) dx \\ &= p_1 \int_{-\infty}^{\infty} e^{xt} \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} dx + (1-p_1) \int_{-\infty}^{\infty} e^{xt} \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} dx \end{aligned}$$

Which is the sum of two normal distribution moment generating functions weighted by respective probabilities. After integration, the moment generating function of a normal distribution  $N(\mu, \sigma)$  is given by  $M(t) = e^{\mu t} e^{\frac{1}{2}\sigma^2 t^2}$ . Therefore,

$$M_x(t) = p_1 e^{\mu_1 t} e^{\frac{1}{2}\sigma_1^2 t^2} + (1-p_1) e^{\mu_2 t} e^{\frac{1}{2}\sigma_2^2 t^2}$$

Now suppose we have another independently distributed random variable  $y$  with distribution  $N(y; \mu_3, \sigma_3^2, \mu_4, \sigma_4^2, p_2)$ . Then we can find the distribution of  $x-y$  by looking at the moment generating function  $M_{x-y}(t)$  with  $N(-y; -\mu_3, \sigma_3^2, \mu_4, \sigma_4^2, p_2)$

$$\begin{aligned} M_{x-y}(t) &= M_x(t) * M_{-y}(t) \\ &= (p_1 e^{\mu_1 t} e^{\frac{1}{2}\sigma_1^2 t^2} + (1-p_1) e^{\mu_2 t} e^{\frac{1}{2}\sigma_2^2 t^2}) (p_2 e^{-\mu_3 t} e^{\frac{1}{2}\sigma_3^2 t^2} + (1-p_2) e^{-\mu_4 t} e^{\frac{1}{2}\sigma_4^2 t^2}) \\ &= p_1 p_2 e^{\mu_1 t} e^{\frac{1}{2}\sigma_1^2 t^2} e^{-\mu_3 t} e^{\frac{1}{2}\sigma_3^2 t^2} + p_1 (1-p_2) e^{\mu_1 t} e^{\frac{1}{2}\sigma_1^2 t^2} e^{-\mu_4 t} e^{\frac{1}{2}\sigma_4^2 t^2} \\ &\quad + (1-p_1) p_2 e^{\mu_2 t} e^{\frac{1}{2}\sigma_2^2 t^2} e^{-\mu_3 t} e^{\frac{1}{2}\sigma_3^2 t^2} + (1-p_1) (1-p_2) e^{\mu_2 t} e^{\frac{1}{2}\sigma_2^2 t^2} e^{-\mu_4 t} e^{\frac{1}{2}\sigma_4^2 t^2} \\ &= p_1 p_2 e^{(\mu_1 - \mu_3) t} e^{\frac{(\sigma_1^2 + \sigma_3^2) t^2}{2}} + p_1 (1-p_2) e^{(\mu_1 - \mu_4) t} e^{\frac{(\sigma_1^2 + \sigma_4^2) t^2}{2}} \\ &\quad + (1-p_1) p_2 e^{(\mu_2 - \mu_3) t} e^{\frac{(\sigma_2^2 + \sigma_3^2) t^2}{2}} + (1-p_1) (1-p_2) e^{(\mu_2 - \mu_4) t} e^{\frac{(\sigma_2^2 + \sigma_4^2) t^2}{2}} \end{aligned}$$

This is of the same form as the moment generating function of the distribution given by  $N(w; \mu_1, \sigma_1^2, \mu_2, \sigma_2^2, \mu_3, \sigma_3^2, \mu_4, \sigma_4^2, p_1, p_2, p_3)$ , which can be found by integrating a sum of four normal distribution moment generating functions

#### 4.3.5 Simulation Studies

To assess the ability of the 4c model to estimate the true distribution, heterogeneous case/control data of varying sample sizes is simulated from normal mixture distributions with the following parameters:

$$cases \sim N(\mu_{ctrl_1} = 6, \quad \mu_{ctrl_2} = 14, \quad \sigma_{ctrl_1} = 1, \quad \sigma_{ctrl_2} = 1, \pi_{ctrl} = .75)$$

$$controls \sim N(\mu_{ctrl_1} = 5, \quad \mu_{ctrl_2} = 10, \quad \sigma_{ctrl_1} = 1, \quad \sigma_{ctrl_2} = 1, \pi_{ctrl} = .85)$$

The inverse Box-Cox transformation is then applied to the simulated data setting  $\lambda$  to 0.5.

Bimixt model parameters are then estimated on the simulated data set and the corresponding ROC curves are generated. For comparison, the density smoothed and binormal smoothed ROC curves are also generated.

The ROC curve based on the true distribution of parameters is also simulated. The area between curves (ABC) is then calculated between the true ROC curve and the smoothed ROC curves using the trapezoid rule.

#### 4.3.6 Proteomic Biomarker Screening Study Application

To determine how well the 4c model fits real world data, the model is applied to normalized data from a proteomic array screening experiment of 45 basal-like breast cancer patients and 45 controls ([Wang et al., 2015](#)). The subjects for the protein array came from the Polish Breast Cancer study ([Garcia-Closas et al., 2006](#)).

For each subject, 9,180 proteins were screened. The abundance of each protein was measured using fluorescent intensity values normalized to a control spot on the array. For each protein, the 4c model was fit using the protein expression data from the 90 subjects.

To see how many of the proteins the 4c model fit, a modified one-sample Kolmogorov-Smirnov (KS) test ([Massey Jr, 1951](#)) is performed. The null hypothesis of this test is that the true cumulative distribution function (CDF) of the data is equal to the 4c CDF. The KS test p-values are calculated for all 9,180 proteins as follows:

1. Estimate 4c model parameters from the case/control data
2. Apply estimated case parameters to case CDF and estimated control parameters to control CDF
3. Calculate the KS p-value for the cases compared to the estimated case CDF and a KS p-value for the controls compared to the estimated control CDF.
4. Take the minimum of the two p-values as the final p-value (this is conservative as a larger p-value shows a better fit).

After the KS test p-values are calculated for all 9,180 proteins, they are adjusted for multiple comparison using the Benjamini and Hochberg method ([Benjamini & Hochberg, 1995](#)).

#### 4.4 Results

##### 4.4.1 Simulation Study

The ROC curves associated with each smoothing method were generated for a simulated data set of 30 cases and 30 controls, and ABCs were calculated. The resulting smoothed ROC curves are overlaid on top of the true ROC curves in Figure 4-3.

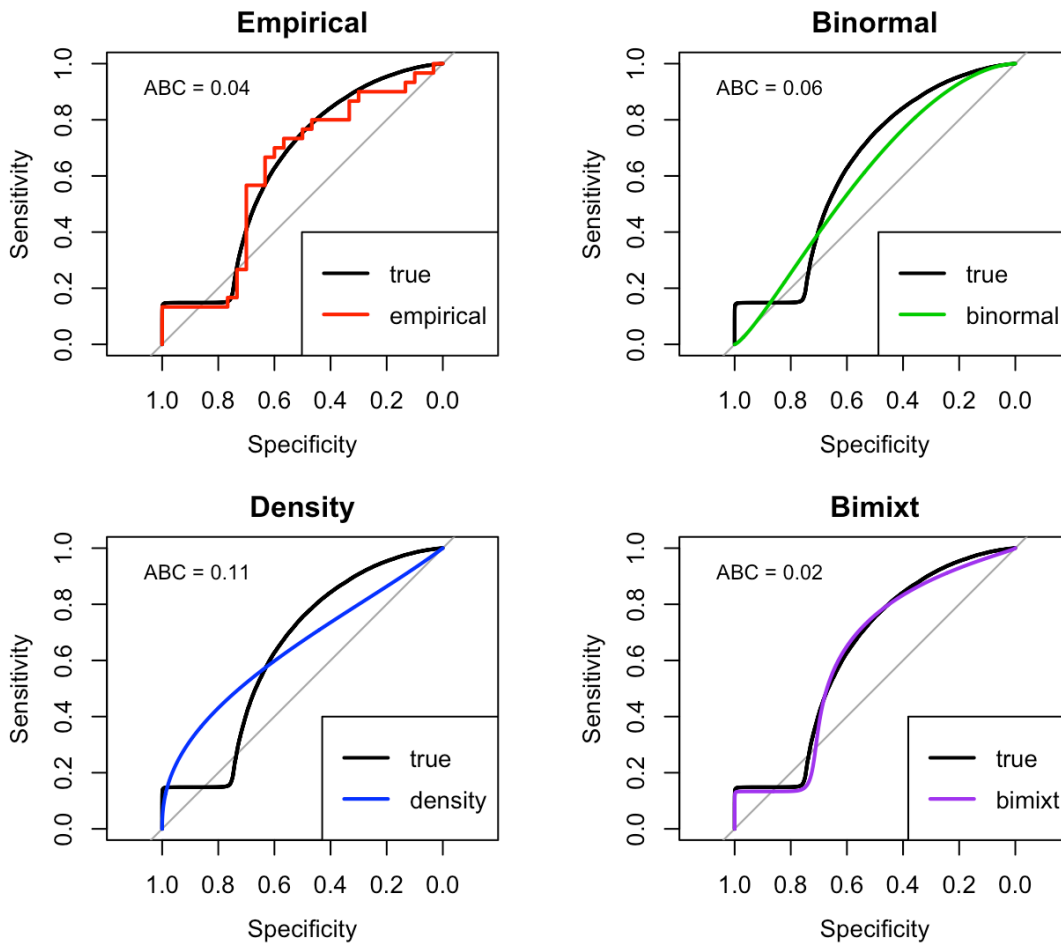


Figure 4-3 Comparison of ROC Curve Estimates for Non-Gaussian, Bimodal Data

The figure compares, from top left to bottom right, the empirical method, the binormal method, the kernel density method, and the bimixt method for estimating ROC curves for non-Gaussian bimodal data with 30 cases and 30 controls. The true ROC curve for the population the data were sampled from is shown in black. Area between the ROC curves (ABC) is calculated between the true ROC curve and each of the estimated ROC curves to show how well the method used to estimate the ROC curve on the sample set approximates the true population curve.

Data sets were then simulated for sample sizes ranging from 20 (10 cases and 10 controls) to 100 (50 cases and 50 controls). The effects of sample size on the ABC across each of the smoothing methods are shown in Figure 4-4.

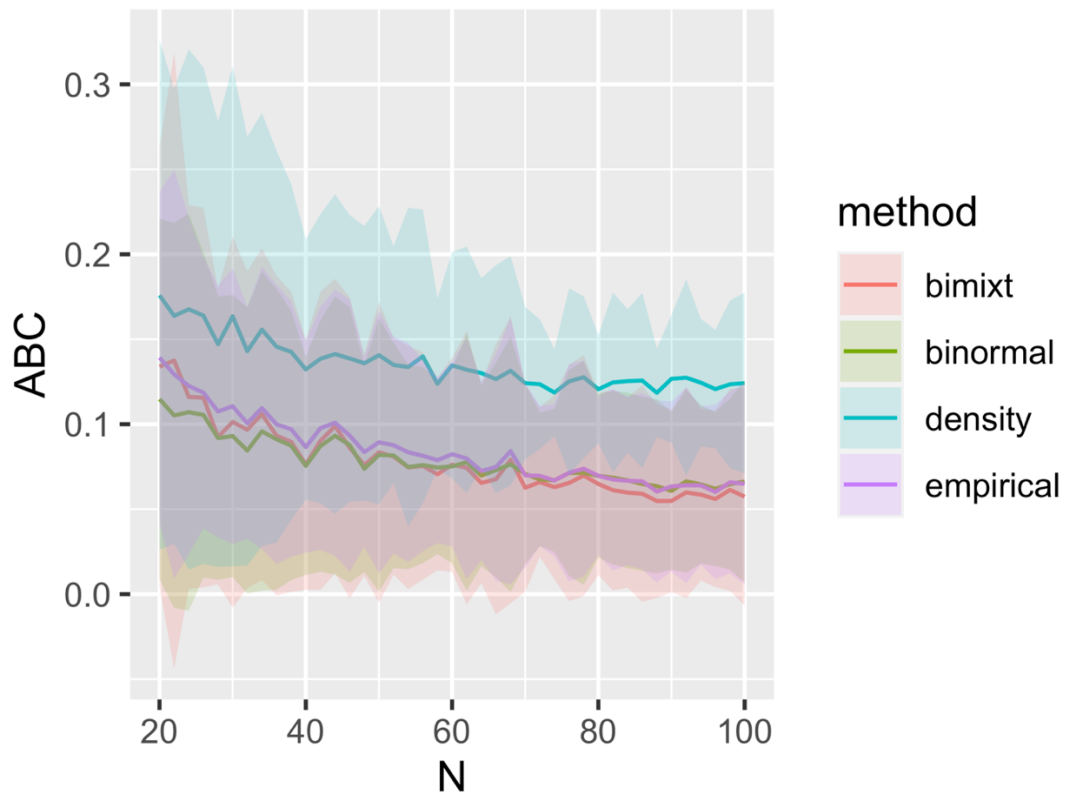


Figure 4-4 Effects of Sample Size on Area Between ROC Curves

The figure shows how sample size effects the area between smoothed and true ROC curves for non-Gaussian bimodal data. The solid lines show the ABC averaged over 50 replicates, and the ribbons show the average  $\pm 2$  standard deviations.

#### 4.4.2 Breast Cancer Data

The 9,180 proteins in the breast cancer screening array were fit with the 4c model. The parameters estimated during model fitting were used to compare the 4c model CDF to the raw data using the `ks.test` function in R. The KS test P-values were calculated on all 9,180 proteins in the breast cancer screening array, and adjusted for multiple comparisons. All 9,180 proteins had high p-values (range of 0.57 to 0.99) indicating that the 4c distribution function with the Bimixt estimated parameters was a reasonable fit for all potential markers. The distribution of the adjusted p-values is shown in Figure 4-5.



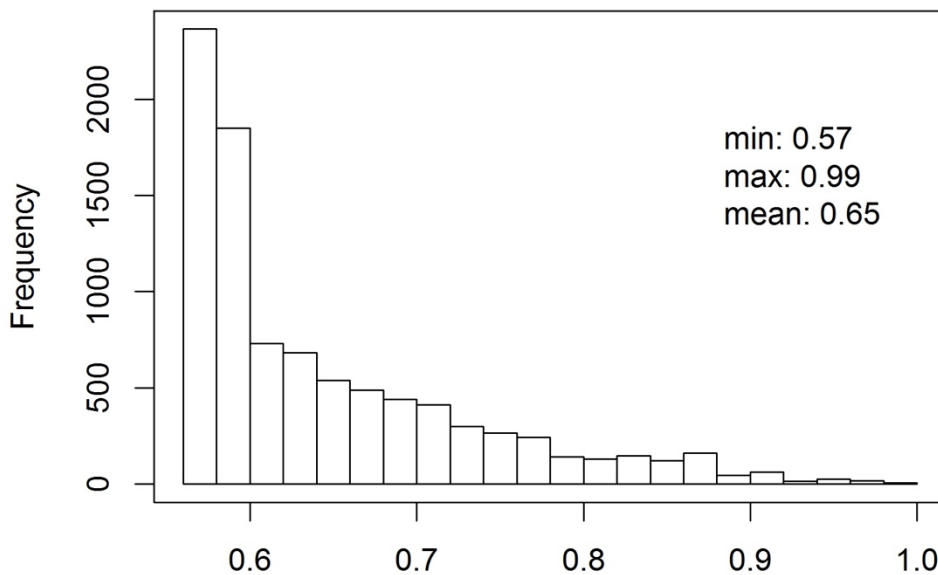


Figure 4-5 Histogram of P-Values for Bimixt Fit of Proteomic Data

The histogram shows the distribution of the adjusted p-values for the KS test applied to the 9,180 proteins from the proteomic breast cancer screen. The p-values range from 0.57 – 0.99 with a mean of 0.65.

#### 4.5 Discussion

The model presented in this chapter balances the need to represent Gaussian and non-Gaussian, bimodal data with the ability to handle small sample sizes. The model fits 100% of empirical data from a proteomic breast cancer screen. It can also be used to estimate smooth receiver operator characteristic (ROC) curves for individual markers.

Simulation studies show that the bimixt method provides an advantage for estimating AUC compared to the other smoothing techniques, especially for moderately small sample sizes.

Mixture modeling is a natural approach for modeling latent heterogeneity. While there are other methods for estimating mixture models, the novelty of the method presented in this chapter is that it facilitates mixture modeling of non-Gaussian data by incorporating the Box-Cox transformation parameter in the optimization process. The algorithm also supports different variations of constraints on the heterogeneity, with a 4c model at one end and a binormal model at the other.

One drawback of the method is that it does not allow for modelling more than four components. However, for many biological applications 4 components should be sufficient. Additionally, the more components we estimate, the more likely we are to overfit the data and potentially estimate noise or outliers as their own components. This would not generalize to new data well. Finally, four component models are easily interpretable; it is easy to interpret two components biologically as high expressers or low expressers of a protein.

## CHAPTER 5

### CONCLUSION AND FUTURE WORK

#### 5.1 Family Rank

##### 5.1.1 Conclusion

In chapter 2, a novel method for ranking features was presented that incorporated both empirical scores and graphical domain knowledge. Through simulations studies, it was shown that the method outperformed other state-of-the-art graphical ranking methods in scenarios where large feature spaces and small sample sizes were present.

Furthermore, the method was applied to real-world oncology data to demonstrate the utility of the method in predicting response to chemotherapy from gene expression data. The method showed that mutations in cellular meiosis and cell cycle pathways were predictive of response to chemotherapy.

The primary benefit of the method is that it may find more appropriate predictors in data sets where there is high degree of complexity between true predictors and outcome and the sample size is too small to differentiate true predictors from noise given the degree of complexity.

##### 5.1.2 Future Work

In this research, simulation studies were used to show the number of samples required to detect true predictors in the scenario where a third of the positive samples were defined by one set of 5 unique predictors, a third were defined by a second set of 5 unique predictors, and the last third were defined by a third set of 5 unique predictors. In this case, the complexity was defined by 3 unique sets of 5 interacting predictors. While these numbers were meant to imitate the heterogeneity seen in oncology data sets, the selection of 3 subtypes and 5 predictors to define each subtype was arbitrary. In different domains, researchers may have expectations for either more or less complexity in their datasets. Therefore, further analysis to explore additional complexity scenarios, and the number of samples required to detect true predictors in such scenarios is warranted.

Additionally, in the oncology example provided in this research, protein-protein interaction data from the STRING database was used as input for analysis. While all protein-protein interactions were utilized in this research, there may be instances where only a subset of protein-protein interactions are relevant to the research. For example, if a researcher is interested in only a single pathway's effect on outcome, it may be of use to look at the impact of utilizing sub-networks of protein-protein interactions as input to family rank.

## 5.2 Population Bias in MSI

### 5.2.1 Conclusion

In chapter 3, novel work on NGS based methods to calculate MSI was presented. It was shown that the method developed within this chapter reduced the apparent bias observed in NGS results which routinely inflated the false positive rate of MSI-H calls in African American ancestries.

In the initial phase of the research, bias in the human reference genome, which is primarily derived from European DNA, was hypothesized to result in biased estimates of MSI. The hypothesized bias was supported by analysis of MSI in a cohort of pan-tumor specimen with African American genetic ancestry. It was found that FA and IHC methods of MSI analysis that did not rely on alignment to the human reference genome consistently agreed on MSS or MSI-L status, while the NGS method in which data were aligned to the reference genome consistently designated the specimen in the cohort as MSI-H.

The NGS method was then modified by utilizing novel techniques to select a subset of NGS loci that were able to classify MSI without the bias observed in the initial phase. The novel model was verified by cross validation on the training set as well as independent validation on an independent test cohort.

The primary advantage of the model is that it reduces the false positive rate and subsequently increases the specificity of the NGS MSI diagnostic test in the African ancestry population. An NGS diagnostic for MSI is usually preferable to IHC or FA in clinical settings. The

diagnostic allows for more biomarkers to be assessed at once, utilizing less tissue from patients. Therefore, having an equitable test with good specificity and sensitivity is important.

Additionally, MSI specifically is utilized directly in patient care as it is a primary indicator of response to immunotherapies. If a patient has high levels of MSI, they are more likely to respond to immunotherapies. Therefore, false positives in the test may lead patients to receive treatment that may not be effective for them, and waste time that could be used to try other, more promising therapies for their specific genetic signatures.

### 5.2.2 Future Work

Liquid biopsies are a promising technique that offer less invasive diagnostics compared to tissue based approaches. As liquid biopsy assays improve, future assessments of MSI may be able to utilize blood-based normal data for alignment as opposed to normal tissue. If patient samples can be aligned to their own normal genetic signatures, it would eliminate inherent bias from aligning to reference genomes.

Additionally, in the future, alignment techniques that utilize multiple ancestral genomes could be incorporated in the bioinformatics pipelines used to determine MSI calls. For example, if an allele aligns to any of the ancestral genomes assessed, it would not be considered a mutation.

Finally, additional techniques of estimating response to immunotherapies may be developed in the future, such as measurements of neoantigen load via human leukocyte antigen (HLA) analysis. Such techniques can be performed on NGS data, and can be used as additional evidence to support immunotherapy.

## 5.3 Bimixt

### 5.3.1 Conclusion

In chapter 4, a novel approach for generating maximum likelihood estimates of ROC curves for multimodal, non-Gaussian data was presented. In this chapter, the algorithm was motivated by the need to capture heterogeneity in genetic and proteomic biomarkers. Simulation studies were conducted to show that the estimations fit the data as well or better than other state-

of-the-art ROC smoothing methods. Additionally, application of the method to real world proteomic data showed that the method significantly fit 100% of the 9,180 proteins assayed.

The main advantage of the method presented in this chapter is that it can be used to model multimodal Gaussian or non-Gaussian data. This is beneficial particularly in oncology data where genetic heterogeneity is present. Often the expression levels of proteins in a cohort of patients with identical tumor types appear bimodal due to inherent tumor heterogeneity.

Compounding the challenges of heterogeneity are small sample sizes. Estimating ROC curves on small sample sizes can lead to “jumpy” curves, in which a small perturbation in sensitivity may cause a large change in specificity or vice versa. The jumpiness of empirical ROC curves can be mitigated using smoothing techniques, making proper smoothing techniques for non-Gaussian, bimodal data useful.

### 5.3.2 Future Work

Power analysis for biomarker screening studies are necessary to design cost effective and efficient trials. One endpoint in such screening studies is often ROC analysis. Therefore, to properly plan such studies utilizing the methods developed here, future work in how to apply the methods to power analyses are warranted.

## REFERENCES

- Anderson, K. S., & LaBaer, J. (2005). The sentinel within: exploiting the immune system for cancer biomarkers. *Journal of proteome research*, 4(4), 1123-1133.
- Andy Liaw, & Wiener, M. (2002). Classification and Regression by randomForest. *R News*, 2, 3. Retrieved from <https://CRAN.R-project.org/doc/Rnews/>
- Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1), 289-300.
- Boland, C. R., Thibodeau, S. N., Hamilton, S. R., Sidransky, D., Eshleman, J. R., Burt, R. W., . . . Ranzani, G. N. (1998). A National Cancer Institute Workshop on Microsatellite Instability for cancer detection and familial predisposition: development of international criteria for the determination of microsatellite instability in colorectal cancer. *Cancer research*, 58(22), 5248-5257.
- Box, G. E., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2), 211-243.
- Cheam, A. S., & McNicholas, P. D. (2016). Modelling receiver operating characteristic curves using Gaussian mixtures. *Computational Statistics & Data Analysis*, 93, 192-208.
- Consortium, M. (2010). The MicroArray Quality Control (MAQC)-II study of common practices for the development and validation of microarray-based predictive models. *Nature biotechnology*, 28(8), 827.
- Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695. Retrieved from <http://igraph.org>
- Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1), 1-22.
- Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D., & Weingessel, A. (2008). Misc functions of the Department of Statistics (e1071), TU Wien. *R package*, 1, 5-24.
- Dunnwald, L. K., Rossing, M. A., & Li, C. I. (2007). Hormone receptor status, tumor characteristics, and prognosis: a prospective cohort of breast cancer patients. *Breast Cancer Research*, 9(1), 1-10.
- E pluribus unum. (2010). If the human reference genome is to reflect more of the actual genomic diversity in humans, community participation is needed. *Nat Methods*, 7(5), 331.
- Fan, S., Kelly, D. E., Beltrame, M. H., Hansen, M. E. B., Mallick, S., Ranciaro, A., . . . Tishkoff, S. A. (2019). African evolutionary history inferred from whole genome sequence data of 44 indigenous African populations. *Genome Biol*, 20(1), 82. doi:10.1186/s13059-019-1679-2

- Garcia-Closas, M., Brinton, L., Lissowska, J., Chatterjee, N., Peplonska, B., Anderson, W., . . . Blair, A. (2006). Established breast cancer risk factors by clinically important tumour characteristics. *British journal of cancer*, *95*(1), 123-129.
- Genomes Project Consortium. (2010). A map of human genome variation from population-scale sequencing. *Nature*, *467*(7319), 1061.
- Green, D. M., & Swets, J. A. (1966). *Signal detection theory and psychophysics* (Vol. 1): Wiley New York.
- Group, E. B. C. T. C. (2011). Relevance of breast cancer hormone receptors and other factors to the efficacy of adjuvant tamoxifen: patient-level meta-analysis of randomised trials. *The lancet*, *378*(9793), 771-784.
- Hartigan, J. A., & Wong, M. A. (1979). AK-means clustering algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *28*(1), 100-108.
- Huynen, M. A., Snel, B., von Mering, C., & Bork, P. (2003). Function prediction and protein networks. *Current opinion in cell biology*, *15*(2), 191-198.
- Jiricny, J. (2006). The multifaceted mismatch-repair system. *Nature reviews Molecular cell biology*, *7*(5), 335-346.
- Kanehisa, M. (2002). *The KEGG database*. Paper presented at the Novartis Foundation Symposium.
- Karczewski, K., & Francioli, L. (2017). The genome aggregation database (gnomAD). *MacArthur Lab*.
- Kent, W. J., Sugnet, C. W., Furey, T. S., Roskin, K. M., Pringle, T. H., Zahler, A. M., & Haussler, D. (2002). The human genome browser at UCSC. *Genome Res*, *12*(6), 996-1006. doi:10.1101/gr.229102
- Kuhn, M. (2015). Caret: classification and regression training. *Astrophysics Source Code Library*, ascl: 1505.1003.
- Le, D. T., Durham, J. N., Smith, K. N., Wang, H., Bartlett, B. R., Aulakh, L. K., . . . Lubner, B. S. (2017). Mismatch repair deficiency predicts response of solid tumors to PD-1 blockade. *Science*, *357*(6349), 409-413.
- Le, D. T., Uram, J. N., Wang, H., Bartlett, B. R., Kemberling, H., Eyring, A. D., . . . Laheru, D. (2015). PD-1 blockade in tumors with mismatch-repair deficiency. *New England Journal of Medicine*, *372*(26), 2509-2520.
- Lee, V., Murphy, A., Le, D. T., & Diaz Jr, L. A. (2016). Mismatch repair deficiency and response to immune checkpoint blockade. *The oncologist*, *21*(10), 1200.
- Lek, M., Karczewski, K. J., Minikel, E. V., Samocha, K. E., Banks, E., Fennell, T., . . . Exome Aggregation, C. (2016). Analysis of protein-coding genetic variation in 60,706 humans. *Nature*, *536*(7616), 285-291. doi:10.1038/nature19057



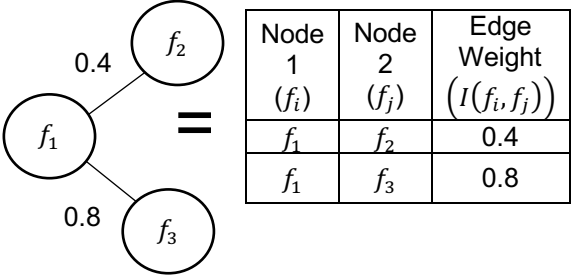
- Lemery, S., Keegan, P., & Pazdur, R. (2017). First FDA Approval Agnostic of Cancer Site - When a Biomarker Defines the Indication. *N Engl J Med*, *377*(15), 1409-1412. doi:10.1056/NEJMp1709968
- Lind, M. (2008). Principles of cytotoxic chemotherapy. *Medicine*, *36*(1), 19-23.
- Marcus, L., Lemery, S. J., Keegan, P., & Pazdur, R. (2019). FDA Approval Summary: Pembrolizumab for the Treatment of Microsatellite Instability-High Solid Tumors. *Clin Cancer Res*, *25*(13), 3753-3758. doi:10.1158/1078-0432.CCR-18-4070
- Massey Jr, F. J. (1951). The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association*, *46*(253), 68-78.
- McConechy, M., Talhouk, A., Li-Chang, H., Leung, S., Huntsman, D., Gilks, C. B., & McAlpine, J. (2015). Detection of DNA mismatch repair (MMR) deficiencies by immunohistochemistry can effectively diagnose the microsatellite instability (MSI) phenotype in endometrial carcinomas. *Gynecologic oncology*, *137*(2), 306-310.
- Mering, C. v., Huynen, M., Jaeggi, D., Schmidt, S., Bork, P., & Snel, B. (2003). STRING: a database of predicted functional associations between proteins. *Nucleic acids research*, *31*(1), 258-261.
- Metz, C. E., Herman, B. A., & Shen, J. H. (1998). Maximum likelihood estimation of receiver operating characteristic (ROC) curves from continuously-distributed data. *Statistics in medicine*, *17*(9), 1033-1053.
- Morrison, J. L., Breitling, R., Higham, D. J., & Gilbert, D. R. (2005). GeneRank: using search engine technology for the analysis of microarray experiments. *BMC Bioinformatics*, *6*, 233. doi:10.1186/1471-2105-6-233
- Nabavi, S., Schmolze, D., Maitituoheti, M., Malladi, S., & Beck, A. H. (2016). EMDomics: a robust and powerful method for the identification of genes differentially expressed between heterogeneous classes. *Bioinformatics*, *32*(4), 533-541.
- Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The computer journal*, *7*(4), 308-313.
- Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The pagerank citation ranking: Bringing order to the web*. Retrieved from
- Pardoll, D. M. (2012). The blockade of immune checkpoints in cancer immunotherapy. *Nature Reviews Cancer*, *12*(4), 252-264.
- Pepe, M. S. (2003). *The statistical evaluation of medical tests for classification and prediction*: Medicine.
- Popovici, V., Chen, W., Gallas, B. D., Hatzis, C., Shi, W., Samuelson, F. W., . . . Nikolskaya, T. (2010). Effect of training-sample size and classification difficulty on the accuracy of genomic predictors. *Breast Cancer Research*, *12*(1), R5.

- R Core Team. (2019). R: A language and environment for statistical computing. *R Foundation for Statistical Computing*. Retrieved from <https://www.R-project.org/>
- Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J. C., & Muller, M. (2011). pROC: an open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics*, *12*, 77. doi:10.1186/1471-2105-12-77
- Saul, M. (2021). FamilyRank: Algorithm for Ranking Predictors Using Graphical Domain Knowledge (Version 1.0). CRAN. Retrieved from <https://CRAN.R-project.org/package=FamilyRank>
- Saul, M., & Dinu, V. (2021). Family Rank: A graphical domain knowledge informed feature ranking algorithm. *Bioinformatics*. doi:10.1093/bioinformatics/btab387
- Saul, M., Poorman, K., Tae, H., Vanderwalde, A., Stafford, P., Spetzler, D., . . . Swensen, J. (2020). Population bias in somatic measurement of microsatellite instability status. *Cancer medicine*, *9*(17), 6452-6460.
- Sherman, R. M., Forman, J., Antonescu, V., Puiu, D., Daya, M., Rafaels, N., . . . Salzberg, S. L. (2019). Assembly of a pan-genome from deep sequencing of 910 humans of African descent. *Nat Genet*, *51*(1), 30-35. doi:10.1038/s41588-018-0273-y
- Slamon, D. J., Leyland-Jones, B., Shak, S., Fuchs, H., Paton, V., Bajamonde, A., . . . Pegram, M. (2001). Use of chemotherapy plus a monoclonal antibody against HER2 for metastatic breast cancer that overexpresses HER2. *New England Journal of Medicine*, *344*(11), 783-792.
- Strimbu, K., & Tavel, J. A. (2010). What are biomarkers? *Current Opinion in HIV and AIDS*, *5*(6), 463.
- Szklarczyk, D., Franceschini, A., Wyder, S., Forslund, K., Heller, D., Huerta-Cepas, J., . . . Tsafou, K. P. (2015). STRING v10: protein–protein interaction networks, integrated over the tree of life. *Nucleic acids research*, *43*(D1), D447-D452.
- Vanderwalde, A., Spetzler, D., Xiao, N., Gatalica, Z., & Marshall, J. (2018). Microsatellite instability status determined by next-generation sequencing and compared with PD-L1 and tumor mutational burden in 11,348 patients. *Cancer Med*, *7*(3), 746-756. doi:10.1002/cam4.1372
- Venables, W. N., & Ripley, B. D. (2002). *Modern Applied Statistics with S* (Fourth ed.). New York: Springer.
- Von Mering, C., Jensen, L. J., Snel, B., Hooper, S. D., Krupp, M., Foglierini, M., . . . Bork, P. (2005). STRING: known and predicted protein–protein associations, integrated and transferred across organisms. *Nucleic acids research*, *33*(suppl\_1), D433-D437.
- Wallstrom, G., Anderson, K. S., & LaBaer, J. (2013). Biomarker discovery for heterogeneous diseases. *Cancer Epidemiology and Prevention Biomarkers*, *22*(5), 747-755.

- Wang, J., Figueroa, J. D., Wallstrom, G., Barker, K., Park, J. G., Demirkan, G., . . . LaBaer, J. (2015). Plasma autoantibodies associated with basal-like breast cancers. *Cancer Epidemiology and Prevention Biomarkers*, 24(9), 1332-1340.
- Winerip, M., Wallstrom, G., & LaBaer, J. (2015). Bimixt: Estimates Mixture Models for Case-Control Data. R package version 1.0. Retrieved from <https://CRAN.R-project.org/package=bimixt>
- Zou, K. H., & Hall, W. (2000). Two transformation models for estimating an ROC curve derived from continuous data. *Journal of Applied Statistics*, 27(5), 621-631.
- Zou, K. H., Hall, W., & Shapiro, D. E. (1997). Smooth non-parametric receiver operating characteristic (ROC) curves for continuous diagnostic tests. *Statistics in medicine*, 16(19), 2143-2156.
- Zweig, M. H., & Campbell, G. (1993). Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine. *Clinical chemistry*, 39(4), 561-577.

APPENDIX A  
FAMILY RANK WORKED EXAMPLE

I Example input

Input	Description	Value									
Features	Vector of features to be ranked	$\vec{f} = (f_1, f_2, f_3)$									
Empirical Feature Score	Vector of empirical feature scores calculated by user, e.g. AUCs for data with numeric features and binary response	$\vec{s} = (0.6, 0.2, 0.9)$									
Graphical Domain Knowledge	Graphical object containing interaction scores between features.	 <table border="1" data-bbox="1047 766 1372 966"> <thead> <tr> <th>Node 1 (<math>f_i</math>)</th> <th>Node 2 (<math>f_j</math>)</th> <th>Edge Weight (<math>I(f_i, f_j)</math>)</th> </tr> </thead> <tbody> <tr> <td><math>f_1</math></td> <td><math>f_2</math></td> <td>0.4</td> </tr> <tr> <td><math>f_1</math></td> <td><math>f_3</math></td> <td>0.8</td> </tr> </tbody> </table>	Node 1 ( $f_i$ )	Node 2 ( $f_j$ )	Edge Weight ( $I(f_i, f_j)$ )	$f_1$	$f_2$	0.4	$f_1$	$f_3$	0.8
Node 1 ( $f_i$ )	Node 2 ( $f_j$ )	Edge Weight ( $I(f_i, f_j)$ )									
$f_1$	$f_2$	0.4									
$f_1$	$f_3$	0.8									
Damping Parameter	Value between 0 and 1 that determines how much weight to give to the domain knowledge.	$d = 0.5$									
Tolerance	Stopping criteria. If the weighted score of a feature added to a family is less than or equal to the tolerance value, iterations stop.	$tol = 0$									
Families to Build	Value indicating number of families to be built. Must be between 1 and the number of features to rank. If less than number of features, the features with highest empirical scores are used to initiate family building.	$n. families = 3$									

## II Example Equations

Equations for calculations used to build families	
$w_{i,j_k} = \begin{cases} s_k, & i = 1, k = j \\ 0, & i = 1, k \neq j \\ 0, & i > 1, k = m \\ (1 - d) * s_k + d * I(f_k, f_m), & i = 2, k \neq m \\ (1 - d) * w_{i-1,j_k} + d * I(f_k, f_m), & i > 2, k \neq m \end{cases} \quad (1)$	<p>where</p> $f_m = \arg \max_{f_k \in \hat{f}} \bar{w}_{i-1,j}$
$\Omega_{i,j} = \max (\bar{w}_{i,j}) \quad (2)$	
$\Lambda_{i,j} = \arg \max_{f_k \in \hat{f}} \bar{w}_{i,j} \quad (3)$	

III Example: Building Family 1

Step			Calculation				Stop
Family ( <i>j</i> )	Iteration ( <i>i</i> )	Weight ( <i>k</i> )	$w_{i,j,k}$	$\vec{w}_{i,j}$	$\Omega_{i,j}$	$\Lambda_{i,j}$	Y/N
1	1	1	$w_{1,1,1} = s_1$ $= 0.6$	$\vec{w}_{1,1}$ $= (0.6, 0, 0)$	$\Omega_{1,1} = \max(\vec{w}_{1,1})$ $= 0.6$	$\Lambda_{1,1}$ $= \operatorname{argmax}(\vec{w}_{1,1})$ $= f_1$	No
		2	$w_{1,1,2} = 0$				
		3	$w_{1,1,3} = 0$				
	2	1	$w_{2,1,1} = 0$	$\vec{w}_{2,1}$ $= (0, 0.3, 0.85)$	$\Omega_{2,1} = \max(\vec{w}_{2,1})$ $= 0.85$	$\Lambda_{2,1}$ $= \operatorname{argmax}(\vec{w}_{2,1})$ $= f_3$	No
		2	$w_{2,1,2}$ $= (1 - d) * s_2 + d * I(f_1, f_2)$ $= (1 - 0.5) * 0.2 + 0.5 * 0.4$ $= 0.3$				
		3	$w_{2,1,3}$ $= (1 - d) * s_3 + d * I(f_3, f_1)$ $= (1 - 0.5) * 0.9 + 0.5 * 0.8$ $= 0.85$				
	3	1	$w_{3,1,1}$ $= (1 - d) * w_{2,1,1} + d * I(f_1, f_3)$ $= (1 - 0.5) * 0 + 0.5 * 0.8$ $= 0.4$	$\vec{w}_{3,1}$ $= (0.4, 0.15, 0)$	$\Omega_{3,1} = \max(\vec{w}_{3,1})$ $= 0.6$	$\Lambda_{3,1}$ $= \operatorname{argmax}(\vec{w}_{3,1})$ $= f_1$	Yes $f_1$ already selected
		2	$w_{3,1,2}$ $= (1 - d) * w_{2,1,2} + d * I(f_2, f_3)$ $= (1 - 0.5) * 0.3 + 0.5 * 0$ $= 0.15$				
		3	$w_{3,1,3} = 0$				

IV Example: Building Family 2

Step			Calculation				Stop
Family ( <i>j</i> )	Iteration ( <i>i</i> )	Weight ( <i>k</i> )	$w_{i,jk}$	$\vec{w}_{i,j}$	$\Omega_{i,j}$	$\Lambda_{i,j}$	Y/N
2	1	1	$w_{1,21} = 0$	$\vec{w}_{1,2} = (0, 0.2, 0)$	$\Omega_{1,2} = \max(\vec{w}_{1,2}) = 0.2$	$\Lambda_{1,2} = \operatorname{argmax}(\vec{w}_{1,2}) = f_2$	No
		2	$w_{1,22} = s_2 = 0.2$				
		3	$w_{1,23} = 0$				
	2	1	$w_{2,21} = (1-d) * s_1 + d * I(f_1, f_2) = (1-0.5) * 0.6 + 0.5 * 0.4 = 0.5$	$\vec{w}_{2,2} = (0.5, 0, 0.45)$	$\Omega_{2,2} = \max(\vec{w}_{2,2}) = 0.5$	$\Lambda_{2,2} = \operatorname{argmax}(\vec{w}_{2,2}) = f_1$	No
		2	$w_{2,22} = 0$				
		3	$w_{2,23} = (1-d) * s_3 + d * I(f_3, f_2) = (1-0.5) * 0.9 + 0.5 * 0 = 0.45$				
	3	1	$w_{3,21} = 0$	$\vec{w}_{3,2} = (0, 0.2, 0.625)$	$\Omega_{3,2} = \max(\vec{w}_{3,2}) = 0.625$	$\Lambda_{3,2} = \operatorname{argmax}(\vec{w}_{3,2}) = f_3$	Yes all features selected
		2	$w_{3,22} = (1-d) * w_{2,22} + d * I(f_2, f_1) = (1-0.5) * 0 + 0.5 * 0.4 = 0.2$				
		3	$w_{3,23} = (1-d) * w_{2,23} + d * I(f_3, f_1) = (1-0.5) * 0.45 + 0.5 * 0.8 = 0.625$				



V Example: Building Family 3

Step			Calculation				Stop
Family ( <i>j</i> )	Iteration ( <i>i</i> )	Weight ( <i>k</i> )	$w_{i,j,k}$	$\bar{w}_{i,j}$	$\Omega_{i,j}$	$\Lambda_{i,j}$	Y/N
3	1	1	$w_{1,3,1} = 0$	$\bar{w}_{1,3} = (0, 0, 0.9)$	$\Omega_{1,3} = \max(\bar{w}_{1,3}) = 0.9$	$\Lambda_{1,3} = \operatorname{argmax}(\bar{w}_{1,3}) = f_3$	No
		2	$w_{1,3,2} = 0$				
		3	$w_{1,3,3} = s_3 = 0.9$				
	2	1	$w_{2,3,1} = (1-d) * s_1 + d * I(f_1, f_3) = (1-0.5) * 0.6 + 0.5 * 0.8 = 0.7$	$\bar{w}_{2,3} = (0.7, 0.1, 0)$	$\Omega_{2,3} = \max(\bar{w}_{2,3}) = 0.7$	$\Lambda_{2,3} = \operatorname{argmax}(\bar{w}_{2,3}) = f_1$	No
		2	$w_{2,3,2} = (1-d) * s_2 + d * I(f_2, f_3) = (1-0.5) * 0.2 + 0.5 * 0 = 0.1$				
		3	$w_{2,3,3} = 0$				
	3	1	$w_{3,3,1} = 0$	$\bar{w}_{3,3} = (0, 0.25, 0.4)$	$\Omega_{3,3} = \max(\bar{w}_{3,3}) = 0.4$	$\Lambda_{3,3} = \operatorname{argmax}(\bar{w}_{3,3}) = f_3$	Yes $f_3$ already selected
		2	$w_{3,3,2} = (1-d) * w_{2,3,2} + d * I(f_2, f_1) = (1-0.5) * 0.1 + 0.5 * 0.4 = 0.25$				
		3	$w_{3,3,3} = (1-d) * w_{2,3,3} + d * I(f_3, f_1) = (1-0.5) * 0 + 0.5 * 0.8 = 0.4$				

VI Example: Calculating Weighted Scores

$$\Omega = \begin{bmatrix} 0.6 & 0.2 & 0.9 \\ 0.85 & 0.5 & 0.7 \\ - & 0.625 & - \end{bmatrix}$$

$$\Lambda = \begin{bmatrix} f_1 & f_2 & f_3 \\ f_3 & f_1 & f_1 \\ - & f_3 & - \end{bmatrix}$$

Feature	Original Score	Weighted Score
$f_1$	0.6	$0.6 + 0.5 + 0.7 = 1.8$
$f_2$	0.2	0.2
$f_3$	0.9	$0.9 + 0.85 + 0.625 = 2.375$

APPENDIX B  
FAMILY RANK R PACKAGE DOCUMENTATION

## I Package Description

The 'FamilyRank package contains tools for performing feature ranking via the family rank algorithm. Family rank grows families of features by selecting features that maximize a weighted score calculated from empirical feature scores and graphical knowledge. The final weighted score for a feature is determined by summing a feature's family-weighted scores across all families in which the feature appears.

## II Functions

### **createCase**

#### **Description**

Numerical feature simulation for positive samples. Called by createData.

#### **Usage**

```
createCase(subtype, upper.mean, lower.mean, upper.sd, lower.sd, n.features,  
subtype1.feats = 1:5, subtype2.feats = 6:10, subtype3.feats = 11:15)
```

#### **Arguments**

subtype	Numeric number indicating which subtype to simulate. Currently supports three subtype: 1, 2 or 3.
upper.mean	The mean of the upper component of the bimodal Gaussian distribution from which features are simulated.
lower.mean	The mean of the lower component of the bimodal Gaussian distribution from which features are simulated.
upper.sd	The standard deviation of the upper component of the bimodal Gaussian distribution from which features are simulated.
lower.sd	The standard deviation of the lower component of the bimodal Gaussian distribution from which features are simulated.
n.features	Number of features to simulate.
subtype1.feats	Numeric vector representing the indices of features that define subtype 1.
subtype2.feats	Numeric vector representing the indices of features that define subtype 2.
subtype3.feats	Numeric vector representing the indices of features that define subtype 3.

#### **Details**

Simulations support 3 subtypes, each defined by 5 different features.

Subtype 1 is defined as having the first 3 subtype1.feats and at least one of the next

2 subtype1.feats simulated from the upper component of the bimodal Gaussian distribution.

Subtype 2 is defined as having all 5 subtype2.feats simulated from the upper component.

Subtype 3 is defined as having the first 4 subtype3.feats simulated from the upper component and the last subtype3.feats simulated from the lower component.

**Value**

Returns a vector of simulated features

**Note**

createCase is not meant to be called alone. It is designed as a helper function for createData.

**Examples**

# Toy Example

```
case <- createCase(subtype = 1, upper.mean = 13, lower.mean = 5,  
upper.sd = 1, lower.sd = 1, n.features = 20,  
  subtype1.feats = 1:5,  
  subtype2.feats = 6:10,  
  subtype3.feats = 11:15)
```

<b>createControl</b>	
<b>Description</b>	
Numerical feature simulation for negative samples. Called by createData.	
<b>Usage</b>	
createControl(upper.mean, lower.mean, upper.sd, lower.sd, n.features, subtype1.feats = 1:5, subtype2.feats = 6:10, subtype3.feats = 11:15)	
<b>Arguments</b>	
upper.mean	The mean of the upper component of the bimodal Gaussian distribution from which features are simulated.
lower.mean	The mean of the lower component of the bimodal Gaussian distribution from which features are simulated.
upper.sd	The standard deviation of the upper component of the bimodal Gaussian distribution from which features are simulated.
lower.sd	The standard deviation of the lower component of the bimodal Gaussian distribution from which features are simulated.
n.features	Number of features to simulate.
subtype1.feats	Numeric vector representing the indices of features that define subtype 1.
subtype2.feats	Numeric vector representing the indices of features that define subtype 2.
subtype3.feats	Numeric vector representing the indices of features that define subtype 3.
<b>Details</b>	
Simulates data such that none of the 3 subtypes defined in createCase are represented.	
To ensure subtype 1 is not represented, at least one of the first three subtype1.feats and/or both of the next 2 subtype1.feats are simulated from the lower component of the Gaussian distribution.	
To ensure subtype 2 is not represented, at least one of the five subtype2.feats is simulated from the lower component.	
To ensure subtype 3 is not represented, at least one of the first 4 subtype3.feats is simulated from the lower component and/or the last subtype3.feats is simulated from the upper component.	
<b>Value</b>	
Returns a vector of simulated features	
<b>Note</b>	
createControl is not meant to be called alone. It is designed as a helper function for createData.	
<b>Examples</b>	
# Toy Example	

```
control <- createControl(upper.mean = 13, lower.mean = 5,  
upper.sd = 1, lower.sd = 1, n.features = 20,  
  subtype1.feats = 1:5,  
  subtype2.feats = 6:10,  
  subtype3.feats = 11:15)
```

## **createData**

### **Description**

Simulate data sets meant to emulate gene expression data in oncology.

### **Usage**

```
createData(n.case, n.control, mean.upper = 13, mean.lower = 5, sd.upper = 1, sd.lower = 1,  
n.features = 10000, subtype1.feats = 1:5, subtype2.feats = 6:10, subtype3.feats = 11:15)
```

### **Arguments**

n.case	Number of cases to simulate.
n.control	Number of controls to simulate
mean.upper	Mean of upper component of bimodal Gaussian distribution from which features are simulated.
mean.lower	Mean of lower component of bimodal Gaussian distribution from which features are simulated.
sd.upper	Standard deviation of upper component of bimodal Gaussian distribution from which features are simulated.
sd.lower	Standard deviation of lower component of bimodal Gaussian distribution from which features are simulated.
n.features	Number of features to simulate
subtype1.feats	Index of features used to define subtype 1.
subtype2.feats	Index of features used to define subtype 2.
subtype3.feats	Index of features used to define subtype 3.

### **Details**

Simulates case/control data as described in createCase and createControl, and graphical domain knowledge as described in createGraph.

### **Value**

Returns a named list with a simulated feature matrix (x), simulated binary response vector (y), vector of subtype labels (subtype), and simulated domain knowledge graph (graph).

### **Examples**

```
## Toy Example  
# Simulate data set  
# 10 samples  
# 20 features  
# Features 1 through 15 perfectly define response  
# All other features are random noise.  
data <- createData(n.case = 5, n.control = 5, mean.upper=13, mean.lower=5,  
                  sd.upper=1, sd.lower=1, n.features = 20,  
                  subtype1.feats = 1:5, subtype2.feats = 6:10,
```



```
        subtype3.feats = 11:15)  
x <- data$x  
y <- data$y  
graph <- data$graph
```

## **createGraph**

### **Description**

Simulate domain knowledge graph.

### **Usage**

```
createGraph(subtype1.feats = 1:5, subtype2.feats = 6:10, subtype3.feats = 11:15,  
n.interactions = 1e+06, n.features = 10000)
```

### **Arguments**

subtype1.feats    Index of features used to define subtype 1.  
subtype2.feats    Index of features used to define subtype 2.  
subtype3.feats    Index of features used to define subtype 3.  
n.interactions    Number of pairwise interactions to simulate.  
n.features        Number of features to simulate

### **Value**

Returns a data frame representation of a graph. The first two columns represent graph nodes and the third column represents the edge weights between nodes.

All pairwise combinations of subtype1.feats have an edge weight of 1.

All pairwise combinations of subtype2.feats have an edge weight of 1.

All pairwise combinations of subtype3.feats have an edge weight of 1.

All other pairwise combinations have an edge weight uniformly distributed between 0 and 1.

### **Examples**

```
# Toy Example
```

```
graph <- createGraph(subtype1.feats = 1:5, subtype2.feats = 6:10, subtype3.feats = 11:15,  
n.interactions = 100, n.features = 20)
```

## familyRank

### Description

Ranks features by incorporating graphical knowledge to weight empirical feature scores. This is the main function of the FamilyRank package.

### Usage

```
familyRank(scores, graph, d = 0.5, n.rank = min(length(scores), 1000), n.families = min(n.rank, 1000), tol = 0.001)
```

### Arguments

scores	A numeric vector of empirical feature scores. Higher scores should indicate a more predictive feature.
graph	A matrix or data frame representation of a graph object.
d	Damping factor
n.rank	Number of features to rank.
n.families	Number of families to grow.
tol	Tolerance

### Details

The scores vector should be generated using an existing statistical method. Higher scores should correspond to more predictive features. It is up to the user to adjust accordingly. For example, if the user wishes to use p-values as the empirical score, the user should first adjust the p-values, perhaps by subtracting all p-values from 1, so that a higher value corresponds to a more predictive feature.

The graph must be supplied in matrix form, where the first two columns represent graph nodes and the third column represents the edge weights between nodes. The graph nodes must be represented by the index of the feature that corresponds with the index in the score vector. For example, a node corresponding to the first value of the score vector should be indicated by a 1 in the graph object, the second by a 2, etc. It is not necessary that every feature in the score vector appear in the graph. Missing pairwise interactions will be considered to have interaction scores of 0.

The damping factor,  $d$ , represents the percentage of weight given to the interaction scores. The damping factor must be between 0 and 1. Higher values give more weight to the interaction score while lower values give more weight to the empirical score.

The value for  $n.rank$  must be less than or equal to the number of scored features. The algorithm will include only the top  $n.rank$  features in the ranking process (e.g. the  $n.rank$  features with the highest values in the score vector will be used to grow families). Higher values of  $n.rank$  require longer compute times.

The value for  $n.families$  must be less than or equal to the value of  $n.rank$ . This is the number of families the algorithm will grow. If  $n.families$  is less than  $n.rank$ , the algorithm will initiate families using the  $n.families$  highest scoring features. Higher values of  $n.families$  require longer

compute times.

The tolerance variable, `tol`, tells the algorithm when to stop growing a family. Features are added to families until the weighted score is less than the tolerance level, or until all features have been added.

#### Value

Returns a vector of the weighted feature scores.

#### Examples

```
# Toy Example
```

```
scores <- c(.6, .2, .9)
```

```
graph <- cbind(c(1,1), c(2,3), c(.4, .8))
```

```
familyRank(scores = scores, graph = graph, d = .5)
```

```
# Simulate data set
```

```
# 100 samples
```

```
# 1000 features
```

```
# Features 1 through 15 perfectly define response
```

```
# All other features are random noise
```

```
simulatedData <- createData(n.case = 50, n.control = 50, mean.upper=13, mean.lower=5,  
                           sd.upper=1, sd.lower=1, n.features = 10000,  
                           subtype1.feats = 1:5, subtype2.feats = 6:10,  
                           subtype3.feats = 11:15)
```

```
x <- simulatedData$x
```

```
y <- simulatedData$y
```

```
graph <- simulatedData$graph
```

```
# Score simulated features using absolute difference in group means
```

```
scores <- apply(x, 2, function(col){
```

```
  spl <- split(col, y)
```

```
  group.means <- unlist(lapply(spl, mean))
```

```
  score <- abs(diff(group.means))
```

```
  names(score) <- NULL
```

```
  return(score)
```

```
})
```

```
# Display top 15 features using empirical score
```

```
order(scores, decreasing = TRUE)[1:15]
```

```
# Rank scores using familyRank
scores.fr <- familyRank(scores = scores, graph = graph, d = .5)
# Display top 15 features using empirical scores with Family Rank
order(scores.fr, decreasing = TRUE)[1:15]
```

## grow

### Description

Call to the C++ function that grows the families.

### Usage

```
grow(n, f, d, graph, scores, feat_mat, score_mat, tol, weight_mat, selected)
```

### Arguments

n	Number of features to rank.
f	Number of families to grow.
d	Damping factor
graph	A matrix or data frame representation of a graph object.
scores	A numeric vector of empirical feature scores.
feat_mat	Matrix to store selected features.
score_mat	Matrix to store weighted scores of selected features.
tol	Tolerance
weight_mat	A matrix to store the cumulative weighted scores of selected futures across all families.
selected	Vector indicating whether a feature has been selected yet.

### Details

This is the workhorse function for the Family Rank algorithm.

### Value

Returns a matrix with  $1+2xn.families$  columns and  $n.rank$  rows. The first column is the cumulative feature score for each of the ranked features  $1:n.rank$ . The row number corresponds to the re-indexed feature index. The next  $n.families$  columns contain the indices of selected features for each iteration of feature selection. The last  $n.families$  columns contain the weighted scores of selected features for each iteration.

### Examples

```
# Toy Example
```

```
scores <- c(.6, .2, .9)
```

```
graph <- cbind(c(1,1), c(2,3), c(.4, .8))
```

```
# initialize matrices
```

```
n <- n.families <- length(scores)
```

```
feat.mat <- score.mat <- matrix(0, nrow = n, ncol = n.families)
```

```
feat.mat[1,] <- order(scores, decreasing = TRUE)
```

```
score.mat[1,] <- sort(scores, decreasing = TRUE)
```

```
# Grow families
mats <- grow(n = n, f = n.families, d = 0.5, graph = as.matrix(graph),
            scores = scores,
            feat_mat = feat.mat, score_mat = score.mat, tol = 0,
            weight_mat = as.matrix(scores), selected = rep(1, n))
# Selected Feature Matrix
## columns represent families
## rows represent iterations
## values indicate indices of selected features
feat.mat <- mats[, 2:(n.families+1)]
feat.mat
# Corresponding Score Matrix
## columns represent families
## rows represent iterations
## values indicate max weighted score of selected features
score.mat <- mats[, (n.families+2):(1+2*n.families)]
score.mat
```

## indexFeats

### Description

Re-index features based on number to rank. Called by familyRank.

### Usage

```
indexFeats(scores, graph, n.rank = NULL)
```

### Arguments

- scores A numeric vector of empirical feature scores.
- graph A matrix or data frame representation of a graph object.
- n.rank Number of features to rank.

### Details

This function is used to re-index features for the Family Rank algorithm. The function takes in the scores for all features, and returns scores for the top n.rank features. It also takes in the full domain knowledge graph and returns the subgraph that only includes interactions between the top n.rank features. Finally, it re-indexes the top features in both the score vector and domain knowledge graph to 1:n.rank.

### Value

Returns a named list with re-indexed domain knowledge graph (graph.w), re-indexed scores (score.w), a mapping between original and new indices (loc.map), and the number of features to rank (n.rank).

### Note

indexFeats is not meant to be called alone. It is designed as a helper function for familyRank.



## **rbinorm**

### **Description**

Simulates random data from a bimodal Gaussian distribution.

### **Usage**

```
rbinorm(n, mean1, mean2, sd1, sd2, prop)
```

### **Arguments**

n            Number of observations to simulate  
mean1       Mean of mode 1  
mean2       Mean of mode 2  
sd1          Standard deviation of mode 1  
sd2          Standard deviation of mode 2  
prop         Probability of being in mode 1. 1 - prop is the probability of being in mode 2.

### **Details**

This function is modeled off of the rnorm function.

### **Value**

Generates random deviates

### **Examples**

```
## Generate 100 samples from a two component Gaussian curve  
samples <- rbinorm(n=100, mean1=10, mean2=20, sd1=1, sd2=2, prop=.5)
```

```
## Plot distribution of simulated data  
plot(density(samples))
```

APPENDIX C  
BIMIXT R PACKAGE DOCUMENTATION

## I Package Description

The 'bimixt' package contains tools for estimating non-Gaussian mixture models of case-control data. The four types of models supported are binormal, two component constrained, two component unconstrained, and four component. The most general model is the four-component model, under which both cases and controls are distributed according to a mixture of two unimodal distributions. In the four-component model, the two component distributions of the control mixture may be distinct from the two components of the case mixture distribution. In the two-component unconstrained model, the components of the control and case mixtures are the same; however, the mixture probabilities may differ for cases and controls. In the two-component constrained model, all controls are distributed according to one of the two components while cases follow a mixture distribution of the two components. In the binormal model, cases and controls are distributed according to distinct unimodal distributions. These models assume that Box-Cox transformed case and control data with a common lambda parameter are distributed according to Gaussian mixture distributions. Model parameters are estimated using the expectation-maximization (EM) algorithm. Likelihood ratio test comparison of nested models can be performed using the `lr.test` function. AUC and PAUC values can be computed for the model-based and empirical ROC curves using the `auc` and `pauc` functions, respectively. The model-based and empirical ROC curves can be graphed using the `roc.plot` function. Finally, the model-based density estimates can be visualized by plotting a model object created with the `bimixt.model` function.

## II Functions

## **bc.binorm**

### **Description**

Implementation of binormal model. The binormal model estimates a single unimodal component for the cases and a single unimodal component for the controls.

### **Usage**

```
bc.binorm(case, control, lambda.bounds = c(-5, 5))
```

### **Arguments**

case	a numeric vector of case values
control	a numeric vector of control values
lambda.bounds	numeric vector of bounds: c(upper bound, lower bound). Specifies the range for optim to search for the optimization of lambda. Default: c(-5,5).

### **Value**

lambda	Box-Cox transformation parameter
type	model type ("binorm")
mu.cases	mean of the Box-Cox transformed case component
sig.cases	standard deviation of the Box-Cox transformed case component
pi.cases	proportion of cases in each case component (always equal to 1 for binorm since all cases are forced into one component)
mu.controls	mean value of the Box-Cox transformed control component
sig.controls	standard deviation of the Box-Cox transformed control component
pi.controls	proportion of controls in each control component (always equal to 1 for binorm since all controls are forced into one component)
max.loglike	the maximum log likelihood value for the model
case	original case values
control	original control values
mu.cases.unt	an estimate of the untransformed mean of the case component. Based on Monte Carlo simulations. Values will differ by computer seed.
sig.cases.unt	an estimate of the untransformed standard deviation of the case component. Based on Monte Carlo simulations. Values will differ by computer seed.

mu.controls.unt an estimate of the untransformed mean of the control component. Based on Monte Carlo simulations. Values will differ by computer seed.

sig.controls.unt an estimate of the untransformed standard deviation of the control component. Based on Monte Carlo simulations. Values will differ by computer seed.

## **bc.fourcomp**

### **Description**

Implementation of four component model. The four-component model estimates an upper and lower component for the cases and an upper and lower component for the controls.

### **Usage**

```
bc.fourcomp(x.cases, x.controls, lambda.bounds = c(-5, 5), start.vals.cases=NULL,  
start.vals.controls=NULL)
```

### **Arguments**

x.cases	a numeric vector of case values
x.controls	a numeric vector of control values
lambda.bounds	numeric vector of bounds: c(upper bound, lower bound). Specifies the range for optim to search for the optimization of lambda. Default: c(-5,5).
start.vals.cases	starting values for the EM algorithm for the cases. If NA, the starting values are estimated from the data.
start.vals.controls	starting values for the EM algorithm for the controls. If NA, the starting values are estimated from the data.

### **Value**

lambda	Box-Cox transformation parameter
type	model type ( "4c")
mu.cases	means of the Box-Cox transformed case components
sig.cases	standard deviations of the Box-Cox transformed case components
pi.cases	proportion of cases in each case component
max.loglike.cases	the maximum log likelihood value for the fit of the cases
mu.controls	means of the Box-Cox transformed control components
sig.controls	standard deviations of the Box-Cox transformed control components
pi.controls	proportion of controls in each control component
max.loglike.controls	the maximum log likelihood value for the fit of the controls
max.loglike	the maximum log likelihood value for the model

mu.cases.unt	an estimate of the untransformed means of the case components. Based on Monte Carlo simulations. Values will differ by computer seed.
sig.cases.unt	an estimate of the untransformed standard deviations of the case components. Based on Monte Carlo simulations. Values will differ by computer seed.
mu.controls.unt	an estimate of the untransformed means of the control components. Based on Monte Carlo simulations. Values will differ by computer seed.
sig.controls.unt	an estimate of the untransformed standard deviations of the control components. Based on Monte Carlo simulations. Values will differ by computer seed.
case	original case values
control	original control values
time	running time for the model fit

## **bc.fourcomp**

### **Description**

Implementation of four component model. The four-component model estimates an upper and lower component for the cases and an upper and lower component for the controls.

### **Usage**

```
bc.fourcomp(x.cases, x.controls, lambda.bounds = c(-5, 5), start.vals.cases=NULL,  
start.vals.controls=NULL)
```

### **Arguments**

x.cases	a numeric vector of case values
x.controls	a numeric vector of control values
lambda.bounds	numeric vector of bounds: c(upper bound, lower bound). Specifies the range for optim to search for the optimization of lambda. Default: c(-5,5).
start.vals.cases	starting values for the EM algorithm for the cases. If NA, the starting values are estimated from the data.
start.vals.controls	starting values for the EM algorithm for the controls. If NA, the starting values are estimated from the data.

### **Value**

lambda	Box-Cox transformation parameter
type	model type ( "4c")
mu.cases	means of the Box-Cox transformed case components
sig.cases	standard deviations of the Box-Cox transformed case components
pi.cases	proportion of cases in each case component
max.loglike.cases	the maximum log likelihood value for the fit of the cases
mu.controls	means of the Box-Cox transformed control components
sig.controls	standard deviations of the Box-Cox transformed control components
pi.controls	proportion of controls in each control component
max.loglike.controls	the maximum log likelihood value for the fit of the controls
max.loglike	the maximum log likelihood value for the model



mu.cases.unt	an estimate of the untransformed means of the case components. Based on Monte Carlo simulations. Values will differ by computer seed.
sig.cases.unt	an estimate of the untransformed standard deviations of the case components. Based on Monte Carlo simulations. Values will differ by computer seed.
mu.controls.unt	an estimate of the untransformed means of the control components. Based on Monte Carlo simulations. Values will differ by computer seed.
sig.controls.unt	an estimate of the untransformed standard deviations of the control components. Based on Monte Carlo simulations. Values will differ by computer seed.
case	original case values
control	original control values
time	running time for the model fit

## **bc.twocomp**

### **Description**

Implementation of two component models. In the two-component unconstrained model, the components of the control and case mixtures are the same; however, the mixture probabilities may differ for cases and controls. In the two-component constrained model, all controls are distributed according to one of the two components while cases follow a mixture distribution of the two components.

### **Usage**

```
bc.twocomp(x.cases, x.controls, constrained = T, lambda.bounds = c(-5, 5), control.comp = 1, start.vals=NULL)
```

### **Arguments**

x.cases	a numeric vector of case values
x.controls	a numeric vector of control values
constrained	Boolean indicating whether the two-component constrained model should be used (default T) or the two component unconstrained model should be used (F)
lambda.bounds	numeric vector of bounds: c(upper bound, lower bound). Specifies the range for optim to search for the optimization of lambda. Default: c(-5,5).
control.comp	indicator of which component contains the controls (1 or 2)
start.vals	starting values for the EM algorithm. If NA, the starting values are estimated from the data.

### **Value**

lambda	Box-Cox transformation parameter
type	model type ( "2cc" or "2cu")
mu.cases	means of the Box-Cox transformed case components
sig.cases	standard deviations of the Box-Cox transformed case components
pi.cases	proportion of cases in each case component
mu.controls	means of the Box-Cox transformed control components
sig.controls	standard deviations of the Box-Cox transformed control components
pi.controls	proportion of controls in each control component (always equal to 1 for 2cc since all controls are forced into one component)

max.loglike	the maximum log likelihood value for the model
mu.cases.unt	an estimate of the untransformed means of the case components. Based on Monte Carlo simulations. Values will differ by computer seed.
sig.cases.unt	an estimate of the untransformed standard deviations of the case components. Based on Monte Carlo simulations. Values will differ by computer seed.
mu.controls.unt	an estimate of the untransformed means of the control components. Based on Monte Carlo simulations. Values will differ by computer seed.
sig.controls.unt	an estimate of the untransformed standard deviations of the control components. Based on Monte Carlo simulations. Values will differ by computer seed.
case	original case values
control	original control values
time	running time for the model fit

## **bimixt.model**

### **Description**

Estimates mixture model components based on model type.

### **Usage**

```
bimixt.model(case, control, type = "binorm", start.vals=NULL)
```

### **Arguments**

- |            |  |
|------------|--|
| case       | a numeric vector of case values. NA's will be omitted.   |
| control    | a numeric vector of control values. NA's will be omitted.  |
| type       | a string specifying the mixture model to be used to fit the data. Valid types are "binorm", "2cc", "2cu", or "4c". These correspond to binormal, two component constrained, two component unconstrained, and four component models respectively. Defaults to "binorm". |
| start.vals | an (optional) list of starting values for the EM algorithm used in the "2cc", "2cu", and "4c" models. If not specified by the user, starting values are estimated from the data using kmeans clustering. The format of the lists are described in the details section. |

### **Details**

Starting values for the EM algorithm can be provided by the user. The starting values must be given as lists. Each element in the list is a named numeric vector of length 2 containing starting estimates for the model parameters. Names must match the names given below exactly (See examples section for "4c" model example).

For "2cc" start.vals is a list of 3 named vectors:

- mu Starting estimates for component means
- sig Starting estimates for component standard deviations
- pi Starting estimates for component proportions. Must sum to 1.

For "2cu", start.vals is a list of length 4:

- mu Starting estimates for component means.
- sig Starting estimates for component standard deviations.
- pi.cs Starting estimates for case component proportions. Must sum to 1.
- pi.ctrl Starting estimates for control component proportions. Must sum to 1.

For "4c", start.vals is a list of length 6:

mu.cs Starting estimates for case component means.

mu.ctrl Starting estimates for control component means.

sig.cs Starting estimates for case component standard deviations.

sig.ctrl Starting estimates for control component standard deviations.

pi.cs Starting estimates for component proportions for cases. Must sum to 1.

pi.ctrl Starting estimates for component proportions for controls. Must sum to 1.

### Value

Returns an object of type model with parameters specified by bc.binorm, bc.twocomp, or bc.fourcomp.

### Examples

```
case=rmix(50,10,1.2,15,1,.7)
```

```
control=rmix(50,10,1.2,15,1,.95)
```

```
model1=bimixt.model(case=case, control=control, type="4c", start.vals=list(mu.cs=c(10,15),
```

```
mu.ctrl=c(10,15),sig.cs=c(1.2,1),sig.ctrl=c(1.2,1),pi.cs=c(.7,.3),pi.ctrl=c(.95,.05)))
```

```
model2=bimixt.model(case=case, control=control, type="2cu")
```

```
model3=bimixt.model(case=case, control=control, type="2cc")
```

```
model4=bimixt.model(case=case, control=control, type="binorm")
```

## **boxcox**

### **Description**

Implementation of the Box-Cox normalization transformation method. Called internally in `bc.twocomp` and `bc.fourcomp`.

### **Usage**

```
boxcox(x, lambda)
```

### **Arguments**

`x` a numeric vector or scalar

`lambda` Box-Cox transformation variable

### **Value**

A vector or scalar of the transformed values of `x`.

### **References**

Box, George EP, and David R. Cox. "An analysis of transformations." *Journal of the Royal Statistical Society. Series B (Methodological)* (1964): 211-252.

**boxcox.deriv****Description**

Derivative of the Box-Cox transformation function.

**Usage**

```
boxcox.deriv(x, lambda)
```

**Arguments**

x            a numeric vector or scalar

lambda    Box-Cox transformation variable.

**Value**

A vector or scalar of the derivative of the Box-Cox function evaluated at x.

**References**

Box, George EP, and David R. Cox. "An analysis of transformations." *Journal of the Royal Statistical Society. Series B (Methodological)* (1964): 211-252.

**boxcox.inv****Description**

Inverse of the Box-Cox transformation. Called internally in `bc.twocomp` and `bc.fourcomp`.

**Usage**

```
boxcox.inv(y, lambda)
```

**Arguments**

`y` a numeric vector or scalar

`lambda` Box-Cox transformation variable.

**Value**

A vector or scalar of the untransformed values of `x`.

**References**

Box, George EP, and David R. Cox. "An analysis of transformations." *Journal of the Royal Statistical Society. Series B (Methodological)* (1964): 211-252.



## **boxcox.inv.density**

### **Description**

A variable transformation that gives the probability density function (PDF) of the inverse Box-Cox transformation of a normal random variable. Called internally in plot.model.

### **Usage**

```
boxcox.inv.density(y, lambda, mu, sig)
```

### **Arguments**

y            a numeric vector or scalar

lambda      the transformation parameter

mu           the mean of the transformed component

sig          the standard deviation of the transformed component

### **Value**

A vector or scalar of the untransformed x values.

### **References**

Box, George EP, and David R. Cox. "An analysis of transformations." *Journal of the Royal Statistical Society. Series B (Methodological)* (1964): 211-252.

## em.twocomp.m1

### Description

Expectation maximization (EM) algorithm for estimating two-component Gaussian mixtures in which all controls are constrained to one component and the cases follow a mixture of the two components (two component constrained model). This is used as an internal method and is called from bc.twocomp.

### Usage

```
em.twocomp.m1(x.all, case.indicator, max.iters = 1000, errtol = 1e-09, control.comp = 1, start.vals=NULL)
```

### Arguments

x.all	vector of cases and controls
case.indicator	a vector of equal length to x.all with 1's in the case positions and 0's in the control positions
max.iters	the maximum number of iterations to run
errtol	Error tolerance level. Approximates convergence of the maximum log likelihood value.
control.comp	indicator of which component contains the controls (1 or 2)
start.vals	starting values for the EM algorithm. If NA, the starting values are estimated from the data.

### Value

max.loglike	the maximum log likelihood value for the algorithm
mu	estimated means for each component
sig	estimated standard deviations for each component
pi	estimated proportion of cases in each component
n.iters	the number of iterations the algorithm took to converge
control.comp	indicator of which component contains the controls (1 or 2)

### References

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the royal statistical society. Series B (methodological)* (1977): 1-38.

## em.twocomp.m2

### Description

Expectation maximization (EM) algorithm for estimating two-component Gaussian mixture models. This is used as an internal method and is called twice from bc.fourcomp: once for the cases and once for the controls (four component model).

### Usage

```
em.twocomp.m2(x.all, max.iters = 1000, errtol = 1e-09, start.vals=NULL)
```

### Arguments

- |            |   |
|------------|---|
| x.all      | vector of data  |
| max.iters  | the maximum number of iterations to run   |
| errtol     | Error tolerance level. Approximates convergence of the maximum log likelihood value.          |
| start.vals | starting values for the EM algorithm. If NA, the starting values are estimated from the data. |

### Value

- |             |   |
|-------------|---|
| max.loglike | the maximum log likelihood value for the algorithm      |
| mu          | estimated means for each component                      |
| sig         | estimated standard deviation for each component         |
| pi          | estimated proportion of data in each component          |
| n.iters     | the number of iterations the algorithm took to converge |

### References

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the royal statistical society. Series B (methodological)* (1977): 1-38.

## em.twocomp.m3

### Description

Expectation maximization (EM) algorithm for estimating two-component Gaussian mixtures with different mixture proportions for cases and controls (two component unconstrained model). This is used as an internal method and is called from bc.twocomp.

### Usage

```
em.twocomp.m3(x.all, case.indicator, max.iters = 1000, errtol = 1e-09, control.comp = 1, start.vals=NULL)
```

### Arguments

x.all	vector of cases and controls
case.indicator	a vector of equal length to x.all with 1's in the case positions and 0's in the control positions
max.iters	the maximum number of iterations to run
errtol	Error tolerance level. Approximates convergence of the maximum log likelihood value.
control.comp	indicator of which component contains the controls (1 or 2)
start.vals	starting values for the EM algorithm. If NA, the starting values are estimated from the data.

### Value

max.loglike	the maximum log likelihood value for the algorithm
mu	estimated means for each component
sig	estimated standard deviations for each component
pi.cs	estimated proportion of cases in each component
pi.ctrl	estimated proportion of controls in each component
n.iters	the number of iterations the algorithm took to converge
control.comp	indicator of which component contains the controls (1 or 2)

### References

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the royal statistical society. Series B (methodological)* (1977): 1-38.

## Lambda

### Description

An accessor function. Retrieves the transformation parameter, lambda, of a model object.

### Usage

```
lambda(model)
```

### Arguments

`model` an object of type `model` from `bimixt.model`

### Value

The numeric value for the Box-Cox transformation parameter, lambda.

### Examples

```
case=rmix(50,10,1.2,15,1,.7)
```

```
control=rmix(50,10,1.2,15,1,.95)
```

```
model=bimixt.model(case=case,control=control, type="4c")
```

```
lambda(model)
```

## **lr.test**

### **Description**

Computes the likelihood ratio test to compare two bimixt models.

### **Usage**

```
lr.test(model1, model2)
```

### **Arguments**

model1 an object of type model from bimixt.model.

model2 an object of type model from bimixt.model

### **Details**

The model fits for model1 and model2 will be compared using the likelihood ratio test. Models must have been fit on the same data sets.

### **Value**

Returns a p-value indicating the significance of the likelihood ratio test.

### **Examples**

```
case=rmix(50,10,1.2,15,1,.7)
```

```
control=rmix(50,10,1.2,15,1,.95)
```

```
model1=bimixt.model(case=case,control=control, type="4c")
```

```
model2=bimixt.model(case=case,control=control, type="binorm")
```

```
lr.test(model1, model2)
```

## **Maxll**

### **Description**

An accessor function. Retrieves the maximum log likelihood value of a model object.

### **Usage**

```
maxll(model)
```

### **Arguments**

`model` an object of type `model` from `bimixt.model`

### **Value**

The numeric value for the maximum log likelihood value for the model.

### **Examples**

```
case=rmix(50,10,1.2,15,1,.7)
```

```
control=rmix(50,10,1.2,15,1,.95)
```

```
model=bimixt.model(case=case,control=control, type="4c")
```

```
maxll(model)
```

## Mn

### Description

An accessor function. Retrieves the component means of a model object.

### Usage

```
mn(model, transformed = F)
```

### Arguments

model	an object of type model from bimixt.model
transformed	A Boolean indicating whether to return the mean values on the transformed scale (TRUE) or the original scale (FALSE default). The transformed means are estimates of the Gaussian component means. The original scale means are Monte Carlo estimates of the mean of the distribution of the inverse Box-Cox function applied to the estimated Gaussian component distribution.

### Value

cases	A vector (or scalar) of numeric values for the mean of each case component in the model.
controls	A vector (or scalar) of numeric values for the mean of each control component in the model.

### Examples

```
case=rmix(50,10,1.2,15,1,.7)
control=rmix(50,10,1.2,15,1,.95)
model=bimixt.model(case=case,control=control, type="4c")
mn(model)
```



## plot.model

### Description

Plot method for a mixture model object.

### Usage

```
## S3 method for class 'model'
```

```
plot(x, histogram = T, breaks = "Sturges", main = model$type, cols = c("#008ED6", "#990033"),  
ylab = "Density", xlab = "", ...)
```

### Arguments

x	an object of type model from bimixt.model
histogram	a Boolean indicating whether to plot a histogram of the original data (default = true). Histogram is plotted using the hist function.
breaks	the types of breaks to be used in hist
main	a character string to be used as the title of the plot
cols	a vector of length 2 specifying the colors of the components c(color of control component, color of case component)
ylab	y label of the plot
xlab	x label of the plot
...	Not used.

### Value

Plots a model object.

### Examples

```
case=rmix(50,10,1.2,15,1,.7)
```

```
control=rmix(50,10,1.2,15,1,.95)
```

```
model=bimixt.model(case=case,control=control, type="4c")
```

```
plot(model)
```

## **print.model**

### **Description**

Print method for a mixture model object.

### **Usage**

```
## S3 method for class 'model'
```

```
print(x, ...)
```

### **Arguments**

x an object of type model from bimixt.model

... Not used.

### **Value**

Values used in fitting a mixture model object.

### **Examples**

```
case=rmix(50,10,1.2,15,1,.7)
```

```
control=rmix(50,10,1.2,15,1,.95)
```

```
model=bimixt.model(case=case,control=control, type="4c")
```

```
print(model)
```

## Prop

### Description

An accessor function. Retrieves the case component proportions and control component proportions of a model object.

### Usage

```
prop(model)
```

### Arguments

`model` an object of type `model` from `bimixt.model`

### Value

`cases` A vector (or scalar) of numeric values for the proportion of cases in each case component of the model.

`controls` A vector (or scalar) of numeric values for the proportion of controls in each control component of the model.

### Examples

```
case=rmix(50,10,1.2,15,1,.7)
```

```
control=rmix(50,10,1.2,15,1,.95)
```

```
model=bimixt.model(case=case,control=control, type="4c")
```

```
prop(model)
```

## Rmix

### Description

Creates a random sample from a normal mixture distribution with two components.

### Usage

```
rmix(n, mu1, s1, mu2, s2, p1)
```

### Arguments

- n size of random sample
- mu1 mean of first component
- s1 standard deviation of first component
- mu2 mean of second component
- s2 standard deviation of second component
- p1 proportion of values in the first component

### Value

A vector of n numeric values from a sample mixture distribution.

### Examples

```
rmix(30,5,1,10,1.2,.95)
```

## ROCauc

### Description

Finds the area under the ROC curve.

### Usage

```
ROCauc(model, direction = "auto")
```

### Arguments

**model** an object of type model from bimixt.model

**direction** same as roc: the direction in which to make the comparison. "auto" (default): automatically define in which group the median is higher and take the direction accordingly. ">": if the values for the control group are higher than the values of the case group (controls > t >= cases). "<": if the values for the control group are lower than the values of the case group (controls < t <= cases).

### Value

Returns the area under the curve (AUC) for the fitted and empirical receiver operator characteristic (ROC) curves. The empirical AUC value is calculated using the pROC package.

### References

Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frederique Lisacek, Jean-Charles Sanchez and Markus Miller (2011). "pROC: an open-source package for R and S+ to analyze and compare ROC curves". BMC Bioinformatics, 12, p. 77. DOI: 10.1186/1471-2105-12-77

### Examples

```
cases=rmix(50,10,1.2,20,1.3,.7)
controls=rmix(50,9,1.1,17,1.3,.95)
model=bimixt.model(cases,controls,"4c")
ROCauc(model)
```

## ROCcoords

### Description

Takes in a threshold, specificity, or sensitivity value and calculates the other two values.

### Usage

```
ROCcoords(model, direction = "auto", x, input)
```

### Arguments

- model** an object of type model from bimixt.model
- direction** same as roc: the direction in which to make the comparison. "auto" (default): automatically define in which group the median is higher and take the direction accordingly. ">": if the values for the control group are higher than the values of the case group (controls > t >= cases). "<": if the values for the control group are lower than the values of the case group (controls < t <= cases).
- x** The numeric value for the input. If input is "sensitivity" or "specificity" x must be between 0 and 1.
- input** A string that defines what the input type is. Valid inputs are "sensitivity", "specificity", or "threshold". These can be shortened to "sens", "spec", "thr" or "se", "sp", "t".

### Value

Returns a numeric vector with the values of threshold, specificity, and sensitivity.

### Examples

```
cases=rmix(50,10,1.2,20,1.3,.7)
controls=rmix(50,9,1.1,17,1.3,.95)
model=bimixt.model(cases,controls,"4c")
ROCcoords(model,x=.95,input="sens")
ROCcoords(model,x=.95,input="spec")
ROCcoords(model,x=9,input="thr")
```

## ROCpauc

### Description

Finds the partial area under the ROC curve.

### Usage

```
ROCpauc(model, spec.lower = 0.95, spec.upper = 1, direction = "auto")
```

### Arguments

model	an object of type model from bimixt.model
spec.lower	a value between 0 and 1 that serves as the lower bound of the specificity to be used in the PAUC calculation
spec.upper	a value between 0 and 1 that serves as the upper bound of the specificity to be used in the PAUC calculation
direction	same as roc: the direction in which to make the comparison. "auto" (default): automatically define in which group the median is higher and take the direction accordingly. ">": if the predictor values for the control group are higher than the values of the case group (controls > t >= cases). "<": if the predictor values for the control group are lower or equal than the values of the case group (controls < t <= cases).

### Value

Returns the partial area under the curve (pAUC) for the fitted and empirical receiver operator characteristic (ROC) curves between spec.lower and spec.upper. The empirical pAUC value is calculated using the pROC package.

### References

Xavier Robin, Natacha Turck, Alexandre Hainard, et al. (2011) "pROC: an open-source package for R and S+ to analyze and compare ROC curves". BMC Bioinformatics, 7, 77. DOI: 10.1186/1471-2105-12-77.

### Examples

```
cases=rmix(50,10,1.2,20,1.3,.7)
controls=rmix(50,9,1.1,17,1.3,.95)
model= bimixt.model(cases,controls,"4c")
ROCpauc(model, spec.lower = .85, spec.upper = 1)
```

## ROCplot

### Description

Creates a ROC plot.

### Usage

```
ROCplot(model, direction = "auto")
```

### Arguments

**model**      an object of type model from bimixt.model

**direction**    same as roc: same as pROC: the direction in which to make the comparison. "auto" (default): automatically define in which group the median is higher and take the direction accordingly. ">": if the values for the control group are higher than the values of the case group (controls > t >= cases). "<": if the values for the control group are lower than the values of the case group (controls < t <= cases).

### Value

Plots empirical and model-based estimates of the receiver operator characteristic (ROC) curve. The empirical plot comes from the pROC package.

### References

Xavier Robin, Natacha Turck, Alexandre Hainard, et al. (2011) "pROC: an open-source package for R and S+ to analyze and compare ROC curves". BMC Bioinformatics, 7, 77. DOI: 10.1186/1471-2105-12-77.

### Examples

```
cases=rmix(50,10,1.2,20,1.3,.7)
controls=rmix(50,9,1.1,17,1.3,.95)
model=bimixt.model(cases,controls,"4c")
ROCplot(model)
```



## **stdev**

### **Description**

An accessor function. Retrieves the component standard deviations of a model object.

### **Usage**

```
stdev(model, transformed = F)
```

### **Arguments**

model	an object of type model from bimixt.model
transformed	A Boolean indicating whether to return the standard deviation values on the transformed scale (TRUE) or the original scale (FALSE default). The transformed standard deviations are estimates of the Gaussian component standard deviations. The original scale standard deviations are Monte Carlo estimates of the standard deviation of the inverse Box-Cox of the estimated Gaussian component distribution.

### **Value**

cases	A vector (or scalar) of numeric values for the standard deviation of each case component in the model.
controls	A vector (or scalar) of numeric values for the standard deviation of each control component in the model.

### **Examples**

```
case=rmix(50,10,1.2,15,1,.7)
control=rmix(50,10,1.2,15,1,.95)
model=bimixt.model(case=case,control=control, type="4c")
stdev(model)
```

## summary.model

### Description

Summary method for a mixture model object.

### Usage

```
## S3 method for class 'model'
```

```
summary(object, ...)
```

### Arguments

`object` an object of type `model` from `bimixt.model`

`...` Not used.

### Value

Gives a table with the estimated means and standard deviations of the Gaussian components (following the Box-Cox transformation), the estimated means and standard deviations of the untransformed components (before transforming for normality), and the estimated case and control proportions for each component in the mixture model.

### Examples

```
case=rmix(50,10,1.2,15,1,.7)
```

```
control=rmix(50,10,1.2,15,1,.95)
```

```
model=bimixt.model(case=case,control=control, type="4c")
```

```
summary(model)
```

## **type**

### **Description**

An accessor function. Retrieves the model type of a model object.

### **Usage**

```
type(model)
```

### **Arguments**

`model` an object of type `model` from `bimixt.model`

### **Value**

Returns the type of the model, either "4c", "2cu", "2cc", or "binorm".

### **Examples**

```
case=rmix(50,10,1.2,15,1,.7)
```

```
control=rmix(50,10,1.2,15,1,.95)
```

```
model=bimixt.model(case=case,control=control, type="2cu")
```

```
type(model)
```