2016

An extension of penalized ordinal response models

Qing Zhou
Virginia Commonwealth University

An extension of penalized ordinal response models

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

by

Qing Zhou

Director: Kellie J. Archer, Ph.D., Professor, Department of Biostatistics

Department of Biostatistics

Virginia Commonwealth University

Richmond, Virginia, USA

April, 2016

Acknowledgements

First and foremost, I would like to express my special appreciation to my thesis advisor Dr. Kellie Archer for her support and guidance. She not only helped me in research but also inspired me to be an enthusiastic, hardworking, responsive and kindhearted person as she is. It has truly been a great privilege and pleasure having her as my mentor.

I would also like to thank my thesis committee: Dr. Sabo, Dr. Mukhopadhyay, Dr. Cook and Dr. Elmore for their time and effort to serve on my thesis committee and for their helpful comments and suggestions.

I would like to thank the professors of the Biostatistics Department for their outstanding teaching and help. I also owe my thanks to Dr. McClish, Dr. Sabo and Russell Boyle for making sure I met all deadlines, passed my qualifying and comprehensive exams, and graduated on time.

Finally, I thank my family, especially my husband for his continuous support and my mother for fiercely pushing me to go to graduate school.

Contents

List of Tables

List of Figures

Abstract

An extension of penalized ordinal response models

by Qing Zhou

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

Virginia Commonwealth University, 2015

Major Director: Kellie J. Archer, Ph.D., Professor, Department of Biostatistics

Ordinal responses are commonly seen in medical research. Many pathological evaluations and health status outcomes are reported on an ordinal scales. Some examples of ordinal outcomes include cancer stage (I, II, III and IV), or stage of liver disease (normal liver, chronic hepatitis, cirrhosis and end of stage liver disease or hepatocellular carcinoma (HCC)).

In recent years, there has been a demand to understanding the pathogenic association between ordinal clinical outcomes and molecular characteristics. Genomic charactersitics are often assayed using a high-dimensional platform where the number of interrogated sites (P) exceeds the number of samples (n). Unfortunately, traditional ordinal response models often do not perform well when the number of parameter (P) exceed the number of observations (n). A good solution to this problem is penalization, for example, least absolute shrinkage and selection operator (LASSO). Here, we extend a LASSO method, the generalized monotone incremental

forward stagewise algorithm (GMIFS) method, to ordinal response models. Specifically, this research details the extension of the GMIFS method to probit link ordinal response models and the stereotype logit model.

Moreover, motivated by the Bayesian LASSO proposed by Park and Casella (2008), we developed an ordinal response model that incorporates a penalty term so that both feature selection and outcome prediction are achievable. The ordinal response model we are focusing on is the cumulative logit model, and the performance will be compared with the frequentist LASSO cumulative logit model (GMIFS).

In addition to GMIFS and penalized Bayesian cumulative logit model, this research also addresses filtering, which is another dimension reduction method (different from penalization). We compare filtering, or univariate feature selection methods, with penalization methods using grouped survival data.

Chapter 1

Introduction

1.1 Ordinal outcomes and high-dimensional data

Ordinal responses are commonly used in medical research. Many pathological evaluations and health status outcomes are reported on an ordinal scale. Some examples of ordinal outcomes include cancer stage (stage I, II, III or IV), stage of liver disease (normal liver, chronic hepatitis, cirrhosis and end of stage liver disease or hepatocellular carcinoma (HCC)) or grouped survival outcomes (short, intermediate and long-term survival). Unlike nominal scales for categorical variables, ordinal variables have some unique features. For example, there is a clear ordering of the levels, but the absolute distances among these levels are unknown. For instance, stage II cancer is more severe than stage I; however, it is hard to quantify the difference between two levels by a numerical measure [Agresti, 2010]. Because of the distinct nature of an ordinal response, it is often recommended to use a traditional ordinal response

model for the analysis of ordinal data. Traditional ordinal response models include the cumulative link model which has various link functions (logit, probit or complementary loglog link), adjacent category model, and continuation ratio models. These models have many advantages over a multinomial regression model. They are usually more parsimonious, have simpler interpretations than multinominal models, because multinomial models have different sets of slope coefficients for the log-odds of each response while most ordinal response models assume proportional odds and therefore only have one set of slope coefficients regardless of response level. Ordinal response models also have greater power for detecting relevant trend effects of predictors [Agresti, 2010]. In section 1.3, some common ordinal models are briefly described.

In recent years, as genomic technologies have advanced and the number of genetic studies have increased, there has been a demand to understand the pathogenic association between ordinal clinical outcomes and molecular characteristics. For example, it may be of interest to identify how methylation of CpG sites or gene/protein expression values are predictive of an ordinal outcome. Unlike traditional clinical variables, genomic characteristics are often assayed using a high-dimensional platform where the number of interrogated sites (P) exceeds the number of samples (n). For example, the Illumina HumanMethylation450 Array from Illumina can interrogate methylation levels of more than 485,000 CpG sites ($P = 485,512$)[Bibikova et al., 2011], and the Reverse phase protein array platform from the MD Anderson Cancer Center can assess protein levels for more than 135 different antibodies ($P \geq 135$) [Tibes et al.,

2006]. This high-dimensionality often causes problems when fitting models, for example, traditional methods for modeling ordinal data do not perform well in the presence of a high-dimensional covariate space, because traditional methods require that the number of samples be greater than the number of covariates and assumes the covariates are independent [Archer et al., 2014b].

Penalized methods have been shown to perform well in linear, logistic regression and Cox proportional hazards models and have just recently been extended to the ordinal response setting [Archer et al., 2014b, Archer and Williams, 2012]. In section 1.2, a brief overview of frequentist-based penalized methods for linear and logistic regression models, as well as a penalized Bayesian approach for the linear model is provided. Section 1.3 focuses on ordinal response models. In section 1.4, two penalized methods for ordinal response models, glmpath.cr and glmnet.cr, which served as motivation for expanding penalized ordinal response models are reviewed.

Throughout this thesis, mathematical notation is consistent unless specifically mentioned. For example, n is always the number of samples and P is always the number of covariates. For observations $i = 1, 2, ...n$, Y_i is the response for i^{th} observation, and $\boldsymbol{x_i}$ is the $P \times 1$ covariate vector for observation i.

1.2 Penalized methods for continuous and binary responses

Tibshirani (1996) proposed a widely used method for dimensionality reduction called the least absolute shrinkage and selection operator (LASSO) [Tibshirani, 1996]. In linear regression, where Y_i is the response variable for the i^{th} individual, α is the intercept, $\boldsymbol{x}_i$ is the $P \times 1$ covariate vector, $\boldsymbol{\beta}$ is the $P \times 1$ vector of slope coefficient and ϵ is the error term that follows a normal distribution with mean 0 and variance σ^2, the model is expressed as

$$Y_i = \alpha + \boldsymbol{x}_i^T \boldsymbol{\beta} + \epsilon.$$

LASSO penalizes the sum of the absolute value of the coefficients (L1-norm) by introducing a regularization parameter λ which shrinks some coefficients to be exactly 0. For a dataset with n observations and P covariates, the LASSO solution can be estimated as

$$\hat{\boldsymbol{\beta}}_{LASSO} = \underset{\beta}{\operatorname{argmin}} \Big(\sum_{i=1}^{n} (Y_i - \alpha - \sum_{j=1}^{P} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{P} |\beta_j| \Big). \tag{1.1}$$

Another common regularization method is ridge regression [Hoerl and Kennard, 1970]. The ridge solution is chosen to minimize the penalized sum of squares:

$$\sum_{i=1}^{n} (Y_i - \alpha - \sum_{j=1}^{P} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{P} \beta_j^2.$$

However, ridge regression does not reduce any coefficients to zero; therefore, it cannot be used for feature selection.

Since LASSO has been proposed, there have been both frequentist and Bayesian approaches developed to obtain a LASSO solution. In section 1.2.1, some frequentist approaches, specifically, the incremental forward stagewise (IFS) method for the linear regression setting and generalized monotone incremental forward stagewise (GMIFS) for the logistic regression setting, are described. In addition, the coordinate descent algorithm introduced by Park and Hastie (2007) for fitting generalized linear models and Cox proportional hazard models is described. In section 1.2.2, the Bayesian LASSO for linear regression is also reviewed.

1.2.1 Frequentist penalized methods

When the outcome variable is continuous, several algorithms can be used to obtain a LASSO solution based on the likelihood. These includes the incremental forward stagewise (IFS) and the L1-Regularization path for generalized linear models (glmpath). First, the IFS method is reviewed given it is the foundation of our penalized method for ordinal response data. The glmpath algorithm is also reviewed, given we later compare our penalized ordinal method with glmpath in the analysis of grouped survival data.

IFS method for linear regression

Hastie, Tibshirani & Friedman (2001) observed that the LASSO solution is strikingly similar to the coefficients estimated by the incremental forward stagewise algorithm (IFS)[Hastie et al., 2007]. IFS increments the coefficient that is most correlated with current residual vector ($\boldsymbol{r}$) by an amount of $\pm\epsilon$ at each step, the sign of ϵ being determined by the sign of the correlation coefficient between the covariate and the residual vector. The algorithm is:

1. Standardize all the predictors.
2. Initialize the residuals, $\boldsymbol{r} = \boldsymbol{y} - \bar{\boldsymbol{y}}$ and for $p = 1, ..., P$ coefficients, let $\beta_1, \beta_2, ..., \beta_P = 0$.
3. Find the predictor, x_j, that is most correlated with $\boldsymbol{r}$ as the predictor to be updated, $j = \text{argmax}_p \, |\rho(r, x_p)|$.
4. Update $\beta_j \rightarrow \beta_j + \delta_j$, where $\delta_j = \epsilon \times sign[\rho(\boldsymbol{r}, x_j)]$.
5. Update $\boldsymbol{r} \rightarrow \boldsymbol{r} - \delta_j x_j$.
6. Repeat steps 3-5 until no predictor has any correlation with $\boldsymbol{r}$.

ϵ is some very small number, for example, 0.001. Although not specified in the original publication, we also defined no correlation as the correlation between a predictor and $\boldsymbol{r}$ is less than a certain threshold, for example, 0.2.

GMIFS method for logistic regression

Hastie et al. (2007) further extended the IFS method to the logistic regression setting, and called it the generalized monotone incremental forward stagewise algorithm (GMIFS) [Hastie et al., 2007]. When the response is discrete, minimizing the residual sum of squares is not reasonable. An alternative procedure was proposed to maximize the log-likelihood incrementally. For a logistic regression model with log-likelihood

$$\log L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (Y_i \log \pi_i + (1 - Y_i) \log(1 - \pi_i))$$

where $\pi_i = \frac{\exp(\boldsymbol{x}_i^T \boldsymbol{\beta})}{1+\exp(\boldsymbol{x}_i^T \boldsymbol{\beta})}$, the algorithm estimates the LASSO solution following the steps below:

1. Standardize the predictors then expand the covariate matrix $\boldsymbol{X}$ to $\tilde{\boldsymbol{X}} = [\boldsymbol{X} : -\boldsymbol{X}]$.
2. Initialize the components of $\hat{\boldsymbol{\beta}}^{(s)}$ at step s=0 to be all 0.
3. Find the predictor x_j that minimizes $-\delta logL/\delta\beta_p$ at the current estimate $\hat{\boldsymbol{\beta}}^{(s)}$, $j = \text{argmin}_p -\delta logL/\delta\beta_p$.
4. Update $\hat{\beta}_j^{(s+1)} \rightarrow \hat{\beta}_j^{(s)} + \epsilon$.
5. Repeat steps 2 - 4 many times.

In step 5, the original paper did not specify the number of times GMIFS needs to be updated. We proposed to stop the algorithm when the difference between two successive $\log L$ less than a small threshold, such as 0.0001. The reason for

expanding the covariate matrix is to simplify computations by mitigating the need for calculating the second derivative to find the direction of the update. Note because the positive and negative versions of each covariate are present in the expanded covariate matrix, the final coefficients are given by $\hat{\beta}_p - \hat{\beta}_{2p}$. Hastie et al. (2007) proved that the monotone LASSO coefficients $\boldsymbol{\beta}$ for the expanded covariate matrix $\tilde{\boldsymbol{X}} = [\boldsymbol{X} : -\boldsymbol{X}]$ and loss function $\log L(\beta)$ is the LASSO solution solved by quadratic programming.

L1-regularization path algorithm for generalized linear models

Other than GMIFS, Park and Hastie (2007) also provided a penalized method for generalized linear models, the L1-regularization path algorithm for generalized linear models (glmpath) [Park and Hastie, 2007]. Their method selects variables based on the amount of penalization of the L1-term and is less greedy than forward selection-backward deletion. At any given penalization parameter λ, glmpath calculates the exact solution for the coefficients and connects these coefficients in a piecewise linear manner for solutions corresponding to other values of λ.

The algorithm first determines the λ_{max} which penalizes all coefficients except the intercept to be zero and then alternates between a predictor and a corrector step as λ decreases by a pre-defined step length. Park and Hastie (2007) introduced the concept of the "active" set where only selected variables on the iteration are contained. For example, if $\lambda = \lambda_{max}$, the active set only contains the intercept. They also suggested to use a step length equal to the difference between λ_k and $\lambda_k + 1$

that will change the active set of variables (k stands for k^{th} iteration).

To illustrate further, on the k^{th} iteration, glmpath proceeds according to the following steps:

1. Determine the step length, $\delta_k = \lambda_{k+1} - \lambda_k$
2. In the predictor step, linearly approximate the coefficient vector: $\hat{\boldsymbol{\beta}}^{k+} = \hat{\boldsymbol{\beta}}^{k} + \delta_k \frac{\delta \boldsymbol{\beta}}{\delta \lambda}$
3. In the correction step, find the exact solution for the coefficient vector by minimizing the likelihood of the generalized linear model, by minimizing the partial likelihood and using $\hat{\boldsymbol{\beta}}^{k+}$ as the initial value. The optimization is achieved by using the coordinate descent algorithm.
4. Repeat the above steps until the active set cannot be augmented any further.

Park and Hasite (2007) also extended their method to the Cox proportional hazard model. The main difference between the original glmpath and the glmpath extended Cox proportional hazard model (Coxpath) is that the algorithm now estimates coefficients by maximizing the partial likelihood of the Cox model.

1.2.2 Bayesian penalized methods

Tibshirani (1996) also suggested that LASSO estimates can be viewed as the modes of the posterior distribution of $\boldsymbol{\beta}$ when $\boldsymbol{\beta}$ have independent and identically distributed

Laplace (e.g, double-exponential) priors [Tibshirani, 1996]. Inspired by this connection, several other authors have proposed using double-exponential (DE) priors to fit a LASSO model from a Bayesian perspective [Hans, 2009, Lykou and Ntzoufras, 2013, Park and Casella, 2008].

Park and Casella (2008) compared their Bayesian LASSO to the ordinary LASSO and Ridge regression using diabetes data studied by Efron et al. [2004]. The diabetes data included 442 diabetes patients (n=442) and 10 baseline covariates (P=10): age, sex, body mass index, average blood pressure, and six blood serum measurements. The response was measured as a quantitative measure of disease progression one year after baseline [Efron et al., 2004].

In their study, they considered a Laplace prior of the form

$$\pi(\boldsymbol{\beta}|\sigma^2) = \prod_{j=1}^{P} \frac{\lambda}{2\sqrt{\sigma^2}} e^{-\lambda|\beta_j|\sqrt{\sigma^2}} \tag{1.2}$$

and because the Laplace distribution can be expressed as a scale mixture of normals

$$\frac{\alpha}{2} e^{-\alpha|z|} = \int_0^\infty \frac{1}{\sqrt{2\pi s}} e^{\frac{-z^2}{2s}} \frac{\alpha^2}{2} e^{\frac{-\alpha^2 s}{2}} ds, \quad \alpha > 0$$

the hierarchical Bayesian LASSO model can be represented as

$$
\begin{aligned}
Y_i|\alpha, \boldsymbol{x}_i, \sigma^2 &\sim N(\alpha + \boldsymbol{x}_i^T \boldsymbol{\beta}, \sigma^2), \\
\boldsymbol{\beta}|\sigma^2, \tau_1^2, \tau_2^2, ..., \tau_P^2 &\sim N_P(\mathbf{0}_P, \sigma^2 \boldsymbol{D}_\tau), \\
\boldsymbol{D}_\tau &= diag(\tau_1^2, ..., \tau_P^2), \\
\sigma^2, \tau_1^2, ..., \tau_P^2 &\sim \pi(\sigma^2) d\sigma^2 \prod_{j=1}^{P} \frac{\lambda^2}{2} e^{-\lambda^2 \tau_j^2/2} d\tau_j^2, \\
\sigma^2, \tau_1^2, ..., \tau_P^2 &> 0.
\end{aligned}
$$

where σ^2 has a flat prior, $\pi(\sigma^2) = \frac{1}{\sigma^2}$.

After performing Gibbs sampling, the final coefficients and their confidence levels were estimated using the medians of posterior distributions and 95% credible intervals. The regularization parameter λ was selected by marginal maximum likelihood using a Monte Carlo Expectation Maximization (EM) algorithm as a complement to the Gibbs sampler. The Monte Carlo EM algorithm estimates λ from the sample of the previous iteration. Specifically, at iteration k,

$$
\lambda^{(k)} = \sqrt{\frac{2P}{\sum_{j=1}^{P} E_{\lambda^{(k-1)}}[\tau_j^2|\tilde{y}]}}
$$

The initial value of λ is $\lambda_0 = P\sqrt{\hat{\sigma}_{LS}^2} / \sum_{j=1}^{p} |\hat{\beta}_j^{LS}|$, where $\hat{\sigma}_{LS}^2$ and $\hat{\beta}_j^{LS}$ are the least squares estimates for the variance and regression parameters. Although Park and Casella (2008) considered a marginal maximum likelihood estimate for λ, they

also mentioned that λ can be chosen by imposing a gamma prior on λ^2 (not λ), where

$$\pi(\lambda^2) = \frac{\delta^r}{\Gamma(r)}(\lambda^2)^{r-1}e^{-\delta\lambda^2}, \;\; \lambda^2 > 0 \; (r > 0, \delta > 0).$$

The prior density for λ^2 should have a high probability near the maximum likelihood estimate. They further argued that although choosing an improper scale-invariant prior for λ^2 (e.g, r=0, $\delta = 0$) is attractive, it will result in an improper posterior. In their diabetes data example, they showed that a Gibbs sampler with $\lambda^2 \sim Gamma(1, 1.78)$ produced posterior medians for the regression coefficients identical to those when λ was selected by marginal maximum likelihood [Park and Casella, 2008]. Hans (2001) also considered selecting λ by assigning it an independent gamma prior distribution, but rather than λ^2, Hans (2001) suggested to impose a gamma prior on λ [Hans, 2009].

In the end, Park and Casella (2008) plotted the LASSO, Bayesian LASSO, and Ridge estimates against their respective L1 norm fraction ($\frac{\|\hat{\beta}\|}{\max\|\hat{\beta}\|}$). They showed that the Bayesian LASSO appeared to be a compromise between the ordinary LASSO and ridge regression. Bayesian LASSO, like ridge estimates, had smoother paths, but the shape was more like the LASSO path. Moreover, the Bayesian LASSO appeared to be able to penalize weakly related parameters to 0 more quickly than ridge regression and the Bayesian posterior medians were very similar to the LASSO estimates determined by cross-validation.

Lykou and Ntzoufras (2013) proposed a Bayesian LASSO that accomplishes both

shrinkage and variable selection in the linear regression setting. To enable variable selection, they utilized binary variable inclusion indicators introduced by George and McCulloch (1993) and widely used thereafter by Kuo and Mallick (1998) and Dellaportas et al. (2002). When Y is a dependent variable and $X_1, X_2, ..., X_P$ is the set of potential predictors, binary variable selection is performed by introducing a vector of binary variable inclusion indicators $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, ..., \gamma_P)$, and then model the j^{th} slope coefficient β_j as having come from a mixture of two normal distributions, where

$$\beta_j|\gamma_j \sim (1-\gamma_j)N(0, \tau_j^2) + r_j N(0, c_j^2\tau_j^2)$$

and

$$P(\gamma_j = 1) = 1 - P(\gamma_j = 0) = \pi_j.$$

George and McCulloch (1993) then set τ_j^2 to be a very small number and c_j to be a very large number so that when $\gamma_j = 0$, $\beta_j \sim N(0, \tau_j^2)$ is almost 0, and when $\gamma_j = 1$, $\beta_j \sim N(0, c_j^2\tau_i^2)$ is estimated as a non-zero slope coefficient. In other words, $\gamma_j = 1$ indicates the j^{th} parameter is included in the model while $\gamma_j = 0$ indicates that the j^{th} parameter is not included in the model. The method can then be processed using Gibbs sampling. Those variables with higher probabilities, the variables that more frequently appear in the Gibbs sample or with higher posterior inclusion probability, are then selected. The binary variable inclusion indicator is especially useful when the number of predictors is large, because most traditional variable selection methods requires some comparison of model selection criteria (e.g. AIC, BIC or Cp) for all 2^p possible submodels, while this method can avoid calculating the posterior

probabilities of all 2^p subsets [George and McCulloch, 1993].

Using a prior Bernoulli distribution of γ_j with hyperparameter π ($\gamma_j \sim Bernoulli(\pi)$ for $j = 1, 2, ..., P$), Lykou and Ntzoufras (2013) express their model as

$$\begin{aligned} Y_i|\alpha, \boldsymbol{\beta}, \boldsymbol{x}_i, \tau, \boldsymbol{\gamma} &\sim N(\alpha + \boldsymbol{x}_i^T \boldsymbol{D}_{\boldsymbol{\gamma}} \boldsymbol{\beta}, \tau^{-1}), \\ \text{where } \boldsymbol{D}_{\boldsymbol{\gamma}} &= diag(\gamma_1, ..., \gamma_P), \\ \boldsymbol{\beta}|\tau &\sim DE(\mathbf{0}, \frac{1}{\tau\lambda}), \\ \gamma_j &\sim Bernoulli(\pi_j), \\ \tau &\sim Gamma(c, d) \end{aligned}$$

where $\tau = \frac{1}{\sigma^2}$ is the precision of the regression model. The posterior mean of γ_j is used to estimate the posterior inclusion probabilities of each covariate. Lykou and Ntzoufras select a variable if its posterior inclusion probability is greater than 0.5 [Lykou and Ntzoufras, 2013].

1.3 Ordinal response models

In this section, four common ordinal response models are presented. This includes the cumulative logit model, adjacent category model, forward and backward continuation ratio models. Each model has its own unique characteristics and depending on the data structure and model interpretation, particular models can perform better than others. For example, the backward continuation ratio model is preferable when

interest lies in estimating the odds of more severe disease compared to less severe disease, and the forward continuation ratio model is preferable when interest lies in estimating the odds of having short survival time compared to longer survival time. In addition to these four commonly used models, an overview of the stereotype logit model, which can be used when the proportional odds assumption does not hold, is provided. Moreover, at the end of this chapter, Bayesian inference for ordinal response data is also briefly described.

For a dataset that contains n observations and P covariates, define Y_i as the response for the i^{th} subject, where Y_i can fall into one of K categories ($k = 1, 2, ..., K$). Define α_j to be a class-specific intercept for the j^{th} class and $\boldsymbol{\beta}$ to be the $P \times 1$ coefficient vector associated with covariate vector $\boldsymbol{x}_i$. Let $\pi_j(\boldsymbol{x}_i)$ be the probability that the i^{th} response falls into the j^{th} category. Using these notations, the different ordinal response models are described in the following subsections.

1.3.1 Cumulative logit model

The cumulative logit model is probably the most frequently used ordinal response model, expressed as

$$P(Y_i \leq j|\boldsymbol{x_i}) = \frac{\exp(\alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta})}{1 + \exp(\alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta})} \quad j = 1, ..., K-1. \tag{1.3}$$

Class-specific probabilities can be calculated by substracting successive cumulative logits,

$$\pi_j(\boldsymbol{x}_i) = P(Y_i \leq j|\boldsymbol{x}_i) - P(Y_i \leq j-1|\boldsymbol{x}_i).$$

Note that the model assumes the $\boldsymbol{\alpha}$ to have a monotonic ordering constraint, $-\infty = \alpha_0 < \alpha_1 \leq \alpha_2 \leq ... \leq \alpha_{K-1} < \alpha_K = \infty$.

Although the logit link is probably the most popular link for the cumulative link model, the probit link is also useful in many situations. For example, the probit link is useful when the outcome is a survey response, because many assume that survey outcomes will follow an underlying normal distribution [Agresti, 2010]. In chapter 2, the extension of the GMIFS penalization method for probit link models is described in detail.

1.3.2 Adjacent category model

The adjacent-category model models the pairs of adjacent categories, and has the form

$$\log\left(\frac{\pi_j(\boldsymbol{x}_i)}{\pi_{j+1}(\boldsymbol{x}_i)}\right) = \alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta}, \quad j = 1, ..., K-1. \tag{1.4}$$

In addition, the adjacent-category model can be expressed as a baseline-category logit model. For baseline category K, the model is

$$\begin{aligned}\log\frac{\pi_j}{\pi_K} &= \log(\frac{\pi_j}{\pi_{j+1}}) + \log(\frac{\pi_{j+1}}{\pi_{j+2}}) + ... + \log(\frac{\pi_{K-1}}{\pi_K}) \\ &= \alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta} + \alpha_{j+1} + \boldsymbol{x}_i^T\boldsymbol{\beta} + ... + \alpha_{K-1} + \boldsymbol{x}_i^T\boldsymbol{\beta} \\ &= \sum_{k=j}^{K-1}\alpha_k + (K-j)\boldsymbol{x}_i^T\boldsymbol{\beta}. \end{aligned} \tag{1.5}$$

1.3.3 Forward and backward continuation ratio models

The backward continuation ratio model using the logit link can be expressed as

$$\text{logit}(P(Y=j|Y\leq j,\boldsymbol{x})) = \log\left(\frac{P(Y=j|Y\leq j,\boldsymbol{x})}{P(Y<j|Y\leq j,\boldsymbol{x})}\right) = \alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta} \tag{1.6}$$

where $j = 2, ..., K$. Then the conditional probabilities can be rewritten as

$$P(Y=j|Y\leq j,\boldsymbol{x}) = \frac{\exp(\alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta})}{1+\exp(\alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta})}.$$

Whereas the forward formulation models can be expressed as

$$\text{logit}(P(Y=j|Y\geq j,\boldsymbol{x})) = \log\left(\frac{P(Y=j|Y\geq j,\boldsymbol{x})}{P(Y<j|Y\geq j,\boldsymbol{x})}\right) = \alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta} \tag{1.7}$$

where $j = 1, ..., K-1$.

1.3.4 Stereotype logit model

The cumulative logit, adjacent category, forward and backward continuation ratio models all assume proportional odds (PO). In other words, covariates will have same effect on the outcome, regardless of the level of the outcome. Unfortunately, this assumption does not hold in many situations. For example, we tested the assumption using a gene expression dataset assayed by Affymetrix HG-U133A GeneChips. The data contains hippocampal gene expression of 9 control and 22 Alzheimer's Disease subjects of varying severity (7 incipient, 8 moderate, and 7 severe). The MAS5 method was used to obtain probe set expression summaries. individual gene probe set values were treated as missing values if they were > standard deviation from the group mean and any probe sets that were missing in all 31 samples were removed, leaving 15,189 probe sets [Blalock et al., 2004]. Score tests indicated that the PO assumption did not hold for approximately 10% of probe sets. By letting some or all predictors have non-proportional odds, we can possibly improve the model fit [Agresti, 2010]. Using a baseline adjacent categories model as an example, if we let predictors have a different effect on each pair of adjacent categories, the model is then called a baseline categories model and can be expressed as

$$\log(\frac{\pi_j(\boldsymbol{x}_i)}{\pi_K(\boldsymbol{x}_i)}) = \alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta}_j, \quad j = 1, 2, ...K-1 \tag{1.8}$$

Note that this model contains $K-1$ slope parameters for each predictor instead of a single slope parameter in equation (1.4). If model contains too many parameters, which is most often the case in the high-dimensional data analysis, the model can

be very complicated.

The stereotype logit model is a compromise between the violation of proportional odds assumption and model being too complicated. This model was proposed by Anderson in 1984 [Anderson, 1984]. For baseline-category K , the stereotype logit model is

$$\log(\frac{\pi_j(\boldsymbol{x}_i)}{\pi_K(\boldsymbol{x}_i)}) = \alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta}, \;\; j = 1, ..., K-1 \tag{1.9}$$

where $\boldsymbol{\phi} = (\phi_1, \phi_2, ..., \phi_{K-1})$ can be regarded as scores for the outcome categories. Because the relationship between the linear components and the response is ordinal, constraints are needed to make the model identifiable. Anderson (1984) recommended the constraint $1 = \phi_1 \geq \phi_2 \geq ... \geq \phi_k = 0$.

For logit j, the explanatory variable x_k now has coefficients $\phi_j \beta_k$. That is, if x_k increases by 1 unit, the odds of response j instead of K are $\exp(\phi_j \beta_k)$ times greater than the original odds. This model is more parsimonious than the baseline-category logit model. When the ordinal model contains a large number of predictors (P is large), this model contains $K-1$ pairs of $\boldsymbol{\alpha}$ and $\boldsymbol{\phi}$ parameters and P slope coefficients. This is less than the baseline-category model, which contains $K-1$ intercepts, and $(K-1)P$ slopes [Agresti, 2010].

1.3.5 Bayesian ordinal regression model

Over the last few years, there has been increasing popularity of an alternative approach for analyzing ordinal response data, the Bayesian approach. Different from the frequentist approach, the Bayesian approach includes probability distributions

for parameters as well as for data. It assumes a prior distribution for the parameters which may reflect our prior beliefs, and these priors are combined with the data likelihood function to generate a posterior distribution for the parameters [Agresti, 2010]. However, in many situations, there is no closed-form expression for the posterior distribution of the ordinal model parameters. Fortunately, simulation methods can be used to approximate the posterior distribution. One of these simulation methods is the Markov Chain Monte Carlo (MCMC) method. MCMC is a stochastic process of Markov chains designed so that its long-run stationary distribution is the posterior distribution. Herein, the program Just Another Gibbs Sampler (JAGS) was used to perform MCMC for the statistical analysis of the Bayesian hierarchical models.

In this section, a Bayesian cumulative logit model is presented as an example of a Bayesian ordinal response model. It can be challenging to find a sensible prior when the parameters relate to cumulative logit. One simple approach is taking the prior distribution to be constant over all possible parameter values, in this way, the posterior distribution is a constant multiple of the likelihood. In other words, the posterior is a scaling of the likelihood so that it integrates to 1. The mode of the posterior distribution is then the ML estimate [Agresti, 2010].

In the Bayesian cumulative logit model, Chipman and Hamada suggested to use a multivariate normal prior distribution for the slope terms $\boldsymbol{\beta}$ and a truncated multivariate normal distribution for the intercept terms $\boldsymbol{\alpha}$ that enable ordering of the $\boldsymbol{\alpha}$ values [Chipman and Hamada, 1996]. If we assume independent covariates,

the model can be represented as

$$
\begin{aligned}
P(Y_i \leq K|\boldsymbol{x_i}) &= \frac{\exp(\alpha_k + \boldsymbol{x}_i^T\boldsymbol{\beta})}{1+\exp(\alpha_k + \boldsymbol{x}_i^T\boldsymbol{\beta})} \\
\pi_K(\boldsymbol{x}_i) &= P(Y_i \leq K|\boldsymbol{x_i}) - P(Y_i \leq K-1|\boldsymbol{x_i}) \\
Y_i &\sim dcat(\pi_1, ..., \pi_K) \\
\alpha_1 &\sim N(0, \sigma_a^2) \\
\alpha_j \sim N(0, \sigma_a^2) \;\; \alpha_j \in (\alpha_{j-1}, \infty) \;\; j &= 2, ..., K-1 \\
\beta_j &\sim N(0, \sigma_b^2)
\end{aligned} \tag{1.10}
$$

where σ_a^2 and σ_b^2 are very large numbers so that the prior distribution is extremely diffuse.

1.4 Penalized methods for ordinal response models

The first penalized ordinal response models were penalized continuation ratio models estimated using the coordinate descent algorithm and implemented using glmpath and glmnet [Archer and Williams, 2012]. These two methods have been made available in the R programming environment in the *glmpathcr* and *glmnetcr* Comprehensive R Archive Network packages [Archer et al., 2014a]. Both methods benefit from the reconstruction of the likelihood function for the continuation ratio model.

In these methods, we again define $\boldsymbol{x}_i$ as a $P \times 1$ covariate vector for individual

i $(i = 1, ..., n)$ and Y_i as the ordinal response for the i^{th} individual that can take on one of K ordinal levels. Y_i can then be re-written as a response matrix containing n rows and K columns where

$$y_{ij} = \begin{cases} 1 & \text{if observation } i \text{ is class } j \\ 0 & \text{otherwise.} \end{cases} \tag{1.11}$$

In this way, we can represent the likelihood for an ordinal response with K ordinal levels as

$$L = \prod_{i=1}^{n} \prod_{j=1}^{K} \pi_j(\boldsymbol{x}_i)^{y_{ij}} \tag{1.12}$$

and the log-likelihood as

$$logL = \sum_{i=1}^{n} \sum_{j=1}^{K} y_{ij} \log(\pi_j(\boldsymbol{x}_i)) \tag{1.13}$$

where $\pi_j(\boldsymbol{x}_i)$ is the probability that individual i with covariates $\boldsymbol{x}_i$ falls into the j^{th} class. In terms of continuation ratio model, the conditional probability can be modeled as

$$\delta_{ij} = \delta_j(\boldsymbol{x}_i) = P(Y = j | Y \leq j, \boldsymbol{x}_i) = \frac{\exp(\alpha_j + \boldsymbol{x}_i^T \boldsymbol{\beta})}{1 + \exp(\alpha_j + \boldsymbol{x}_i^T \boldsymbol{\beta})}.$$

The likelihood is then the product of conditionally independent binomial terms,

$$L(\boldsymbol{\beta} | \boldsymbol{y}, \boldsymbol{x}) = \prod_{i=1}^{n} \prod_{j=2}^{K} \delta_{ij}^{y^{ij}} (1 - \delta_{ij})^{1 - \sum_{k=j}^{K} y_{ik}},$$

such that the likelihood can be factored into $K-1$ independent likelihoods, so that maximization of the independent likelihoods will lead to an overall maximum likelihood. Therefore, a penalized continuation ratio model can be estimated by passing a restructured dataset to a penalized logistic regression function like glmpath or glmnet. The resulting L1 penalized continuation ratio models were referred to as glmpath.cr and glmnet.cr, respectively [Archer et al., 2014a, Archer and Williams, 2012].

Unfortunately, extending glmpath or glmnet to other types ordinal models, for example, cumulative link, adjacent category, and the stereotype logit models is not straightforward. As a result, we extend the GMIFS method to ordinal response models [Archer et al., 2014b]. Specifically, this research details the extension of the GMIFS method to probit link ordinal response models and the stereotype logit model. Since the GMIFS algorithm for logit link models and adjacent category model are similar to the extension to probit link and sterotype logit model, the GMIFS will not be reviewed here in detail. In chapter 2, the GMIFS algorithm is described thoroughly with examples when implementing probit link models and the stereotype logit model.

Although the Bayesian LASSO has been implemented in the linear regression setting, there is currently no penalized Bayesian ordinal response models. Therefore, another goal of this dissertation is to implement Bayesian LASSO to ordinal response models such as cumulative logit, cumulative probit, adjacent category, and

continuation ratio models.

In the following chapters, our GMIFS extension to both the probit link ordinal model and the stereotype logit model is reviewed (Chapter 2). In chapter 3, our implementation of the Bayessian LASSO for modeling an ordinal response is described. In chapter 4, filtering, which is another dimension reduction method (other than penalization) is addressed, and we compare filtering, or univariate feature selection methods, with penalization methods using grouped survival data. The last chapter provides our conclusions and discussion.

Chapter 2

Penalized Probit models and Stereotype logit model

2.1 GMIFS ordinal response models

In this chapter, the GMIFS algorithm for three probit link models: cumulative probit model, forward continuation-ratio model with probit link and backward continuation-ratio model with probit link, as well as the stereotype logit model, are described. Although GMIFS was originally proposed by Hastie for logistic regression [Hastie et al., 2007], Archer et al. (2014) extended GMIFS algorithm to the ordinal response setting and developed an R package, *ordinalgmifs*, to implement the algorithm [Archer et al., 2014b]. In this section, the GMIFS algorithm for ordinal response model is reviewed. In section 2.2, we detail the extension of the GMIFS method to probit link models, specifically, the cumulative probit model, as well as the backward and

forward continuation ratio models with a probit link. The computational details necessary for utilizing the GMIFS algorithm are also described. In section 2.3, GMIFS is extended to the stereotype logit model. The *ordinalgmifs* package provides functions that fits both penalized probit models and the stereotype logit model. The function code is included in Appendix A.

Similar to glmnet.cr and glmpath.cr (section 1.4), the GMIFS algorithm for ordinal response models also requires the construction of an ordinal response variable, Y_i and its corresponding likelihood function (equation (1.12), equation (1.13)). The likelihood and log-likelihood function differs based on the model as shown in next two sections.

In general, most ordinal response models have $K-1$ intercept terms $\boldsymbol{\alpha}$ and P slope terms, $\boldsymbol{\beta}$. The $\boldsymbol{\alpha}$ are essential for an ordinal model that assumes proportional odds because slope terms, $\boldsymbol{\beta}$ do not have category specific effects. Therefore, $\boldsymbol{\alpha}$ are the only parameters that distinguish between the different ordinal levels. GMIFS produces a series of solutions, each time selecting the slope coefficient which leads to the maximum decrease in the negative likelihood and updating that slope coefficient by a small increment, ϵ. Then the algorithm calculates Maximum Likelihood (ML) estimates of $\boldsymbol{\alpha}$ based on the current slope coefficients using a quasi-Newton or modified quasi-Newton optimization. This iterative process stops until the difference between two successive log-likelihoods is smaller than a pre-specified tolerance τ, where a typical τ is 0.0001. Additionally, during each iteration, the AIC or BIC

can be calculated, which enables model selection based on commonly used criteria.

For K ordinal classes and P predictors, the GMIFS algorithm is as follows:

1. Standardize the predictors then expand covariate matrix $\boldsymbol{X}$ to $\tilde{\boldsymbol{X}} = [\boldsymbol{X} : -\boldsymbol{X}]$.
2. Initialize $\boldsymbol{\alpha}$ based on the specific model being fit (see the following 2.2 and 2.3 sections).
3. Initialize the components of $\hat{\boldsymbol{\beta}}^{(s)}$ at step s=0 to be all 0.
4. Find $m = \text{argmin}_p(-\delta logL/\delta\beta_p)$ at the current estimate $\hat{\boldsymbol{\beta}}^s$.
5. Update $\hat{\boldsymbol{\beta}}_m^{s+1} \rightarrow \hat{\boldsymbol{\beta}}_m^s + \epsilon$.
6. Consider $\hat{\boldsymbol{\beta}}^{s+1}$ as fixed, update $\boldsymbol{\alpha}_s$ using the maximum likelihood method.
7. Repeat steps 4 - 6 until $logL^{(s+1)} - logL^{(s)} < \tau$.

2.2 Penalized Probit models

For binary or ordinal data, regression models can use link functions other than the logit, for example, the probit link. The term probit is short for "probability unit" and was first introduced in the 1930's by biologists to model data such as the percentage of a pest killed by pesticide [Bliss, 1934]. The probit function is the inverse of the standard normal cumulative distribution function (CDF) denoted by Φ. The standard normal CDF has similar shape to that of the symmetric S-shape of standard

logistic CDF, the difference lies in that the standard normal has a mean of 0 and a standard deviation of 1, while standard logistic has a mean of 0 and a standard deviation of $\frac{\pi}{\sqrt{3}} \approx 1.81$. Because of this similarity, a logistic model and a probit model tends to fit similarly for the same data.

In section 2.2.1 and 2.2.2, the cumulative probit model and continuation ratio models with probit link are presented, together with their log-likelihood, gradients, and initial intercept estimates that are necessary for utilizing GMIFS algorithm. In addition, an application of the penalized cumulative probit model was demonstrated to predict depression level in women with breast cancer using methylation dataset in section 2.2.3.

2.2.1 Cumulative probit model

The cumulative probit models $K-1$ probits of the form.

$$P(Y_i \leq j|\boldsymbol{x_i}) = \Phi(\alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta}) \tag{2.1}$$

where Φ is the cumulative distribution function (CDF) of the standard normal distribution, α_j denotes the class-specific intercept, and $\boldsymbol{\beta}$ is a $p \times 1$ vector of coefficients associated with explanatory variables $\boldsymbol{x_i}$. Note that the class-specific probabilities can be calculated by subtracting successive cumulative probabilities,

$$\pi_j(\boldsymbol{x_i}) = P(Y_i \leq j|\boldsymbol{x_i}) - P(Y_i \leq j-1|\boldsymbol{x_i})$$

Therefore, for any class j, we can express the class-specific probabilities by

$$\pi_j(\boldsymbol{x}_i) = \Phi(\alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta}) - \Phi(\alpha_{j-1} + \boldsymbol{x}_i^T\boldsymbol{\beta})$$

Similar to the cumulative logit model (equation (1.3)), when we require $\alpha_0 = -\infty$ and $\alpha_K = \infty$, this expression simplifies to

$$\pi_1 = \Phi(\alpha_1 + \boldsymbol{x}_i^T\boldsymbol{\beta})$$

for $j = 1$ and

$$\pi_K = 1 - \Phi(\alpha_{K-1} + \boldsymbol{x}_i^T\boldsymbol{\beta})$$

for $j = K$.

Therefore, the derivative of the reconstructed log-likelihood (equation (1.13)) with respect to p^{th} slope term β_p is

$$\begin{aligned}\frac{\partial \log L}{\partial \beta_p} &= \frac{\partial \sum_{i=1}^{n} \left(y_{i1} \log(\pi_1(\boldsymbol{x}_i)) + \sum_{j=2}^{K-1} y_{ij} \log(\pi_j(\boldsymbol{x}_i)) + y_{iK} \log(\pi_k(\boldsymbol{x}_i)) \right)}{\partial \beta_p} \\ &= \sum_{i=1}^{n} y_{i1} \frac{\partial \log(\pi_1(\boldsymbol{x}_i))}{\partial \beta_p} + \sum_{i=1}^{n} \sum_{j=2}^{K-1} y_{ij} \frac{\partial \log(\pi_j(\boldsymbol{x}_i))}{\partial \beta_p} + \sum_{i=1}^{n} y_{iK} \frac{\partial \log(\pi_K(\boldsymbol{x}_i))}{\partial \beta_p}\end{aligned}$$

Let ϕ represent the probability density function of the standard normal distribution.

For $j = 1$,

$$\begin{aligned}\frac{\partial \log(\pi_1(\boldsymbol{x}))}{\partial \beta_p} &= \frac{\partial \log(\Phi(\alpha_1 + \boldsymbol{x}^T\boldsymbol{\beta}))}{\partial \beta_p} \\ &= \frac{\phi(\alpha_1 + \boldsymbol{x}^T\boldsymbol{\beta})}{\Phi(\alpha_1 + \boldsymbol{x}^T\boldsymbol{\beta})}\boldsymbol{x}_p^T.\end{aligned}$$

For $j = 2, ..., K-1$,

$$\frac{\partial \log(\pi_j(\boldsymbol{x}))}{\partial \beta_p} = \frac{\phi(\alpha_j + \boldsymbol{x}^T\boldsymbol{\beta}) - \phi(\alpha_{j-1} + \boldsymbol{x}^T\boldsymbol{\beta})}{\Phi(\alpha_j + \boldsymbol{x}^T\boldsymbol{\beta}) - \Phi(\alpha_{j-1} + \boldsymbol{x}^T\boldsymbol{\beta})}\boldsymbol{x}_p^T.$$

For $j = K$,

$$\frac{\partial \log(\pi_K(\boldsymbol{x}))}{\partial \beta_p} = \frac{-\phi(\alpha_{K-1} + \boldsymbol{x}^T\boldsymbol{\beta})}{1 - \Phi(\alpha_{K-1} + \boldsymbol{x}^T\boldsymbol{\beta})}\boldsymbol{x}_p^T.$$

Therefore, we have

$$\begin{aligned}\frac{\partial logL}{\partial \beta_p} =& \boldsymbol{x}_p^T\Big(\frac{\phi(\alpha_1 + \boldsymbol{x}^T\boldsymbol{\beta})\boldsymbol{y}_1}{\Phi(\alpha_1 + \boldsymbol{x}^T\boldsymbol{\beta})} + \sum_{j=2}^{K-1}\frac{(\phi(\alpha_j + \boldsymbol{x}^T\boldsymbol{\beta}) - \phi(\alpha_{j-1} + \boldsymbol{x}^T\boldsymbol{\beta}))\boldsymbol{y}_j}{\Phi(\alpha_j + \boldsymbol{x}^T\boldsymbol{\beta}) - \Phi(\alpha_{j-1} + \boldsymbol{x}^T\boldsymbol{\beta})} \\ &- \frac{\phi(\alpha_{K-1} + \boldsymbol{x}^T\boldsymbol{\beta})\boldsymbol{y}_K}{1 - \Phi(\alpha_{K-1} + \boldsymbol{x}^T\boldsymbol{\beta})}\Big)\end{aligned} \tag{2.2}$$

In the GMIFS algorithm, the α terms are initialized considering all slope terms in equation 2.1 to be 0:

$$\begin{aligned}P(Y_i \leq j|\boldsymbol{x_i}) &= \Phi(\alpha_j) \\ \alpha_j &= \Phi^{-1}(\sum_{i=1}^{n}\sum_{k=1}^{j} y_{ik}/n).\end{aligned}$$

Subsequently, $\boldsymbol{\alpha}$ is estimated with the MLE using quasi-Newton optimization (BFGS method) after each $\boldsymbol{\beta}$ update. Because the class-specific probabilities are

obtained by subtracting successive cumulative probabilities, we require $\alpha_1 \leq \alpha_2 \leq ... \leq \alpha_{K-1}$. In the GMIFS extended cumulative probit model, the solution of the constrained optimization is solved by using an adaptive barrier method. The adaptive barrier method minimizes the twice continuously differentiable function $\log L$ subject to the linear inequality constraints $A\boldsymbol{\theta} - b \geq 0$, where A is a $K-1$ by K matrix,

$$A = \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & \dots & K-1 & K \end{array} \\ \begin{pmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -1 & 1 \end{pmatrix} \end{array} \begin{array}{c} 1 \\ 2 \\ \vdots \\ K-1 \end{array},$$

$\boldsymbol{\theta} = [\alpha_1, \alpha_2, \alpha_3, ..., \alpha_{K-1}]$, and $b = \mathbf{0}_{K-1\times 1}$. We carried out the constrained optimization using the package *constrOptim* in the R programming environment. Since the derivative of log-likelihood function with respect to $\boldsymbol{\alpha}$ is necessary for optimization, we derive the partial derivative as described below.

For $j = 1$,

$$\begin{aligned}\frac{\partial \log L}{\partial \alpha_1} &= \sum_{i=1}^{n}(y_{i1}\frac{\partial \log(\pi_1(x_i))}{\partial \alpha_1} + y_{i2}\frac{\partial \log(\pi_2(x_i))}{\partial \alpha_2}) \\ &= \sum_{i=1}^{n}\Bigg(y_{i1}\frac{\partial \log \Phi(\alpha_1 + x_i^T\boldsymbol{\beta})}{\partial \alpha_1} \\ &- y_{i2}\frac{\partial \log(\Phi(\alpha_2 + x_i^T\boldsymbol{\beta}) - \Phi(\alpha_1 + x_i^T\boldsymbol{\beta}))}{\partial \alpha_2}\Bigg) \\ &= \sum_{i=1}^{n}\Bigg(y_{i1}\frac{\phi(\alpha_1 + x_i^T\boldsymbol{\beta})}{\Phi(\alpha_1 + x_i^T\boldsymbol{\beta})} - \\ & y_{i2}\frac{\phi(\alpha_1 + x_i^T\boldsymbol{\beta})}{\Phi(\alpha_2 + x_i^T\boldsymbol{\beta}) - \Phi(\alpha_1 + x_i^T\boldsymbol{\beta})}\Bigg). \end{aligned} \tag{2.3}$$

For $j = 2, ..., K-2$,

$$\begin{aligned}\frac{\partial \log L}{\partial \alpha_j} &= \sum_{i=1}^{n}(y_{ij}\frac{\partial \log(\pi_j(x_i))}{\partial \alpha_j} + y_{i(j+1)}\frac{\partial \log(\pi_{j+1}(x_i))}{\partial \alpha_j}) \\ &= \sum_{i=1}^{n}\Bigg(y_{ij}\frac{\phi(\alpha_j + x_i^T\boldsymbol{\beta})}{\Phi(\alpha_j + x_i^T\boldsymbol{\beta}) - \Phi(\alpha_{j-1} + x_i^T\boldsymbol{\beta})} \\ &- y_{i(j+1)}\frac{\phi(\alpha_j + x_i^T\boldsymbol{\beta})}{\Phi(\alpha_{j+1} + x_i^T\boldsymbol{\beta}) - \Phi(\alpha_j + x_i^T\boldsymbol{\beta})}\Bigg). \end{aligned} \tag{2.4}$$

For $j = K-1$,

$$\begin{aligned}\frac{\partial \log L}{\partial \alpha_{K-1}} &= \sum_{i=1}^{n}\Bigg(y_{i(K-1)}\frac{\partial \log(\pi_{K-1}(x_i))}{\partial \alpha_{K-1}} + y_{iK}\frac{\partial \log(\pi_K(x_i))}{\partial \alpha_{K-1}}) \\ &= \sum_{i=1}^{n}(y_{i(K-1)}\frac{\phi(\alpha_{K-1} + x_i^T\boldsymbol{\beta})}{\Phi(\alpha_{K-1} + x_i^T\boldsymbol{\beta}) - \Phi(\alpha_{K-2} + x_i^T\boldsymbol{\beta})} \\ &- y_{iK}\frac{\phi(\alpha_{K-1} + x_i^T\boldsymbol{\beta})}{1 - \Phi(\alpha_{K-1} + x_i^T\boldsymbol{\beta})}\Bigg). \end{aligned} \tag{2.5}$$

The code for fitting GMIFS cumulative probit models appears in Appendix A.1.

2.2.2 Continuation ratio models with probit link

Backward continuation ratio model

The backward continuation ratio model with the probit link is similar to the backward continuation ratio model with the logit link. It models the probit of the $j = 2, .., K$ conditional probabilities or

$$\Phi^{-1}(P(Y_i = j|Y_i \leq j, \boldsymbol{x}_i)) = \alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta}. \tag{2.6}$$

The backward formulation can be used when the response variable represents disease states from none, mild, moderate, and severe and interest is in estimating the odds of more severe disease compared to less severe disease. Let δ_{ij} represent the conditional probabilities,

$$\delta_{ij} = \delta_j(\boldsymbol{x_i}) = P(Y_i = j|Y_i \leq j, \boldsymbol{x}_i) = \Phi(\alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta})$$

such that for K ordinal classes, there are $K - 1$ probits. The likelihood can be expressed using these $j = 2, ..., K$ conditionally independent probabilities:

$$L(\boldsymbol{\beta}|\boldsymbol{Y}, \boldsymbol{X}) = \prod_{i=1}^{n}\prod_{j=2}^{K}\delta_{ij}^{y_{ij}}(1 - \delta_{ij})^{1-\sum_{k=j}^{K} y_{ik}}$$

which can be seen as the product of $K-1$ binomial likelihoods. Using this expression, the log-likelihood is

$$\log L = \sum_{i=1}^{n}\sum_{j=2}^{K}\big(y_{ij}\log(\delta_{ij}) + (1-\sum_{k=j}^{K} y_{ik})\log(1-\delta_{ij})\big)$$

Then, the derivative of the log-likelihood with respect to β_p for the backward continuation ratio is given by

$$\begin{aligned}\frac{\partial \log L}{\partial \beta_p} &= \sum_{i=1}^{n}\sum_{j=2}^{K}\frac{y_{ij}}{\delta_{ij}}\frac{\partial \delta_{ij}}{\partial \beta_p} + \frac{\sum_{k=j}^{K} y_{ik} - 1}{1-\delta_{ij}}\frac{\partial \delta_{ij}}{\partial \beta_p} \\ &= \sum_{j=2}^{K}\sum_{i=1}^{n} x_{ip}\Big(y_{ij}\frac{\phi(\alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta})}{\Phi(\alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta})} + (\sum_{k=j}^{K} y_{ik} - 1)\frac{\phi(\alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta})}{1-\Phi(\alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta})}\Big)\end{aligned}$$

which can be rewritten as the matrix form

$$\frac{\partial logL}{\partial \beta_p} = \sum_{j=2}^{K}\boldsymbol{x}_p^T\Big(\boldsymbol{y}_j\frac{\phi(\alpha_j + \boldsymbol{x^T\beta})}{\Phi(\alpha_j + \boldsymbol{x^T\beta})} + (\sum_{k=j}^{K}\boldsymbol{y}_k - 1)\frac{\phi(\alpha_j + \boldsymbol{x^T\beta})}{1-\Phi(\alpha_j + \boldsymbol{x^T\beta})}\Big). \tag{2.7}$$

In the GMIFS algorithm, the α terms are initialized considering all slope terms in equation 2.6 to be 0, where

$P(Y_i = j | Y_i \leq j, \boldsymbol{x}_i) = \Phi(\alpha_j)$, such that

$$\alpha_j = \Phi^{-1}(\frac{\sum_{i=1}^{n} y_{ij}}{\sum_{i=1}^{n}\sum_{k=1}^{j} y_{ik}}).$$

Forward continuation ratio model

The forward continuation ratio model with the probit link models the probit of the $j = 1, .., K-1$ conditional probabilities or

$$\Phi^{-1}(P(Y_i = j|Y_i \geq j, \boldsymbol{x_i})) = \alpha_j + \boldsymbol{x}_i^T\boldsymbol{\beta} \tag{2.8}$$

Here we have used the forward formulation, which can be used when the response variable is grouped survival time, and the goal is to estimate the odds of shorter survival time compared to longer survival time. As with the backward continuation ratio model, the likelihood and log-likelihood for the forward continuation ratio model can be expressed using the $K-1$ conditionally independent probabilities,

$$L(\boldsymbol{\beta}|\boldsymbol{Y}, \boldsymbol{X}) = \prod_{i=1}^{n}\prod_{j=1}^{K-1}\delta_{ij}^{ij}(1-\delta_{ij})^{\sum_{k=j}^{K} y_{ik}-y_{ij}}$$

$$\log L = \sum_{i=1}^{n}\sum_{j=1}^{k-1} y_{ij}\log(\delta_{ij}) + (\sum_{k=j}^{K} y_{ik} - y_{ij})\log(1-\delta_{ij})$$

The partial derivative of the log-likelihood with respect to β_p for the forward continuation ratio model with probit link is given by,

$$\begin{aligned}\frac{\partial logL}{\partial \beta_p} &= \sum_{i=1}^{n}\sum_{j=1}^{K-1}\frac{y_{ij}}{\delta_{ij}}\frac{\partial \delta_{ij}}{\partial \beta_p} + \frac{(y_{ij} - \sum_{k=j}^{K} y_{ik})}{1-\delta_{ij}}\frac{\partial \delta_{ij}}{\partial \beta_p} \\ &= \sum_{j=1}^{K-1}\sum_{i=1}^{n} x_{ip}\Big(y_{ij}\frac{\phi(\alpha_j + x_i^T\boldsymbol{\beta})}{\Phi(\alpha_j + x_i^T\boldsymbol{\beta})} + (y_{ij} - \sum_{k=j}^{K} y_{ik})\frac{\phi(\alpha_j + x_i^T\boldsymbol{\beta})}{1-\Phi(\alpha_j + x_i^T\boldsymbol{\beta})}\Big) \\ &= \sum_{j=1}^{K-1} \boldsymbol{x}_p^T\Big(\boldsymbol{y}_j\frac{\phi(\alpha_j + \boldsymbol{x^T\beta})}{\Phi(\alpha_j + \boldsymbol{x^T\beta})} + (\boldsymbol{y}_j - \sum_{k=j}^{K} \boldsymbol{y}_k)\frac{\phi(\alpha_j + \boldsymbol{x^T\beta})}{1-\Phi(\alpha_j + \boldsymbol{x^T\beta})}\Big) \qquad (2.9)\end{aligned}$$

In the GMIFS algorithm, the α terms are initialized considering all slope terms in equation 2.8 to be 0, where $P(Y_i = j|Y_i \geq j, \boldsymbol{x}_i) = \Phi(\alpha_j)$

$P(Y_i = j|Y_i \geq j, \boldsymbol{x}_i) = \Phi(\alpha_j)$, such that

$$\alpha_j = \Phi^{-1}\Big(\frac{\sum_{i=1}^{n} y_{ij}}{\sum_{i=1}^{n}\sum_{k=j}^{K} y_{ik}}\Big).$$

The code that fits the penalized continuation ratio models using the probit link appears in the Appendix A.2.

2.2.3 Example using cumulative probit model

Breast cancer (BC) is the second most common cancer among women. Research shows that many women with BC experience anxiety, depression, and stress (ADS). A potential mechanism for the development of ADS is epigenetics, such as methylation [Zhou et al., 2015]. In this example, we demonstrate the application of our penalized cumulative probit model to predict severity of psychoneurological symptoms, specifically, ADS levels, using a methylation data assayed using the Illumina Human

Methylation 450K assay. The original dataset contains ADS scales and methylation levels on 485,512 CpG sites. The β-values were defined as the proportion methylated, and CpG sites with all β-values over 0.9 or below 0.1 were filtered out [Zhou et al., 2015]. For each CpG site, β values were plotted against GC content across all subjects. Based on the results, CpG sites with a GC content greater than 40% were also filtered out, left 285,173 CpG sites [Zhou et al., 2015].

Severity of depression and anxiety was measured using the Hospital Anxiety and Depression Scale (HADS), which is a 14-item questionnaire [Zigmond and Snaith, 1983]. Among the 14-items, 7 assess anxiety and 7 assess depression. Each item is on a four level ordinal scale, the scale response is calculated such that the ordinal levels contribute 0-3 points, and the seven items within each scale are summed. Using these sums, subjects are classified into three ordinal levels: normal (score < 8), borderline $(8 - 10)$, or having clinical anxiety or depression (score > 10) [Lambert et al., 2013]. Stress was measured using the 10-item Perceived Stress Scale (PSS). Ten scores were summed, with higher total scores of PSS indicating higher overall stress [Cohen and Williamson, 1988]. Subjects were categorized into four quantiles, where the category 1 has lowest stress, and the category four has highest stress. Based on described categorization, our ordinal response variables are three ADS categories, where anxiety has three levels: normal (n=30), borderline (n=25), and anxiety (n=18); depression has three levels: normal (n=66), borderline (n=4), and depression (n=3); and stress has four levels: I (n=19), II (n=18), III (n=18), and IV (n=18).

Since ADS categories are based on survey scores, assumed to have underlying normal distributions, we used the cumulative probit model to fit the data. Separate cumulative probit models were fit for each ADS scales (anxiety, depression and stress) using the `ordinal.gmifs` function in the *ordinalgmifs* library. The increment, ϵ for each update step was set to be 0.01, and all covariates were standardized to reduce correlations between covariates.

When fitting separate models for each ADS category based on the minimum AIC, among 285,173 CpG sites, 67 CpG sites were significantly associated with anxiety; 19 CpG sites were significantly associated with depression; and 10 CpG sites were significantly associated with stress. Significant CpG sites for each ADS scales are listed in Appendix A.3. We also examined the prediction accuracy of each ADS model by assessing misclassification rate. Anxiety and depression models have no misclassifications; however it may be due to the model overfitting (the number of significant covariates in anxiety and depression models = 67 and 19, respectively). The stress model had a misclassification rate of 0.34.

To avoid the problem of overfitting, we also examined models attaining the minimum BIC. The anxiety model based the minimum BIC only selected 1 significant CpG site, and is not adequate to predict the response. The depression model based on the minimum BIC selected seven significant CpG sites and the corresponding misclassification rate was 0.10. Stress models based on the the minimum BIC selected 10 significant CpG sites, and the misclassification rate was 0.34. All these significant

CpG sites are also listed in Appendix A.3. In conclusion, both AIC and BIC selected anxiety models did not have good prediction accuracy. The BIC selected depression and stress models have similar prediction accuracy compared to their corresponding AIC models but are more parsimonious. However, misclassfication rates were estimated using the training dataset, and therefore, are not good indicators for future performance. A better approach would be to estimate generalization error (i.e., cross-validation error). Since the penalized cumulative probit model took long time to converge due to the high-dimensionality ($P = 285,173$), in the future, we plan to implement parallel processing to speed up our cross-validation computations. The code for performing this example appears in Appendix A.1.

2.3 Penalized stereotype logit model

The GMIFS procedure was also extended to the stereotype logit model described in Chapter 1 (equation (1.9)). The GMIFS procedure for the stereotype logit model and for probit-link models are similar except that the algorithm for the stereotype logit model updates not only $\boldsymbol{\alpha}$ but also $\boldsymbol{\phi}$ during each iteration.

From the definition of the stereotype logit model (equation 1.6), we can rewrite the formula in terms of response probabilities:

$$\pi_j(\boldsymbol{x}_i) = \frac{\exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})}{\sum_{k=1}^{K} \exp(\alpha_k + \phi_k \boldsymbol{x}_i^T \boldsymbol{\beta})} \tag{2.10}$$

where the intercept terms $\boldsymbol{\alpha}$ and scale parameters $\boldsymbol{\phi}$ are both constrained as:

$$\alpha_1 \leq \alpha_2 \leq ... \leq \alpha_K - 1 \leq \alpha_K$$

and

$$\phi_1 \geq \phi_2 \geq ... \geq \phi_K$$

For convenience, we define $\alpha_K = 0$ and $\phi_K = 0$. Then, the expression simplifies to

$$\pi_K(\boldsymbol{x}_i) = \frac{\exp(0)}{\sum_{k=1}^{K} \exp(\alpha_k + \phi_k \boldsymbol{x}_i^T \boldsymbol{\beta})} = \frac{1}{\sum_{j=1}^{K-1} \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})} \tag{2.11}$$

when $j = K$.

Again, the GMIFS algorithm requires the log-likelihood for the model and the partial derivative with respect to β_p to find the solution numerically. Based on equation (1.13), the log-likelihood of the model can be rewritten as a summation of individual log-likelihoods

$$\log L = \sum_{i=1}^{n} \big(\sum_{j=1}^{K-1} y_{ij} \log(\pi_j(\boldsymbol{x}_i)) + y_{iK} \log(\pi_K(\boldsymbol{x}_i)) \big). \tag{2.12}$$

The first derivative of the log-likelihood with respect to β_p is given by

$$\frac{\partial logL}{\partial \beta_p} = \sum_{i=1}^{n} \big(\sum_{j=1}^{K-1} y_{ij} \frac{\partial \log(\pi_j(\boldsymbol{x}_i))}{\partial \beta_p} + \sum_{i=1}^{n} y_{iK} \frac{\partial \log(\pi_K(\boldsymbol{x}_i))}{\partial \beta_p} \big) \tag{2.13}$$

given the simplification from equation (2.11). For $j = K$, we have

$$\frac{\partial \log(\pi_K(\boldsymbol{x}_i))}{\partial \beta_p} = \frac{\partial(\log(1) - \log(1 + \sum_{j=1}^{K-1} \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})))}{\partial \beta_p}$$
$$= -x_{ip} \frac{\sum_{j=1}^{K-1} \phi_j \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})}{1 + \sum_{j=1}^{K-1} \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})}.$$

and because of the equation 2.10, for $j = 1, ..., K-1$, we have

$$\frac{\partial \log(\pi_j(\boldsymbol{x}_i))}{\partial \beta_p} = \frac{\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta} - \log(1 + \sum_{j=1}^{K-1} \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})}{\partial \beta_p}$$
$$= \phi_j x_{ip} - \frac{\sum_{j=1}^{K-1} \phi_j \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})}{1 + \sum_{j=1}^{K-1} \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})} x_{ip}$$
$$= x_{ip}\big(\phi_j - \frac{\sum_{j=1}^{K-1} \phi_j \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})}{1 + \sum_{j=1}^{K-1} \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})}\big).$$

Substituting the above two equations back into equation 2.13, we have

$$\frac{\partial logL}{\partial \beta_p} = \sum_{i=1}^{n} x_{ip} \Bigg(\sum_{j=1}^{K-1} y_{ij} \Big(\phi_j - \frac{\sum_{j=1}^{K-1} \phi_j \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})}{1 + \sum_{j=1}^{K-1} \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})} \Big) - y_{iK} \frac{\sum_{j=1}^{K-1} \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})}{1 + \sum_{j=1}^{K-1} \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})} \Bigg), \quad (2.14)$$

which can be further rewritten as

$$\frac{\partial logL}{\partial \beta_p} = \boldsymbol{x}_p^T \Bigg(\sum_{j=1}^{K-1} \boldsymbol{y}_j \Big(\phi_j - \frac{\sum_{j=1}^{K-1} \phi_j \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})}{1 + \sum_{j=1}^{K-1} \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta})} \Big) - \boldsymbol{y}_K \frac{\log(1 + \sum_{j=1}^{K-1} \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta}))}{1 + \log(1 + \sum_{j=1}^{K-1} \exp(\alpha_j + \phi_j \boldsymbol{x}_i^T \boldsymbol{\beta}))} \Bigg). \quad (2.15)$$

For K ordinal classes and P predictors, the GMIFS algorithm for the stereotype logit model is

1. Standardize the predictors then expand the covariate matrix $\boldsymbol{X}$ to $\tilde{\boldsymbol{X}} = [\boldsymbol{X} : -\boldsymbol{X}]$.

2. Initialize $\boldsymbol{\alpha}$ and $\boldsymbol{\phi}$.

3. Initialize the components of $\hat{\boldsymbol{\beta}}^{(s)}$ at step s=0 to be all 0.

4. Find $m = \text{argmin}_p(-\partial logL/\partial\beta_p)$ at the current estimate $\hat{\boldsymbol{\beta}}^s$.

5. Update $\hat{\boldsymbol{\beta}}_m^{s+1} \rightarrow \hat{\boldsymbol{\beta}}_m^s + \epsilon$.

6. Consider $\hat{\boldsymbol{\beta}}^{s+1}$ as fixed, update $\boldsymbol{\alpha}_s$ and $\boldsymbol{\phi}$ using maximum likelihood method.

7. Repeat steps 4 - 6 until $logL^{(s+1)} - logL^{(s)} < \tau$.

The α terms are initialized considering all slope terms in equation 1.9 to be 0, or $\log(\frac{\pi_j(\boldsymbol{x}_i)}{\pi_K(\boldsymbol{x}_i)}) = \Phi(\alpha_j)$,
such that,

$$\alpha_j = \log(\frac{\sum_{i=1}^{n} y_{ij}}{\sum_{i=1}^{n} y_{iK}}).$$

while $\boldsymbol{\phi}$ are initialized as $\phi_1 = 1$, $\phi_2, ..., \phi_{K-1} = 0.1$ [Agresti, 2010]. The constraints on $\boldsymbol{\alpha}$ and $\boldsymbol{\phi}$ were insured by box-constraints.

2.3.1 Example of penalized stereotype logit model

Alzheimer's disease (AD) has been intensely studied during the last 10 years. Genomic microarray provides new tools in understanding the underlying pathological mechanism in AD. In this section, we illustrate the utility of the steretype logit model using a high-throughput genomic dataset. The goal is to predict the severity of Alzheimer's disease (AD) using microarray gene expression data assayed by Affymetrix HG-U133A GeneChips. The full dataset, GSE1297 was downloaded from

Gene Expression Omnibus [Blalock et al., 2004]. A total of 35 subjects were categorized into four groups based on the MiniMental Status Examination (MMSE) criteria: control AD (MMSE > 25), incipient AD (MMSE $20 - 26$), moderate AD (MMSE $14 - 19$) and severe AD (MMSE < 14). Four subjects with MMSE < 20 were removed from the study because they were potentially affected by confounding conditions. The remain 31 subjects fell in the four levels of AD: control ($n = 9$), incipient ($n = 7$), moderate ($n = 8$) and severe ($n = 7$). After microarray preprocessing, probe set level data was summarized using the MAS5 method. Control probes sets and probe sets absent in all 31 samples were removed, leaving 15,189 probe sets.

Before performing the analysis, we tested the proportional odds assumption using a score test in a univariate ordered logistic model for each probe set. Since the test indicated that 9.6% of the probe sets do not have the same slope coefficient across all levels of the response, modeling the data using the penalized stereotype logit model is a reasonable approach.

After fitting the GMIFS stereotype logit model, with $\epsilon = 0.001$, the penalized stereotype logit model selected by the minimum AIC only select three predictors: 203643_at, 206278_at and 212122_at and the misclassification rate was 45%.

The code for fitting GMIFS stereotype logit models and performing Alzheimer's disease example appears in Appendix A.4.

Chapter 3

Penalized Bayesian Cumulative logit model

Park and Casella (2008) proposed their Bayesian LASSO for linear regression using a double exponential prior (equation (1.2)). The hyperparameter λ in the density function of the double exponential prior determines the total amount of shrinkage and can be selected in several ways. Park and Casella [Park and Casella, 2008], together with other authors [Hans, 2009, Lykou and Ntzoufras, 2013], considered selecting λ by assigning it to an independent gamma prior distribution. The shrinkage property of the LASSO make it a popular variable selection method under the frequentist framework. Under the Bayesian framework, Park and Casella suggested that the Bayesian credible intervals could be used to guide variable selection; however, Lykou and Ntzoufra (2013) argued that the selection based on posterior credible intervals depends both on the selection of the posterior probability attached to such inter-

vals and the way that they are constructed, and does not take into account model uncertainty. Lykou and Ntzoufra (2008) then proposed to use the binary inclusion indicators method for feature selection [Dellaportas et al., 2002, George and McCulloch, 1993, Kuo and Mallick, 1998].

Motivated by these previous developments with respect to the Bayesian LASSO, here we aimed to extend the Bayesian LASSO to an ordinal regression model, specifically, the cumulative logit model. We present a method for choosing λ by giving it a hyperprior, and utilize the binary variable inclusion indicator to perform feature selection. Our method for implementing the Bayesian LASSO cumulative logit model is described in section 3.1. In sections 3.2 and 3.3, the utility of our method is illustrated using both simulated data and a high-throughput genomic dataset. In our simulation study, we compare our penalized ordinal Bayesian model using different priors to a penalized cumulative logit model using a frequentist approach (generalized monotone incremental forward stage-wise method) in term of their abilities to predict the ordinal response and to correctly incorporate true predictors from noise predictors into the model. We will also demonstrate application of our method to predict stage of liver disease (normal, cirrhotic but without hepatocellular carcinoma, hepatocellular carcinoma) using methylation data assayed by the Illumina GoldenGate Methylation BeadArray Cancer Panel I.

3.1 Penalized Bayesian LASSO cumulative logit model

3.1.1 Bayesian cumulative logit model

Let Y_i represent the ordinal response for subject i where Y_i can fall into one of K categories ($k = 1, 2, ...K$). Let $\boldsymbol{x}_i$ denote a $P \times 1$ covariate vector for subject i. Let $\alpha_1, ..., \alpha_{K-1}$ represent the $K-1$ intercept terms in the cumulative logit model where $\alpha_1 \leq \alpha_2 \leq ... \leq \alpha_{K-1}$. Let $\pi_k(\boldsymbol{x}_i)$ represent the probability that subject i falls into the k^{th} category. The non-penalized, Bayesian cumulative logit model can be expressed as

$$
\begin{aligned}
P(Y_i \leq k|\boldsymbol{x_i}) &= \frac{\exp(\alpha_k + \boldsymbol{x}_i^T\boldsymbol{\beta})}{1 + \exp(\alpha_k + \boldsymbol{x}_i^T\boldsymbol{\beta})} \\
\pi_k(\boldsymbol{x}_i) &= P(Y_i \leq k|\boldsymbol{x_i}) - P(Y_i \leq k-1|\boldsymbol{x_i}) \\
Y_i &\sim Cat(\pi_1, ..., \pi_K) \\
\alpha_1 &\sim N(0, 1000) \\
\alpha_k \sim N(0, 1000) \;\; \alpha_k \in (\alpha_{k-1}, \infty) &\qquad \text{for } k = 2, ..., K-1 \\
\beta_j &\sim N(0, 1000) \qquad \text{for } j = 1, ..., P
\end{aligned}
\tag{3.1}
$$

Here, the response variable Y_i follows a categorical distribution, which contains a vector of parameters where each of the K parameters represent the probability an outcome falls into that response category. The probability mass function (PMF) for the categorical distribution is $P(Y_i = i) = \pi_i$. To respect the ordering constraint of

the intercepts, we used a truncated normal prior distribution for $\boldsymbol{\alpha}$, where the k^{th} intercept α_k will have a lower boundary α_{k-1}. To reflect a lack of prior information, the intercepts have a mean of 0 and a large, diffuse standard deviation of 1000. We have a similar prior setting for slope coefficients, where the j^{th} slope β_j $(j = 1, 2, ...P)$ has a normal prior with a mean of 0, and a standard deviation of 1000. Since both intercepts and slopes have non-informative priors, theoretically, we would expect the mode/mean/median of the posterior distribution to be similar to the ML estimates. Other than assuming the normal distribution for the $K - 1$ intercepts, we could also use a gamma distribution for the difference between two adjacent intercepts.

3.1.2 Bayesian LASSO cumulative logit model

In this section, we modify Equation 3.1 to construct our penalized Bayesian cumulative logit model. Inspired by Park and Casella's Bayesian LASSO for linear regression, here we impose shrinkage by giving a double exponential prior to each of the slope coefficients β_j $(j = 1, 2, ...P)$:

$$\pi(\beta_j|\sigma^2) = \frac{\lambda}{2\tau} e^{-\lambda\tau^{-1}|\beta_j|},$$

where λ controls the amount of shrinkage and τ is the standard deviation of the response variable. In our method, the prior for τ is taken to be relatively diffuse, non-informative gamma distribution with a shape of 0.001, and a scale of 0.001. There are many ways to select λ, here we chose λ by giving it a gamma hyperprior,

$\lambda \sim Gamma(a, b)$ with a mean a/b and a variance a/b^2. Although it is attractive to assign a and b small values so the prior is non-informative, in reality, λ cannot be too large or too small. Lykou and Ntzoufra (2013) argued that a large λ can force the posterior distributions of the coefficients to be close to 0; therefore, the data (in comparison to the prior) are not strong enough to provide evidence for a significant coefficient. They further argued that when the λ is too small, the posterior coefficient shrinks back to zero instead of the MLE estimates due the Lindley-Barlett paradox (Lindley-Barlett paradox states that small values of λ lead to posterior model odds that fully support the most parsimonious model which shrink coefficients to zero) [Lykou and Ntzoufras, 2013]. Therefore, as a reasonable start, we assign λ with a $Gamma(1, 1)$ prior distribution. The sensitivity analysis will be conducted in the second simulation study in section 3.2, where we compare models with different λ priors in term of their feature selection accuracy.

Although Tibshirani suggested that LASSO estimates can be interpreted as posterior mode estimates when the regression parameters have independent and identical Laplace priors [Tibshirani, 1996], under the Bayesian framework, we often use the posterior means and medians as points estimates. Since the Bayesian posterior mean/medians lose the ability to shrink coefficients to exactly zero as the frequentist LASSO estimates or the posterior modes, we then utilize the binary variable inclusion indicator method first proposed by George and McCulloch [George and McCulloch, 1993] to enable feature selection. Specifically, for every β_j, we introduce a Bernoulli indicator γ_j, where $\gamma_j = 1$ indicates the j^{th} covariate is selected into the

model and $\gamma_j = 0$ otherwise. George and McCulloch (1993) suggested a Bernoulli prior distribution for γ_j with success probability of 0.5 ($p_{succ} = 0.5$). This can be interpreted as one half of the features will be selected into the model without knowing any data and therefore, can be considered as a non-informative prior. Combining β_j and γ_j together, now our slope coefficient will be denoted as $\beta_j^{(\gamma)} = \beta_j \gamma_j$ for the j^{th} covariate. Similar to equation (3.1), the intercept terms $\boldsymbol{\alpha}$ in the penalized Bayesian logit cumulative model follow truncated normal distributions. For a non-informative prior, the normal distribution can have a very large standard deviation, for example, 1000. Additionally, since it would be strange for an ordinal class to have less than 0.1% of the observations, we further assume the $\boldsymbol{\alpha}$ terms follow a truncated normal with a lower bound of $logit(0.001) = -6.9$ and upper bound of $logit(0.999) = 6.9$. This alleviates obtaining really wildly large values for $\boldsymbol{\alpha}$ terms, speeds up the MCMC convergence, but is still non-informative.

Our penalized Bayesian cumulative logit model is

$$
\begin{aligned}
P(Y_i \leq k|\boldsymbol{x_i}) &= \frac{\exp(\alpha_k + \boldsymbol{x}_i^T\boldsymbol{\beta}^{(\gamma)})}{1+\exp(\alpha_k + \boldsymbol{x}_i^T\boldsymbol{\beta}^{(\gamma)})} \\
\pi_k(\boldsymbol{x}_i) &= P(Y_i \leq k|\boldsymbol{x_i}) - P(Y_i \leq k-1|\boldsymbol{x_i}) \\
Y_i &\sim Cat(\pi_1, ..., \pi_K) \\
\alpha_1 &\sim N(0, 1000) \;\; \alpha_1 \in (-6.9, 6.9) \\
\alpha_k &\sim N(0, 1000) \;\; \alpha_k \in (\alpha_{k-1}, 6.9) \qquad \text{for } k = 2, ..., K-1 \\
\boldsymbol{\beta}^{(\gamma)} &= (\beta_1^{(\gamma)}, \beta_2^{(\gamma)}, ..., \beta_P^{(\gamma)}) \\
\beta_j^{(\gamma)} &= \gamma_j \beta_j \\
\text{For } j = 1, 2, ..., P., \;\; \beta_j &\sim DE(0, \frac{1}{\tau\lambda}) \\
\gamma_j &\sim Bernoulli(p_{succ}) \\
\lambda &\sim Gamma(a, b) \\
\tau &\sim Gamma(0.001, 0.001). \qquad (3.2)
\end{aligned}
$$

Although we have used non-informative truncated normal priors for intercept terms, $\boldsymbol{\alpha}$, we could also consider more informative priors. Agresti (2010) stated that when there are many parameters, the posterior mode need not then necessarily be close to the ML estimate, and Markov chains may converge slowly. Therefore, it is usually more sensible to construct a prior distribution that represents careful expression of our prior beliefs about the parameter values. For example, instead of using a very large standard deviation for a normal prior distribution, use a mean and standard deviation such that the range within three standard deviations of the mean

contains all plausible values the parameter could take[Agresti, 2010]. In our penalized Bayesian cumulative logit model, we can also assign $\boldsymbol{\alpha}$ with more informative priors, where

$$\begin{aligned} \alpha_1 &\sim N(a_{0,1}, 1000) \;\; \alpha_1 \in (-6.9, 6.9) \\ \alpha_k - \alpha_{k-1} &\sim Gamma(a_{0,k} - a_{0,k-1}, 1) \qquad \text{for } k = 2, ..., K. \end{aligned} \tag{3.3}$$

$a_{0,1}$, $a_{0,k}$, and $a_{0,k-1}$ can be any reasonable numbers for means of the intercepts. For example, they can be initial values for the 1^{st}, k^{th} and $(k-1)^{th}$ intercepts, respectively, in the GMIFS method. In the GMIFS method, we set the initial values of the intercepts to $a_j = logit(\sum_{i=1}^{n} \sum_{k=1}^{j} y_{ik}/n)$, for the cumulative logit model, which is equivalent to the intercepts in a null cumulative logit model. These priors reflect a belief that the sizes of the effects are not extremely strong and the difference between two adjacent intercepts has a mean of $a_{0,k} - a_{0,k-1}$, and a variance of $a_{0,k} - a_{0,k-1}$. We compare our penalized Bayesian cumulative logit model using both informative priors (equation 3.3) and non-informative priors (equation 3.2) to a penalized cumulative logit model using our GMIFS method in terms of their ability to accurately select features in simulation study II, section 3.3.

Due to the complexity of double exponential priors, it is less straightforward to find the closed-form expression for the posterior distribution [Lykou and Ntzoufras, 2013]. Therefore, we approximate the posterior distribution using Markov Chain Monte Carlo (MCMC), specifically, Gibbs sampling. Gibbs sampling was carried

out using the R package *R2jags*. After Gibbs sampling, the slope coefficients were estimated using the posterior means of $\boldsymbol{\beta}^{(r)}$, and the intercepts were estimated using the posterior means of $\boldsymbol{\alpha}$. The posterior means of $\boldsymbol{\gamma}$ can be interpreted as the posterior inclusion probabilities. The covariates with high posterior inclusion probabilities will then be selected and the ones with low or zero posterior inclusion probabilities can be ignored. In the simulation study of Kykou and Ntzoufras (2013), Kykou and Ntzoufras selected important predictors as those for which the posterior inclusion probabilities were greater than 0.5. However, in a real genomic data set the sample size is often quite small compared to the number of predictors, such that there may be no high posterior means of $\boldsymbol{\gamma}$ [George and McCulloch, 1993].

The MCMC convergence will be confirmed visually via traceplot when the number of parameters is small and examined statistically by the Gelman-Rubin test. The Gelman-Rubin test calculates within-chain and between-chain variance, and then estimates the variance of the parameter as a weighted sum of the within-chain and between-chain variance. After that, it calculates the potential scale reduction factor $\hat{R}$, which indicates non-convergence when $\hat{R}$ is greater than 1.1. To illustrate the test with formulas, suppose we have m parallel chains ($m \geq 2$) and each chain has a length n. s_{ij} is the parameter of interest at the i^{th} iteration for the j^{th} chain. Then

$\hat{R}$ can be calculated using the following equations:

$$B = \frac{n}{m}\sum_{j=1}^{m}(\bar{s}_{.j} - \bar{s}_{..})^2$$

$$\text{where,} \quad \bar{s}_{.j} = \frac{1}{n}\sum_{i=1}^{n} s_{ij} \quad \texttt{and} \quad \bar{s}_{..} = \frac{1}{m}\bar{s}_{.j}$$

$$W = \frac{1}{m}\sum_{j=1}^{m} s_j^2 \;\text{ where, }\; s_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(s_{ij} - \bar{s}_{.j})^2$$

$$\hat{var}^{+} = \frac{n-1}{n}W + \frac{1}{n}B$$

$$\hat{R} = \sqrt{\hat{var}^{+}/W}$$

In our method, we used three Markov chains, and we considered MCMC to have converged when $\hat{R} \leq 1.1$.

3.1.3 Prediction

In Bayesian analysis, predictions of future observations are based on the posterior predictive distributions. When future observations are $\tilde{\boldsymbol{Y}}$, and the posterior distribution for the modeling parameters $\boldsymbol{\theta}$ is $\pi(\boldsymbol{\theta}|\boldsymbol{Y})$, the posterior predictive distribution is

$$f(\tilde{\boldsymbol{Y}}) = \int f(\tilde{\boldsymbol{Y}}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}|\boldsymbol{Y})\delta\boldsymbol{\theta}.$$

A merit of Bayesian analysis using MCMC is that future observations $\tilde{\boldsymbol{Y}}$ can be viewed as additional parameters under estimation, and therefore, estimated directly from an MCMC sampler. In fact, to predict ordinal responses using the same data used for training the model, one can simply generate replicated responses, $\boldsymbol{Y}^{rep} = \tilde{\boldsymbol{Y}}$

from the posterior predictive distribution by adding a single step within any MCMC sampler using the likelihood function $f(\boldsymbol{Y}^{rep}|\boldsymbol{\theta}^{(t)})$ evaluated at parameter values $\boldsymbol{\theta}^{(t)}$ of the current stage of the algorithm [Ntzoufras, 2008]. In our penalized Bayesian model, we can generate the predicted values by adding the below step

$$Y_i^{rep} \sim Cat(\pi_1, ..., \pi_K).$$

If we want to generate a predicted response Y_{n+1} from a vector of new data $\boldsymbol{x}_{n+1}$, then the below steps can be added to a cumulative logit model.

$$P(Y_{n+1} \leq k|\boldsymbol{x}_{n+1}) = \frac{\exp(\alpha_k + \boldsymbol{x}_{n+1}^T\boldsymbol{\beta})}{1 + \exp(\alpha_k + \boldsymbol{x}_{n+1}^T\boldsymbol{\beta})}$$

$$\pi_k(\boldsymbol{x}_{n+1}) = P(Y_i \leq k|\boldsymbol{x_{n+1}}) - P(Y_i \leq k-1|\boldsymbol{x_{n+1}})$$

$$Y_{n+1} \sim Cat(\pi_1, ..., \pi_K)$$

Thus, a penalized Bayesian cumulative logit model with prediction embedded is

$$
\begin{aligned}
P(Y_i \leq k|\boldsymbol{x_i}) &= \frac{\exp(\alpha_k + \boldsymbol{x}_i^T\boldsymbol{\beta})}{1+\exp(\alpha_k + \boldsymbol{x}_i^T\boldsymbol{\beta})} \\
\pi_k(\boldsymbol{x}_i) &= P(Y_i \leq k|\boldsymbol{x_i}) - P(Y_i \leq k-1|\boldsymbol{x_i}) \\
Y_i &\sim Cat(\pi_1, ..., \pi_K) \\
\alpha_1 &\sim N(0, 1000) \\
\alpha_k \sim trunN(0, 1000)\ \alpha_k \in (\alpha_{k-1}, \infty) &\qquad \text{for } k = 2, ..., K-1 \\
\beta_j \sim DE(0, \frac{1}{\tau\lambda}) &\qquad \text{for } j = 1, ..., P \\
\tau &\sim Gamma(a, b) \\
\lambda &\sim Gamma(0.001, 0.001) \\
P(Y_{n+1} \leq k|\boldsymbol{x}_{n+1}) &= \frac{\exp(\alpha_k + \boldsymbol{x}_{n+1}^T\boldsymbol{\beta})}{1+\exp(\alpha_k + \boldsymbol{x}_{n+1}^T\boldsymbol{\beta})} \\
\pi_k(\boldsymbol{x}_{n+1}) &= P(Y_i \leq k|\boldsymbol{x_{n+1}}) - P(Y_i \leq k-1|\boldsymbol{x_{n+1}}) \\
Y_{n+1} &\sim Cat(\pi_1, ..., \pi_K). \qquad (3.4)
\end{aligned}
$$

Note that the above model uses non-informative prior distributions for the $\boldsymbol{\alpha}$ terms, though they can be changed to informative priors to speed up the MCMC process. In addition, when the number of covariates is relatively small, $\boldsymbol{\beta}$ can be assigned diffuse normal priors. In this way, we will have a non-penalized regular cumulative logit model for prediction. In section 3.3, we demonstrate how our method can be used to predict stage of liver disease using features from a high-throughput methylation array.

3.2 Simulation Study

In this section, we present results from two simulation studies. In subsection 3.2.1, we demonstrate our Bayesian cumulative logit model using a simulated non-high dimensional dataset. The estimated coefficients will be compared to a penalized cumulative logit model using a frequentist approach (GMIFS) and a non-penalized cumulative logit model. In subsection 3.2.2, we compare our penalized Bayesian cumulative logit model using different priors to GMIFS in terms of their abilities to predict the ordinal response and to correctly incorporate true predictors from noise predictors into the model when feature space is high-dimensional. The code for performing these two simulation studies appears in Appendix B.1. The model files that are necessary for running JAGS appear in Appendix B.3.

3.2.1 Simulation study I

The main objective of this simulation study is to illustrate our method and compare it to existing methods. Ordinal responses were simulated according to the cumulative logit model. Five covariates ($P = 5$) were generated from independent standard normal distributions. We then generated a three level ordinal response variable ($K = 3$) for 200 individuals ($n = 200$). The values for the $\boldsymbol{\alpha}$ terms and $\boldsymbol{\beta}$ terms were chosen by trial and error to ensure approximately equal outcome frequencies in the individual categories. The simulation was performed according to the following steps:

1. Randomly generate 5 variables, $\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_5}$, each follows a standard normal

distribution.

2. Let the first two predictors, $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ be associated with the outcome so that $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ are important covariates while the rest of the covariates are non-important.

3. Assign $\alpha_1 = -1, \alpha_2 = 2, \beta_1 = 3, \beta_2 = 1$.

4. Generate $P(Y_i \leq 1)$ and $P(Y_i \leq 2)$ according to the cumulative logit model, Specifically, let
   $P(Y_i \leq 1) = \alpha_1 + \beta_1 x_{i1} + \beta_2 x_{i2}$ and
   $P(Y_i \leq 2) = \alpha_2 + \beta_1 x_{i1} + \beta_2 x_{i2}$.

5. Randomly generate a variable T where $T \sim \mathit{Unif}(0, 1)$.

6. If $T \leq P(Y_i \leq 1)$, then assign $Y_i = 1$; if $P(Y_i \leq 1) < T \leq P(Y_i \leq 2)$, then assign $Y_i = 2$; otherwise, assign $Y_i = 3$.

7. Repeat steps 4-6 for the remaining $n - 1$ observations.

Here, we present the coefficients estimated by three different cumulative logit models. The regular non-penalized cumulative logit model (VGLM) was fit using the `vglm` function in the *VGAM* package. The LASSO estimates were obtained by the generalized monotone incremental forward stage-wise (GMIFS) method using the `ordinal.gmifs` function in the *ordinalgmifs* R package and the final model was selected based on the minimum AIC. For the Bayesian analysis, we used non-informative truncated normal prior distributions for the two intercepts, α_1 and α_2 (equation (3.2)).

The analysis was implemented using the Bayesian software JAGS using an interface through R, implemented in the *R2jags* package. The posterior estimation results were based on three MCMC chains each using 10,000 iterations followed by 5000 iterations auto-updating until the model converged. The first 10,000 iterations were discarded as burn-in and the last 2500 iterations from the updating step were saved with a thinning rate of 2 (number of iterations saved = number of iterations / thinning rate). The initial values for the three chains are shown in the Table 3.1.

Table 3.1: Bayesian initiation table

	α_1	α_2	$\boldsymbol{\beta}$
Chain 1	$logit(\sum_{i=1}^{n} y_{i1}/n)$	$logit(\sum_{i=1}^{n}\sum_{j=1}^{2} y_{ij}/n)$	$\mathbf{0}$
Chain 2	-1	1	$\sim N(0, 0.01^2)$
Chain 3	-2	2	$\sim N(0, 0.001^2)$

The Trace Plot of the last 2500 iterations versus sampled values for each parameter was used to assess convergence. Based on the plots, the three chains mixed well and converged to their stationary distributions (figs. 3.1 to 3.3). The Gelman-Rubin tests supported this conclusion as $\hat{R}$ ranged from 1 to 1.03 for all parameters.

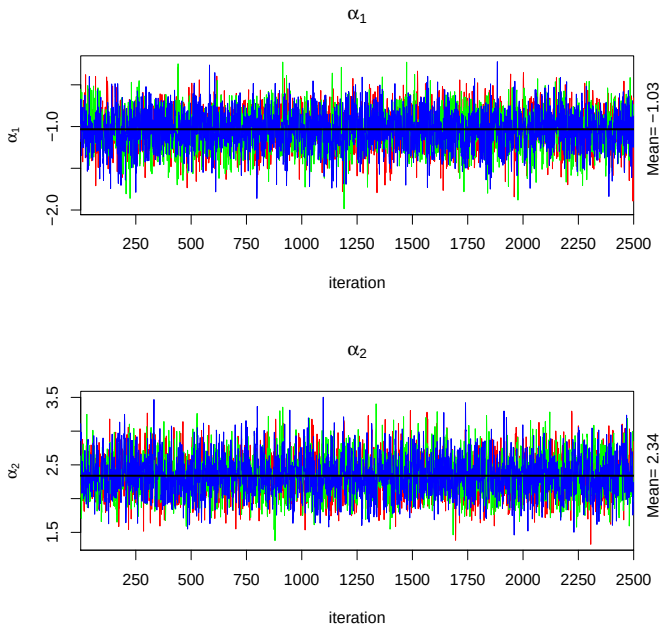

Figure 3.1: Traceplot of the two $\boldsymbol{\alpha}$ terms. The trajectory of our chains is consistent over time, with a relatively constant mean and variance indicating good convergence.

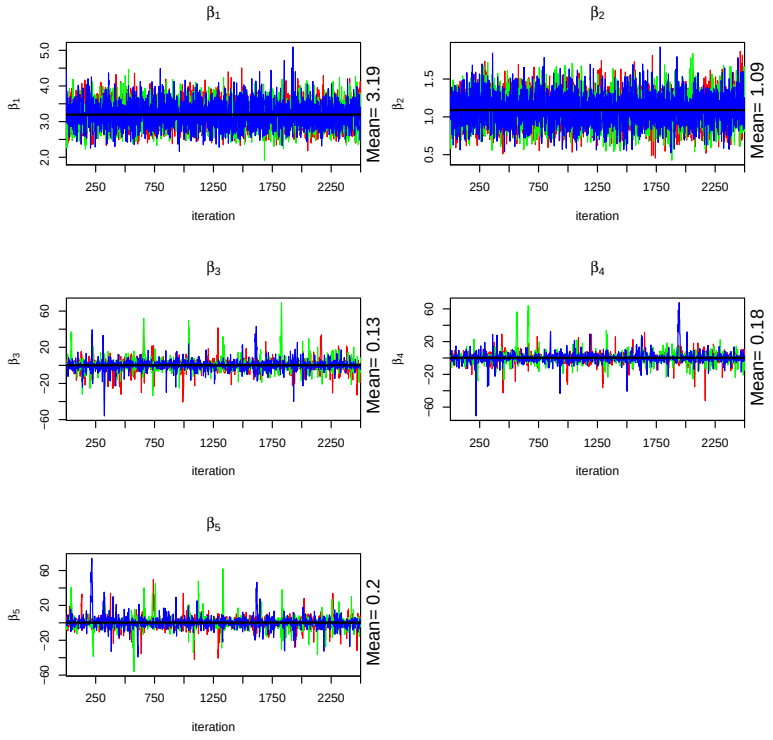

Figure 3.2: Traceplot of the five $\boldsymbol{\beta}$ terms. β_1 and β_2 have a relatively constant mean and variance indicating good convergence. β_3, β_4, and β_5 have means at 0, the high variances are due to random error.

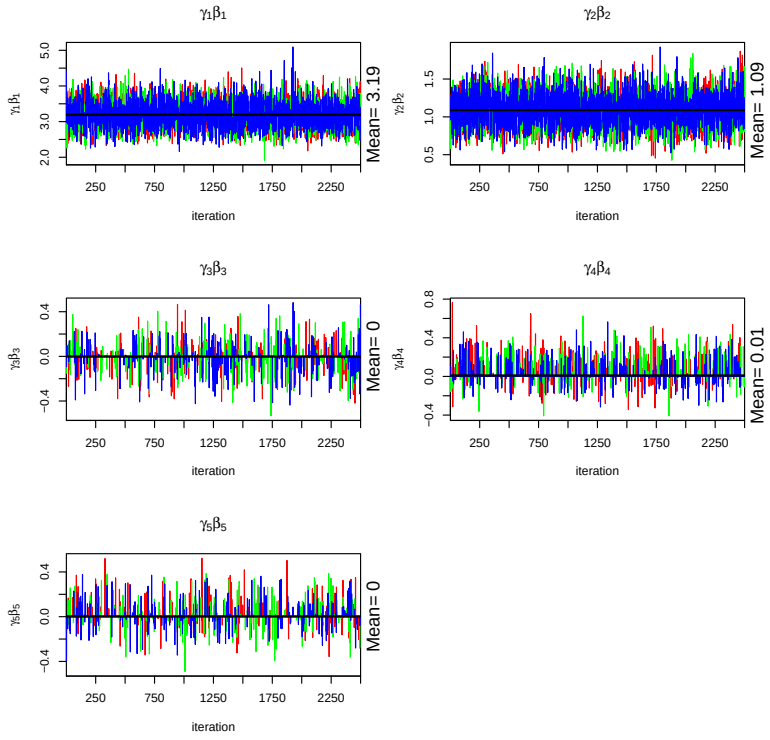

Figure 3.3: Traceplot of the five $\boldsymbol{\gamma\beta}$ terms. $\beta_1\gamma_1$ and $\beta_2\gamma_2$ have a relatively constant mean and variance indicating good convergence. $\beta_3\gamma_3$, $\beta_4\gamma_4$, and $\beta_5\gamma_5$ have means at 0, the high variances are due to random error.

Table 3.2 shows the results from the three different cumulative logit models (VGLM, LASSO, and Bayesian LASSO). In terms of parameter estimation (α_1, α_2, β_1 and β_2), the three analyses produced similar estimates and were close to the true underlying values. Although the penalized Bayesian cumulative logit model was not able to shrink some of the non-important covariates to be exactly zero, like GMIFS algorithm, the model was able to assign important covariates with much higher posterior inclusion probabilities. For example, the posterior inclusion probabilities for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ were both 1; on the contrary, the posterior inclusion probabilities for $\boldsymbol{x}_3$,

$\boldsymbol{x}_4$, and $\boldsymbol{x}_5$ were 0.09, 0.101 and 0.09, respectively (Table 3.2).

Table 3.2: Simulation results from three different cumulative logit models

	True parameters	VGLM coefficient estimates	GMIFS coefficient estimates	Bayesian LASSO coefficient estimates	Bayesian LASSO Posterior inclusion probability
α_1	-1	-1	-1	-1	-
α_2	2	2.3	2.3	2.3	-
β_1	3	3.2	3.1	3.2	1
β_2	1	1.1	1	1.1	1
β_3	-	-0.02	0	-0.001	0.09
β_4	-	0.09	0.07	0.008	0.101
β_5	-	0.023	0	0.002	0.09

The last column is the inclusion probabilities obtained from Bayesian LASSO cumulative logit model which are not directly comparable to the slope estimates (columns 1 to 4)

3.2.2 Simulation II

The goal of simulation study II is to examine the ability of our method to correctly incorporate true predictors from noise predictors into the model when the feature space is high-dimensional. To start the simulation, 90 ordinal responses were predefined ($n = 90$): 30 of them belong to category 1 ($Y_1 = ... = Y_{30} = 1$), 30 belong to category 2 ($Y_{31} = ... = Y_{60} = 2$), and the remaining 30 belong to category 3 ($Y_{61} = ... = Y_{90} = 3$). We then designed our covariate matrix to consist of 100 covariates ($P = 100$) and 90 observations. Among these 100 covariates, we let the first five covariates be the important predictors, which are truly associated with the response, and we let the remaining 95 be non-important covariates, which are not associated with the response. Our covariate matrix $\boldsymbol{X}$ was generated as a combination

of four submatrices,

$$\boldsymbol{X} = \left(\begin{array}{c|c} \begin{array}{c} \boldsymbol{X}_a \\ {\scriptstyle (30\times 5)} \\ \boldsymbol{X}_b \\ {\scriptstyle (30\times 5)} \\ \boldsymbol{X}_c \\ {\scriptstyle 30\times 5} \end{array} & \begin{array}{c} \boldsymbol{X}_d \\ (90\times 95) \end{array} \end{array} \right),$$

where the elements in the submatrices were simulated as follow:

- elements in $\boldsymbol{X}_a$ were randomly generated from $\sim N(0, 0.16)$;
- elements in $\boldsymbol{X}_b$ were randomly generated from $\sim N(1, 0.16)$;
- elements in $\boldsymbol{X}_c$ were randomly generated from $\sim N(2, 0.16)$; and
- elements in $\boldsymbol{X}_d$ were randomly generated from $\sim N(0, 0.16)$.

In this way, $[\boldsymbol{X}_a, \boldsymbol{X}_b, \boldsymbol{X}_c]^T$ is the matrix of true predictors; while, $\boldsymbol{X}_d$ is the matrix of non-important predictors. Note that we used 0.16 as the variance for the normal distributions, because ordinarily the level of gene expression (on log base 2 scale) follows a normal distribution with a variance of approximately 0.16. We simulated the data in this way 100 times, so that in the end, we could examine feature selection performance by counting the number of true predictors that were identified and the number of non-important predictors that were identified. We evaluated our method by two measurements: true positives and false positives. True positives were measured by the median number and the range of correctly identified non-zero predictors over 100 simulations. False positives were measured by the median number and the range of incorrectly identified non-zero predictors over 100 simulations. Here, we compared four models:

1. A penalized cumulative logit model using a frequentist approach (GMIFS);

2. A penalized Bayesian cumulative logit model using a hyperprior $\lambda \sim Gamma(1, 1)$ (equation (3.2));

3. A penalized Bayesian cumulative logit model using a hyperprior $\lambda \sim Gamma(0.025, 0.05)$ (equation (3.2)); and

4. A penalized Bayesian cumulative logit model with informative α priors (equation (3.3)).

The reason we used a hyperprior $\lambda \sim Gamma(0.025, 0.05)$ was because if we treated our response as a continuous outcome and used least angle regression as implemented in the R package *lars*, the λ selected by CV had a mean of 0.5. Therefore, based on this prior information, we used a hyperprior $\lambda \sim Gamma(0.025, 0.05)$, where the distribution has a mean of $0.025/0.05 = 0.5$ and a relatively larger variance of $0.025/0.05/0.05 = 10$. The fourth model used informative priors for the $\boldsymbol{\alpha}$ terms, where $a_{0,1} = logit(\sum_{i=1}^{n} y_{i1}/n) \approx -0.7$, and $a_{0,k} - a_{0,k-1} = logit(\sum_{i=1}^{n} \sum_{j=1}^{2} y_{ij}/n) - logit(\sum_{i=1}^{n} y_{i1}/n) = 1.4$. Therefore, $\alpha_1 \sim N(-0.7, 1000) \quad \alpha_1 \in (-6.9, 6.9)$ and $\alpha_2 - \alpha_1 \sim Gamma(1.4, 1)$.

Table 3.3 show the results of the second simulation study. The median number of correctly identified true covariates using all four methods is 5 (range=5,5), indicating all methods perform well in identifying the true predictors. The median number of incorrectly identified non-zero coefficients using all four methods is 1 for all methods, with slightly different ranges. GMIFS had a slightly larger range compared to the

rest, but all methods had a small number of false positives. Using a different gamma hyperprior for λ did not change the results indicating that feature selection is robust for different gamma hyperpriors. Also, the Bayesian penalized cumulative logit model using informative priors for $\boldsymbol{\alpha}$ had similar results when compared to the other methods, but the chains converged faster than when using non-informative priors.

Table 3.3: Simulation II results

Median (Range)	GMIFS	Bayesian LASSO $\lambda \sim gamma(1,1)$	Bayesian LASSO $\lambda \sim gamma(0.025, 0.05)$	Bayesian LASSO $\lambda \sim gamma(1,1)$ $\alpha_k - \alpha_{k-1} \sim gamma(1.4,1)$
True Positive	5(5,5)	5(5,5)	5(5,5)	5(5,5)
False Positive	1(0,7)	1(0,5)	1(0,4)	1(0,4)

3.3 Application

The penalized Bayesian cumulative logit model was applied to a methylation dataset assayed using the Illumina GoldenGate Methylation Bead Array Cancer Panel I. The data set was downloaded from GEOquery (GSE18081) and was filtered by Archer et al. (2014). The response variable was liver hepatocellular carcinoma (HCC) status: Normal (N=20), cirrhotic but not having HCC (N=16), and HCC (N=20). Cirrhosis is considered as the middle level, because the cirrhotic liver is often described as being a pre-malignant condition of more severe liver disease such as HCC. In this study, All 20 HCC patients had cirrhosis due to HCV infection. 16 independent HCV-cirrhotic tissue from patients without HCC were collected from liver transplant patients [Archer et al., 2010]. Covariates are methylation levels on 1469 CpG sites

[Archer et al., 2014b], since interest is in predicting stage of liver disease using DNA methylation levels on different CpG sites. Traditional non-penalized methods such as ML estimation carried out by functions in the *VGAM* package can not be estimated due to the fact that the number of covariates ($P = 1469$) exceeds the number of samples ($n = 56$).

Previous to the Bayesian analysis, all covariates were standardized to reduce the correlations between explanatory variables. For the Bayesian analysis, we used non-informative normal priors for the $\boldsymbol{\alpha}$ terms. The initial values for all three chains are shown below in Table 3.4.

Table 3.4: Bayesian initiation table for Liver data

	α_1	$\alpha_2 - \alpha_1$	$\boldsymbol{\beta}$
Chain 1	$logit(\sum_{i=1}^{n} y_{i1}/n)$	$logit(\sum_{i=1}^{n}\sum_{j=1}^{2} y_{ij}/n) - logit(\sum_{i=1}^{n} y_{i1}/n)$	$\mathbf{0}$
Chain 2	-1	2	$\sim N(0, 0.01^2)$
Chain 3	-2	4	$\sim N(0, 0.001^2)$

The Bayesian estimates were based on three MCMC chains using 100,000 iterations (first 50,000 are treated as burn-in). Since the number of sampled parameters is too high for graphical evaluation, we determined the convergence solely by the values of $\hat{R}$.

Unfortunately, due to the large predictor space ($P = 1469$) and the sparsity of the model, most of CpG sites have very low posterior inclusion probabilities. And the posterior inclusion probabilities for all covariates did not have a bi-peak distribution to clearly distinguish the important features from the non-important features

like in simulation I, where 2 features had extremely high posterior inclusion probabilities (posterior inclusion probabilities for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ were both 1), and the remaining features had extremely low posterior inclusion probabilities (posterior inclusion probabilities for $\boldsymbol{x}_3$, $\boldsymbol{x}_4$ and $\boldsymbol{x}_5$ were 0.09, 0.101 and 0.09, respectively). Instead, the posterior inclusion probabilities for the liver disease data followed a relatively normal distribution with a mean of 0.5 and a SD of 0.008 (Figure 3.4). Note that the distribution also had a long right tail. Therefore, we proposed to select features that have posterior inclusion probabilities greater than mean + 3$\times$ SD. After using this selection criterion, 9 out of 1,469 features were selected. The selected CpG sites are related to the following genes: ABL1, CDKN2B, DDIT3, GML, HOXA5, MMP7, PADI4, PLSCR3, and S100A2. Note that among these selected features, CDKN2B, DDIT3, GML, PADI4 were also selected by using the generalized monotone incremental forward stage-wise method [Archer et al., 2014b]. Table 3.5 shows the genes that were selected and related literature that has previously linked these genes to liver disease.

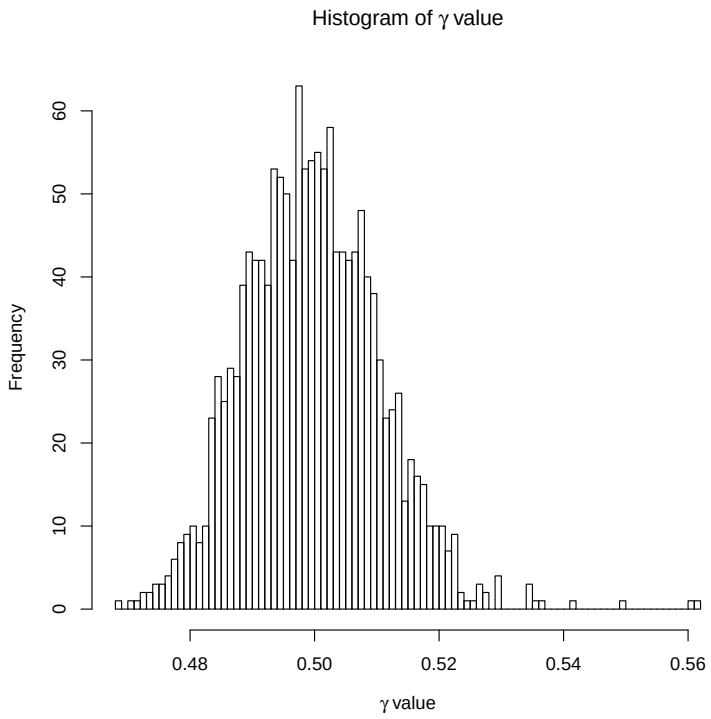

Figure 3.4: Distribution of posterior inclusion probabilities for all covariates.

Table 3.5: Genes identified as Liver disease related by penalized Bayesian cumulative logit model

PROBE ID	GENE NAME (SYMBOL)	RELEVANT PUBLICATION
ABL1_P53_F	ABL proto-oncogene 1 (ABL1)	HCC[Rana et al., 2013] Hepatitis infection [Yamauchi et al., 2015]
CDKN2B_seq_50_S294_F	Cyclin-dependent kinase inhibitor 2B isoform 1 (CDKN2B)	
DDIT3_P1313_R	DNA damage inducible transcript 3 (DDIT3)	Acute liver failure [Rao et al., 2015]
GML_E144_F	GPI anchored molecule like protein (GML)	
HOXA5_E187_F	Homeobox A5 (HOXA5)	HCC [Kanai et al., 2010]
MMP7_E59_F	Matrix metallopeptidase 7 (MMP7)	HCC [Chen et al., 2013]
PADI4_P1158_R	Peptidyl arginine deiminase, type IV (PADI4)	HCC [Zhang et al., 2013]
PLSCR3_P751_R	Phospholipid scramblase 3 (PLSCR3)	
S100A2_E36_R	S100 calcium binding protein A2 (S100A2)	cholangiocarcinoma [Sato et al., 2013] HCC (mouse model) [Wang et al., 2001]

To predict ordinal responses, we proposed to fit predictive models using only the selected features. We compared four Bayesian predictive models by examining two types of errors: re-substitution error and leave-one-out cross validation error (CV error). Lower error rates indicate better prediction power. The four predictive models are

1. Penalized Bayesian cumulative logit model with non-informative priors for $\boldsymbol{\alpha}$ terms ($\beta_j \sim DE(0, \frac{1}{\tau\lambda})$ for $j = 1, ..., P$ and $\alpha_1 \sim N(0, 1000)$, $\alpha_1 \in (-6.9, 6.9)$, $\alpha_k \sim N(0, 1000)$, $\alpha_k \in (\alpha_{k-1}, inf)$ for $k = 1, ..., K-1$);

2. Penalized Bayesian cumulative logit model with informative priors for $\boldsymbol{\alpha}$ terms ($\beta_j \sim DE(0, \frac{1}{\tau\lambda})$ for $j = 1, ..., P$ and $\alpha_1 \sim N(-0.6, 1000)$, $\alpha_1 \in (-6.9, 6.9)$, $\alpha_k - \alpha_{k-1} \sim Gamma(1.2, 1)$ for $k = 1, ..., K-1$);

3. Non-penalized Bayesian cumulative logit model with non-informative priors for $\boldsymbol{\alpha}$ terms ($\beta_j \sim N(0, 1000)$ for $j = 1, ..., P$ and $\alpha_1 \sim N(0, 1000)$, $\alpha_1 \in (-6.9, 6.9)$, $\alpha_K \sim N(0, 1000)$,$\alpha_k \in (\alpha_{k-1}, 6.9)$ for $k = 1, ..., K-1$); and

4. Non-penalized Bayesian cumulative logit model with informative priors for $\boldsymbol{\alpha}$ terms ($\beta_j \sim N(0, \frac{1}{\tau\lambda})$ for $j = 1, ..., P$ and $\alpha_1 \sim N(-0.6, 1000)$, $\alpha_1 \in (-6.9, 6.9)$, $\alpha_k - \alpha_{k-1} \sim Gamma(1.2, 1)$ for $k = 1, ..., K-1$).

The results are displayed in the Table 3.6. All four predictive models provided good predictions (small misclassification rates and CV errors). In conclusion, our penalized Bayesian cumulative logit model performs accurate feature selection and is helpful in predicting an ordinal response when applied to a high-dimensional dataset. Extensions of this approach for other ordinal response models will be investigated in

the near future. The code for performing this application appears in Appendix B.2. The model files that are necessary for running JAGS appear in Appendix B.3.

Table 3.6: Error rates from four penalized Bayesian cumulative response models.

	$\beta_k \sim DE$ $\alpha_k \sim trunN(0, 1000)$	$\beta_j \sim DE$ $\alpha_k - \alpha_{k-1}$ $\sim gamma(1.2, 1)$	$\beta_k \sim N(0, 1000)$ $\alpha_k \sim trunN(0, 1000)$	$\beta_k \sim N(0, 1000)$ $\alpha_k - \alpha_{k-1}$ $\sim gamma(1.2, 1)$
Misclassification rate	0.107	0.107	0.107	0.09
Leave-one-out CV error	0.161	0.178	0.179	0.23

Chapter 4

Univariate feature selection method

This chapter is supplemental to the previous three chapters. The first three chapters focused on penalized models that perform automatic feature selection, so the model building step is not necessary. The frequentist LASSO ordinal models using the GMIFS method in Chapter 2 can shrink coefficients corresponding to non-important covariates to be exactly zero, and therefore, leave only important features in the model. In chapter 3, the Bayesian LASSO is not able to shrink coefficients to exactly zero, but it is able to give the important covariates high posterior inclusion probabilities, so that feature selection is achievable. In this chapter, we describe a competing method for feature selection other than frequentist or Bayesian penalization. We assume that many of the features are irrelevant to predicting overall survival and thus can be removed by some feature selection methods before fitting a multi-

variable model. This is a univariate feature selection or filtering method. In section 4.1, we introduce the survival dataset that inspired us to examine filtering methods, we then examine the performance of filtering methods in comparison to penalization methods. The survival dataset contains clinical variables and proteomic measurements for 187 subjects with acute myeloid leukemia (AML). The goal is to predict survival using both proteomic expression and other covariates. In section 4.2, four univariate feature selection methods are described, together with two penalization approaches for comparison purposes. We also describe our error assessment process. The results of filtering and penalization when applied to high-dimensional protein expression data are presented in section 4.3.

4.1 Introduction

Acute myeloid leukemia (AML) is a cancer of the blood and the bone marrow. Mutations in the myeloid line of blood stem cells lead to the formation of aberrant myeloid blasts and white blood cells. If a treatment fails to destroy all the neoplastic cells, the rapid or delayed regrowth of blasts can eventually lead to death [Kornblau et al., 2013]. It was estimated that in 2014, there were 18,860 new cases of AML in the United States, and the estimated number of deaths reached 10,460 [Society, 2014]. As AML progress rapidly, only about one fourth of the patients diagnosed with AML survive beyond 5 years. Therefore, there is an urgency in finding better treatments for AML [York et al., 2012].

One difficulty in treating AML is that the cancer is heterogeneous. It is a collection of diseases that often share a similar clinical presentation, for example, weight loss, fatigue, fever or low white blood cell counts; however, these diseases can arise from diverse mutations and genetic events [Mardis et al., 2009]. As a consequence, AML patients treated with similar therapies often respond differently. For example, AML associated with cytogenic abnormalities t(8;21), t(15,17) or inv(16) are predicted to survive longer (5-year survival =70%), whereas patients with cytogenic abnormalities -5, -7, del(5q) or abnormal 3q are predicted to survive a shorter time (5-year survival=15%) [Grimwade et al., 1998]. Currently, protein expression has become more important in affecting AML treatment [Kornblau et al., 2009]. For example, it has been shown that patients with low or high Friend leukemia virus integration 1(FLI1) expression had shorter overall survival (22.6 and 30.3 versus 51.1 weeks, respectively) [Kornblau et al., 2011]. In another study, Kornbalu (2009) separated patients into seven protein signature groups using principle component analysis and showed that these signature groups were associated with overall survival [Kornblau et al., 2009].

In this chapter, we sought to determine a statistical model that is able to predict overall survival of AML patients using their protein expression profiles assayed by Reverse phase protein array (RPPA). RPPA is able to assess the total proteins and their corresponding phosphoprotein of an given pathway, and has been shown to have a high precision and reproduciblility in printing, detecting, amplificating, and staining arrays [Tibes et al., 2006].

However, overall survival time is not immediately predictable. The argument has been that point prediction of survival has poor accuracy [Henderson et al., 2001]. Henderson (2001) showed that Cox, Weibull, log-normal and Aalen models all have poor predictive capabilities with practical parameter values [Henderson et al., 2001]. As a consequence, the interest may lie in seeking an alternative method of predicting actual survival time. In this study, the goal is to predict grouped survival time, where overall survival was categorized into several intervals, specifically, short, intermediate and long-term survival.

Another main challenge in predicting survival using protein expression profiles is the high-dimensionality of the covariate space. In our study, 231 proteomic measurements (proteins and phosphoprotein expression levels) and 18 clinical/baseline demographic, cytogenic and blood tests results were measured on 187 AML patients. Most traditional predictive models cannot be estimated when the number of parameters exceeds the sample size. One approach to overcome this issue is to use a penalization method, such as the generalized monotone incremental forward stagewise regression method (GMIFS) described in Chapter 1 and 2. Archer et al. (2014) extended the GMIFS algorithm to several ordinal response models, including the forward continuation ratio model with a complementary log-log link (FCR-cloglog). This model can be used to describe the hazard function for grouped survival data [Ferber and Archer, 2015] and has the advantage of being equivalent to the propor-

tional hazards model of the form:

$$\log[-\log(1-\omega_j(x))] = \alpha_j + \boldsymbol{\beta}^T \boldsymbol{x}$$

where $\omega_j(x) = P(Y = j|Y \geq j)$ for $j = 1, ..., K-1$. K is the level of the grouped survival, α_j denotes the class-specific intercept and the $\boldsymbol{\beta}$ vector represents the coefficients associated with the covariate matrix $\boldsymbol{x}$ in this study [Agresti, 2010]. Although the GMIFS extended FCR-cloglog model (GMIFS-FCR) is helpful in predicting grouped survival time, currently there is no study in comparing its performance to other feature selection methods or to other existing penalized survival models, such as a penalized Cox model. Tibshirani (1997) proposed a version of the LASSO for the Cox model [Tibshirani, 1996]. Park and Hastie (2007) developed a path following method for the Cox PH model that uses the predictor-corrector method called L1-regularization path algorithm for the Cox model (Coxpath) [Park and Hastie, 2007]. The details of the GMIFS extended FCR model and Coxpath are described in section 4.2.2.

Another traditional method to overcome the issue of high-dimensionality is to assume that many of the features are irrelevant to predicting overall survival and thus can be removed by some feature selection method before fitting a multivariable model. Here, we describe competing methods for feature selection. These four methods are univariate Cox proportional hazards (PH) model, Spearman's rank correlation test, and two additional methods that are based on the categorization of continuous feature data (for example, low, intermediate and high-protein expression) referred to

as Importance Scores I_c and I_d. The difference between the two importance scores is that I_c is based on continuous survival time and I_d is based on discretized survival (grouped survival). These two importance scores are inspired by Multigene profile association method (MPAS) proposed by Yan and Zheng [Yan and Zheng, 2008], and can identify important genes for capturing the difference in survival time between expression categories. We describe the four filtering methods in section 4.2.1. In the last section (4.3), we will investigate the performance of all previously mentioned methods to determine if any method yields superior prediction accuracy when applied to our high-dimensional AML dataset.

4.2 Univariate feature selection and penalization methods

This section contains four subsections. In the first two subsections, the four univariate feature selection methods and the two penalized methods: GMIFS extended forward continuation ratio model with a complementary log-log link (GMIFS-FCR) and the L1-regularization path algorithm for the Cox model (Coxpath), are described. After that, we illustrate how these methods can be used in predicting our ordinal response when the dataset is high-dimensional (Section 4.2.3) and how well their prediction can be measured (Error assessment) (section 4.2.4).

4.2.1 Univariate feature selection methods

In this study, we first focused on selecting features based on the importance of individual features. The first feature selection method uses p-values from fitting univariate Cox proportional hazards (PH) models. The Cox PH model has long been used as a statistical technique to explore the relationship between survival time and patient features. Herein, Cox PH models were fit to each of 249 features (18 clinical/demographic and 231 proteomic measurements) and features having an observed p-value less than 0.1 were retained for multivariable model building.

A common way to test association between two continuous variables is correlation. Because survival time may not follow a Gaussian distribution, non-parametric Spearman's rank correlation test was performed to estimate the strength of the relationship between survival and each feature. The significance level was again set at 0.1. We note, however, that Spearman's rank correlation does not take censoring into account.

The remaining two feature selection methods were inspired by methods developed by Yan and Zheng [Yan and Zheng, 2008], where they discretized protein expression into three levels: high, normal, or low using K-means clustering. They argued that discretization can simplify the data and make the analysis more resistant to outliers or extreme values [Yan and Zheng, 2008]. In their first feature selection method, after discretization, they created a multigene profile association score (MAPS) for a given gene among P genes and this score measures the importance of each gene in

terms of its association with a binary outcome, given the current genes. Based on the MAPS, they selected important genes through a backward elimination for class prediction [Yan and Zheng, 2008]. Here, we propose two methods. The first is designed to capture differences in continuous survival time between discretized protein expression categories (I_c), while the second is designed to capture the association between discretized protein expression levels and grouped survival (I_d)

The importance score based on continuous overall survival (I_c) is defined for each feature as

$$I_c(x_p) = \frac{\sum_{k=1}^{K} n_k^2 (\bar{Y_k} - \bar{Y})^2}{\sum_{i=1}^{n} (Y_i - \bar{Y})^2}$$

where K represents the number of discrete levels of the feature. The continuous features were discretized into $K = 3$ groups using K-means clustering independently. K-mean clustering is an algorithm that partitions data points into a pre-specified number of clusters, K. In our study, for each continuous feature, the 187 observations were clustered into one of K groups so that the within cluster variance was minimized [Hastie et al., 2001]. n represents the total number of observations; n_k represents the number of observations at level k; $\bar{Y_k}$ represents the average overall survival at level k; $\bar{Y}$ represents the global average of overall survival and Y_i represents the observed overall survival for observation i. Because I_c is a complicated statistic without a well-defined distribution, a bootstrap technique was used to perform hypothesis testing. In our study, the null hypothesis is that a feature is not associated with overall survival while the alternative hypothesis (H_a) is that a fea-

ture is associated with overall survival. I_c is defined such that the more the data are consistent with alternative hypothesis, the greater I_c tends to be.

The bootstrap resampling method for the p^{th} feature was performed according to the following steps

1. Combine observations for a feature from different discrete levels together, define the number of observations in level 1 = n_1, number of observations in level 2 = n_2, etc.

2. Draw a random sample with replacement (bootstrap sample) where the first n_1 observations are taken to represent group 1, the second set of n_2 observations are taken to represent group 2, etc.

3. Recalculate the importance scores using that bootstrap samples, denoted as $I_c^*(x_p)$.

4. Repeat steps 1 through 3, B=1000 times.

Thus p-values for testing H_0 for I_c were defined as

$$\text{p-value}_{Ic}(x_p) = \frac{\sum_{b=1}^{B} I\left(I_c^*(x_p) \geq I_c^{obs}(x_p)\right)}{B}$$

where $I_c^{obs}(x_p)$ is the observed test statistic for the p^{th} feature.

We developed importance score I_d based on categorized survival to leverage the ordinal nature of grouped survival. The continuous features were again discretized

by applying K-means clustering ($K = 3$). Let $F_k(g)$ represent the proportion of samples characterized by an outcome less than or equal to g within cluster k, then for each variable, the importance score, I_d, is defined as

$$I_d(x_p) = \sum_{k=1}^{K} \sum_{g=1}^{G} F_k(g)(1 - F_k(g))$$

where $g = 1, 2$ or 3. Note I_d is the summation of ordinal impurity functions based on nominal-ordinal association proposed by Piccarreta (2001) over K clusters. The ordinal impurity function for J classes ordinal response is

$$i_{os}(t) = \sum_{g=1}^{G} F(g)(1 - F(g))$$

where $F(g)$ represent the proportion of samples characterized by an outcome less than or equal to g [Archer, 2010, Piccarreta, 2001, 2008].

We also used a bootstrap technique to perform hypothesis testing. I_d is defined such that the more the data are consistent with the alternative, the smaller I_d tends to be. Therefore, p-values for testing of the H_0 for I_d were defined as

$$\text{p-value}_{Id}(x_p) = \frac{\sum_{b=1}^{B} I\left(I_d^*(x_p) \leq I_d^{obs}(x_p)\right)}{B}$$

Where $I_d^{obs}(x_p)$ and $I_d^*(x_p)$ are the observed and bootstrapped test statistics for the p^{th} feature. For both I_c and I_d, features with a p-value less than 0.1 were retained

for multivariable modeling.

4.2.2 Penalization approach

Two penalized approaches were examined for feature selection: GMIFS-FCR and Coxpath.

Ferber and Archer (2015) implemented the GMIFS algorithm for the forward continuation ratio model, which was implemented in the *ordinalgmifs* R package [Archer et al., 2014b] and is often used when interest lies in estimating the odds of shorter survival compared to longer survival. Park and Hastie (2007) proposed the L1-regularization path algorithm for the Cox model (Coxpath). Coxpath implements the predictor-corrector method to calculate the coefficient estimates iteratively as the tuning parameter λ varies. The algorithm first determines the λ_{max} which penalizes all coefficients except the intercept to be zero and then alternates between a predictor and a corrector step as λ decreases by a pre-defined step length. Park and Hastie (2007) introduced the concept of the "active" set where only selected variables on the iteration are contained. For example, if $\lambda = \lambda_{max}$, the active set only contains the intercept. They also suggested to use a step length equal to the difference between λ_k and $\lambda_k + 1$ that will change the active set of variables (k stands for k^{th} iteration). To illustrate further, on the k^{th} iteration, Coxpath linearly approximates the coefficient vector, called $\hat{\boldsymbol{\beta}}^{k+}$ in the predictor step and in the corrector step, it finds the exact coefficient solution $\hat{\boldsymbol{\beta}}^{k+1}$ by minimizing the partial likelihood of Cox PH model and using $\hat{\boldsymbol{\beta}}^{k+}$ as the initial value. The iterative process continues until the active set

cannot be augmented any further [Park and Hastie, 2007].

4.2.3 Prediction

After selecting features using four different methods (Univariate Cox PH model, Spearman's rank correlation test, I_c and I_d), both FCR-cloglog and Cox PH models were fit to predict grouped survival using the more parsimonious set of features as predictors. FCR-cloglog produced the fitted conditional probabilities that can be used to estimate the class specific probability π_{ik} for i^{th} subject and k^{th} class. The predicted class $\hat{\omega}$ for observation i can be determined by

$$\hat{\omega}_i = \arg\max(\pi_{ik}).$$

GMIFS-FCR and Coxpath were also applied to the dataset with no filtering. For GMIFS-FCR, the above approach was applied for prediction.

Predicting grouped survival for the Cox PH model and Coxpath model is more complicated because the Cox model is not designed for predicting discrete survival time. In this study, we proposed to predict grouped survival by estimating the class specific probability from the predicted survival curve. To be more specific, after fitting a Cox PH model or a Coxpath model, the baseline hazards were estimated using the extension of the Nelson-Aalen estimate proposed by Cox and Oakes (1984). In our study, there were n individuals with observed survival or censored times, $t_1, t_2, ..., t_n$ and r distinct failure times. Arranging these failure times in ascending

order $t_{(1)} < t_{(2)}, ..., < t_{(r)}$, then $t_{(i)}$ is the i^{th} failure time. Let d_i denote the number of failures at time $t_{(i)}$; R_i represent the risk set at $t_{(i)}$; and $\boldsymbol{x}_j$ represent a $p \times 1$ covariate vector for individual j $(j = 1, ..., n)$. The baseline hazard at distinct failure time $t_{(i)}$ is estimated as

$$\hat{\lambda}_i = \frac{d_i}{\sum_{j \in R_i} e^{\boldsymbol{x}_j^T \boldsymbol{\beta}}}.$$

The baseline hazards at censored times are zero. Therefore, the cumulative baseline hazard function at time t is estimated as

$$\hat{\Lambda}_0(t) = \sum\nolimits_{i:t(i)<t} \hat{\lambda}_i$$

and the cumulative hazard and the predicted survival for the j^{th} individual at time t are estimated as

$$\hat{\Lambda}_j(t) = \hat{\Lambda}_0(t) e^{\boldsymbol{x}_j^T \boldsymbol{\beta}} \qquad \hat{S}_j(t) = -e^{\hat{\Lambda}_j(t)}.$$

After that, the difference in the range of the estimated survival probability was used for estimating class specific probability for class k. To describe using a formula, for the j^{th} subject, define ΔS_{jk} as the difference in the range of $\hat{S}_j(t)$ when $t \leq 52$, $52 < t \leq 104$, $t \geq 104$, respectively, for class k (k=1, 2 or 3). Then, the class-specific probability π_{jk} for the j^{th} subject and k^{th} class can be estimated as

$$\hat{\pi}_{ik} = \Delta S_{jk}.$$

The class with highest probability is taken to be the predicted class.

4.2.4 Error assessment

We investigated the accuracy of all proposed methods by examining two types of errors: re-substitution error and leave-one-out cross validation error (CV error). Lower error indicates better performance.

4.3 Application

In this section, we applied the two existing penalized methods as well as the four feature selection methods to the AML data. We compared the results to determine if any methods yielded consistently superior predictions. Statistical methods used are displayed in Figure 4.1. The two penalization methods are displayed in yellow blocks while the feature selection methods followed by multivariable models are displayed in blue blocks. All statistical analyses were performed using the R programming environment (version 3.0.2). Cox PH models were fit using the *survival* library; forward continuation ratio models were fit using the `vglm` function in the *VGAM* library; GMIFS-FCR were fit using the `ordinal.gmifs` function in the *ordinalgmifs* library; and Coxpath was fit using `coxpath` the function in the *glmpath* library. The code for performing four univariate feature selection followed by multivariable predictive model appears in Appendix C.1. The code for applying two existing penalized methods appears in Appendix C.2.

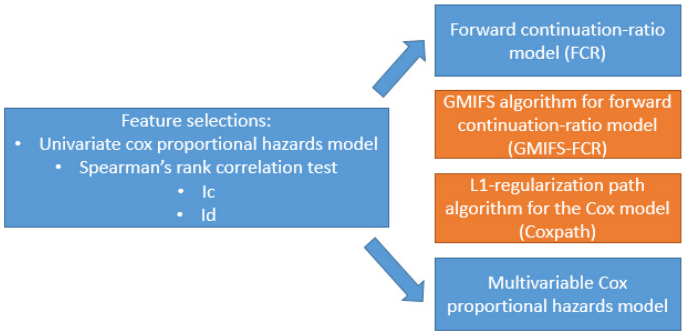

Figure 4.1: Methods flowchat

The training data for 187 AML patients was provided by M.D. Anderson Cancer Center. The data consisted of 18 covariates measuring baseline demographics (sex and age), medical histories (whether a patient has been diagnosed with a prior cancer, infection or had a prior chemotherapy, radiation therapy), cytogenetics and results from standard blood tests (counts of white blood cells, myeloid blast cells, hemoglobin and platelets together with Albumin, Bilirubin, and Creatinine levels measured in blood; myeloid blast cells, moncytes and promegakaryocytes counts in bone marrow). Although the original data contained 16 categories of AML cytogenetic abnormalities ("-5", "-5,-7", "-5,-7,+8", "-7", "-7,+8", "11q23", "21", "8", "diploid", "IM", "inv6", "inv9", "Misc", "t6;9", "t8;21" and "t9;22"), we dichotomized these 16 cytogenic abnormalities to "diploid" and "non-diploid" and labeled them as "abnormal" and "no cytogenetic abnormality" due to the low frequencies in the individual categories. The resulting abnormal group contained 86 subjects (46% of total sample size). All missing covariates were imputed using mean-imputation.

In addition, the data included proteomic measurements probed by Reverse phase

protein array (RPPA) using 231 antibodies. The process started with preparing leukemia enriched whole cell lysates from blood and bone marrow of newly diagnosed, untreated AML patients. After probed with primary and secondary antibodies, the slides coated lysate were scanned. The final data were analyzed by Microvigene® software (Vigene Tech, Carlisle, MA) and normalized by "variable slopes" and "topographical" using SuperCurve software.

Overall survival and censoring times were classified into $G = 3$ groups: 1) 52 weeks or less, 2) more than 52 weeks but less than or equal to 104 weeks or 3) more than 104 weeks to create grouped survival data. These groups were developed to represent short, intermediate and long-term survival.

After applying different feature selection methods, univariate Cox PH model, Spearman's rank correlation test, I_c and I_d methods selected 55, 76, 46 and 41 features, respectively (Table 4.1). I_c and I_d methods selected a smaller number of features relative to the other methods; while Spearman's rank correlation test selected the most. Although I_c and I_d methods produced similar numbers of features, only 21 features overlapped (Figure 4.2). Figure 4.2 is a Venn diagram illustrating the relationship between the features selected by the four feature selection methods.

Table 4.1: Significant features selected using four feature selection methods.

Methods	Selected features
Cox proportional hazards model	Age, PRIOR.MAL, PRIOR.CHEMO, PRIOR.XRT, cyto.cat, WBC, BM.BLAST, HGB, ALBUMIN, ACTB, ARC, BAD.pS136, BCL2L1, BECN1, BIRC2, CASP7.cl198, CCND3, CD74, EGFR.pY992, EIF2AK2, EIF2S1, EIF4E, ERG, Fli1, FN1, FOXO3.S318_321, GSKA_B, H3histon,H3K27Me3,HNRNPK, HSP90AA1_B1, HSPA1A_L,HSPB1, INPPL1, ITGA2, LCK, MAPT, NCL, NRP1, PA2G4, PA2G4.pS65, PA2G4.pT37_46, PA2G4.pT70, PIK3CA, PRKCD.pT507, SMAD2.pS245, SMAD4, SRC.pY416, STAT1.pY701, STAT3, STMN1, TP53, TRIM62, TSC2, YAP1p
Spearman's rank correlation test	Age, PRIOR.XRT, Infection, cyto.cat, WBC, ABS.BLST, BM.BLAST, HGB, ALBUMIN, ACTB, AKT1_2_3.pT308, ARC, ASH2L, BAD.pS136, BCL2L1, BECN1, BIRC2, CASP7.cl198, CASP8, CBL, CCND3, CD74, CDK4, CDKN1A, CDKN2A, EGFR.pY992, EIF2AK2, EIF2S1, EIF4E, Fli1, FN1, FOXO3.S318_321, GSKA_B, GSKA_B.pS21_9, H3histon, H3K27Me3, HDAC1, HIF1A, HNRNPK, HSP90AA1_B1, HSPA1A_L, HSPB1, INPPL1, IRS1.pS1101, ITGA2, ITGB3, KDR, LCK, LEF1, MAPT, NCL, NOTCH1.cl1744, NPM1.3542, NR4A1, PA2G4, PA2G4.pS65, PA2G4.pT70, PIK3CA, PPP2R2A_B_C_D, PRKCD.pS664, PRKCD.pT507, RAC1_2_3, SMAD3, SMAD4, SPP1, SRC.pY416, STAT3, STAT3.pS727, STK11, STMN1,TAZ, TP53, TRIM24, TRIM62, TSC2, YAP1p
I_c	PRIOR.MAL, PRIOR.CHEMO, cyto.cat, Age, WBC HGB, AKT1_2_3.pT308, ARC, BAD.pS136, BCL2L1, CASP7.cl198, CASP8, CAV1, CCNB1, CD74, CDKN2A, DIABLO, EIF2AK2, EIF4E, ERG, GSKA_B, H3K27Me3, H3K4Me3, HNRNPK, HSP90AA1_B1, HSPA1A_L, HSPB1, INPPL1, ITGA2, KDR, MAPK1_3.pT202Y204, MAPK9, NR4A1, NRP1, PA2G4.pT70, PIK3CA, PPARA, PPP2R2A_B_C_D, PRKCB.I, PTK2, SMAD1, SMAD4, SPP1, STAT5A_B.pY694, TRIM62, YAP1p
I_d	PRIOR.MAL, PRIOR.CHEMO, PRIOR.XRT, Infection, cyto.cat, Age, HGB, ALBUMIN, CREATININE, ARC, BAD.pS112, BECN1, CASP8, CCND3, DIABLO, EGFR.pY992, EIF2S1, EIF2S1.pS51, EIF4E, H3histon, H3K27Me3, HNRNPK, HSP90AA1_B1, HSPA1A_L, INPPL1, ITGA2, KDR, MAPK9, MAPT, MET.pY1230_1234_1235, NPM1, PA2G4.pS65, PA2G4.pT70, PRKCD.pT507, PTK2, STAT5A_B, TP53, TRIM62, TSC2, YAP1p, YWHAZ

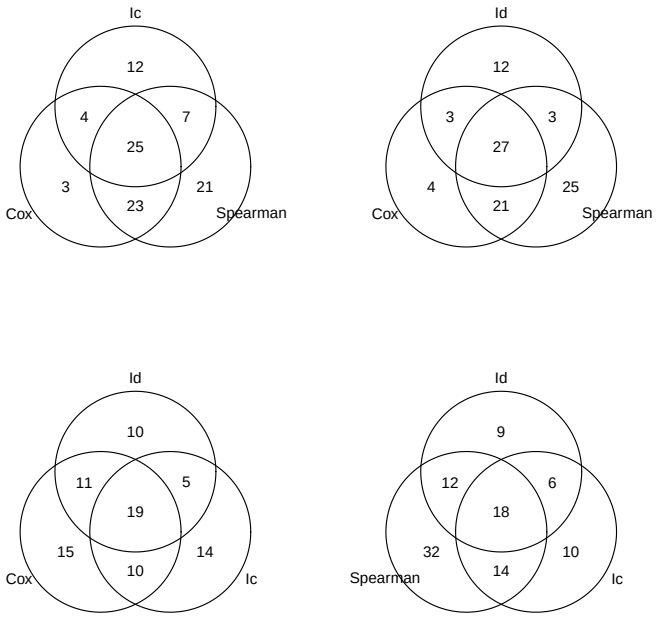

Figure 4.2: Venn diagram

After fitting GMIFS-FCR, Coxpath and FCR-cloglog/ Cox PH model for each of the four univariate feature selection methods, the re-substitution error and cross validation error associated with each method are shown in Table 4.2 and Table 4.3. The graphic presentation of error rates are shown in Figure 4.3 and Figure 4.4. Note that the Spearman's rank correlation test has lowest resubstitution error but the highest CV error because Spearman's rank correlation test selected most number of features which resulted in a complicated predictive model. However, when applied to a future data, a complicated model do not necessary provide a good prediction.

In general, the FCR-cloglog model had smaller re-substitution errors in compar-

ison to the same feature selection method followed by the Cox PH model. Although CV errors were similar regardless of the feature selection method, FCR generally had a slightly lower CV error than Cox PH.

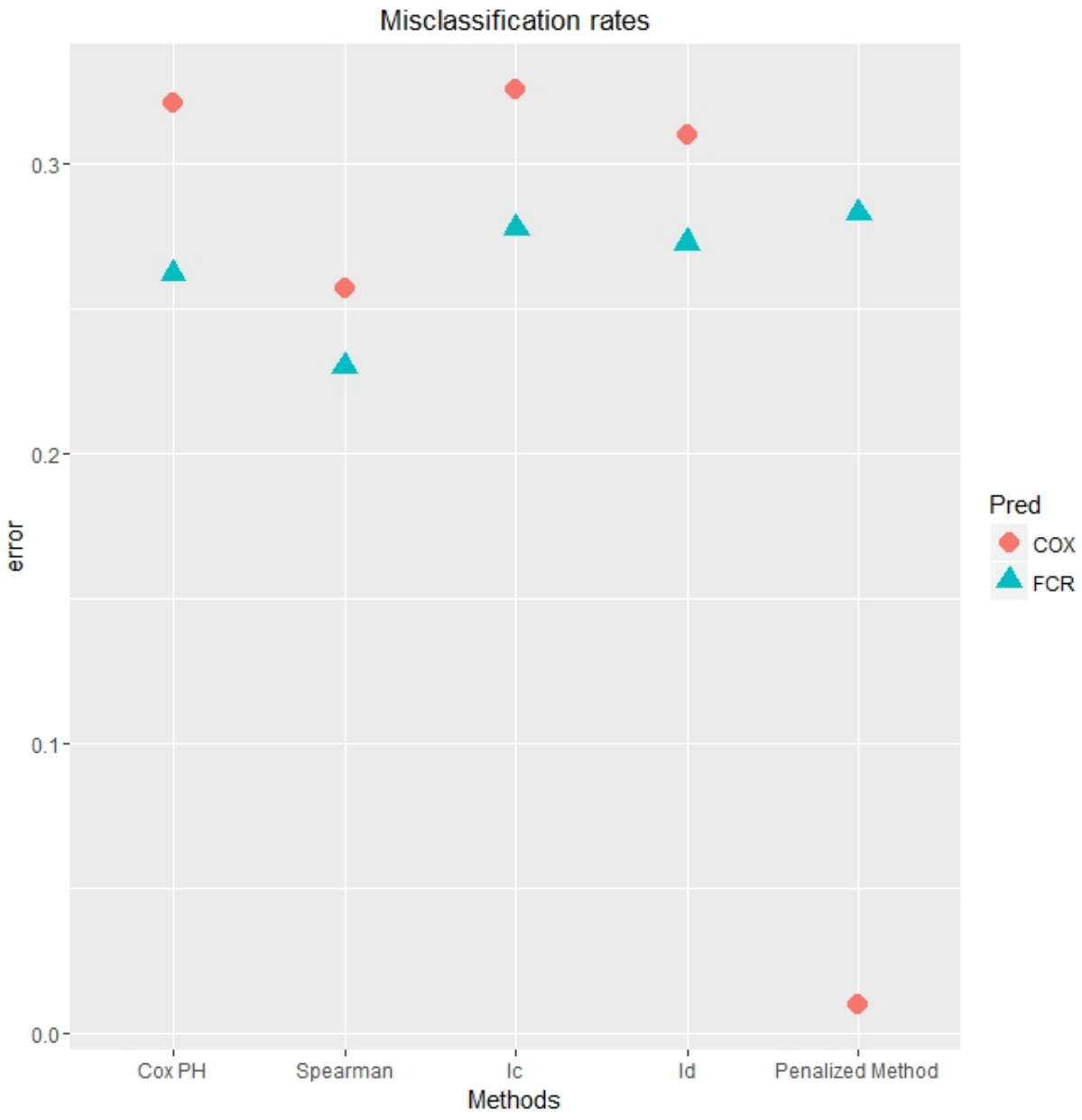

Figure 4.3: Resubstitution misclassification error rates for each filtering and modeling method.

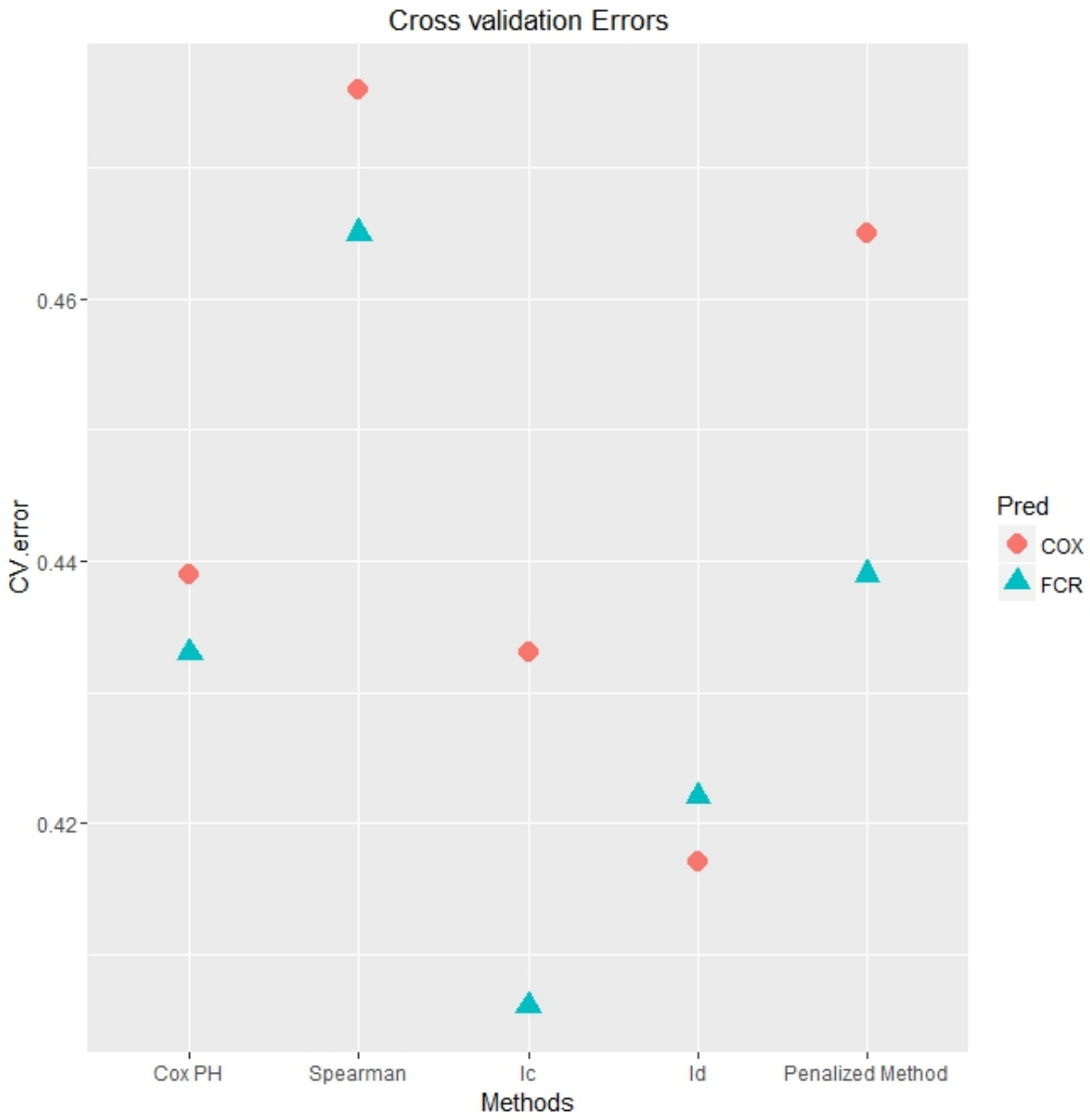

Figure 4.4: Cross-validated misclassification error rates for each filtering and modeling method.

Table 4.2: Re-substitution errors.

Univariate feature selection method	Multivariable model	Re-substitution error
Cox PH	Cox PH	0.321
Spearman	Cox PH	0.257
I_c	Cox PH	0.326
I_d	Cox PH	0.310
Cox PH	FCR	0.262
Spearman	FCR	0.230
I_c	FCR	0.278
I_d	FCR	0.273
	GMIFS-FCR	0.283
	Coxpath	0.01

* Cox PH = Cox proportional hazards model, Spearman = Spearman's rank correlation test, FCR = Forward continuation ratio model; GMIFS-FCR = GMIFS extended forward continuation-ratio model; Coxpath = L1-regularization path algorithm for the Cox model.

Table 4.3: Cross-validation errors.

Univariate feature selection method	Multivariable model	Cross-validation error
Cox PH	Cox PH	0.439
Spearman	Cox PH	0.476
I_c	Cox PH	0.433
I_d	Cox PH	0.417
Cox PH	FCR	0.433
Spearman	FCR	0.465
I_c	FCR	0.406
I_d	FCR	0.422
	GMIFS-FCR	0.439
	Coxpath	0.465

* Cox PH = Cox proportional hazards model, Spearman = Spearman's rank correlation test, FCR = Forward continuation ratio model; GMIFS-FCR = GMIFS extended forward continuation-ratio model; Coxpath = L1-regularization path algorithm for the Cox model.

4.3.1 Discussion

Unfortunately, other than univariate Cox proportional hazards model, the other three feature selection methods cannot handle censored data. Although both penalization methods (GMIFS-FCR and Coxpath) can incorporate censoring information, regular FCR-cloglog cannot. It would be beneficial to extend FCR-cloglog to censored data.

I_c and I_d methods had similar performance in term of cross-validation error and

number of features selected. However, I_d has advantage of analyzing grouped survival data; therefore, if only grouped data are available, the I_d method is recommended.

Chapter 5

Conclusions and Future Work

5.1 Conclusion

Ordinal responses are commonly collected in biomedical studies. There has been increasing emphasis in medical research on the relationship between clinical phenotypes and high-dimensional genomic information. Many clinical phenotypes are on an ordinal scale and thus are recommended to be analyzed using ordinal response models. Traditional methods for modeling ordinal data do not perform well in the presence of a high-dimensional covariate space, because traditional methods require that the number of samples is greater than the number of covariates and assumes covariate independence. A good solution to this problem is penalization, for example, LASSO [Tibshirani, 1996]. In chapter 1, we first reviewed the LASSO method under both a frequentist and a Bayesian framework. Under the frequentist framework, the incremental forward stagewise algorithm (IFS) for linear regression,

generalized monotone incremental forward stagewise algorithm (GMIFS) for logistic regression, and L1-regularization path for generalized linear models (i.e., Cox regression) were described. Under the Bayesian framework, the Bayesian LASSO which uses i.i.d LaPlace priors to enable penalization was described. Next, we reviewed the six classical ordinal response models: cumulative logit model, adjacent category model, forward and backward continuation ratio models, stereotype logit models and Bayesian cumulative logit model. We also reviewed two LASSO ordinal response models, glmnet.cr and glmpath.cr were reviewed.

GMIFS was recently adapted to fit cumulative logit, adjacent category, and continuation ratio models, and were shown to be capable of deriving a parsimonious classifier [Archer et al., 2014b]. However, the GMIFS method had not been adapted to fit ordinal response models with the probit link or the stereotype logit model. In chapter 2, the GMIFS method was extended to the cumulative probit model, forward continuation-ratio model with probit link and the backward continuation-ratio model with probit link. The GMIFS extended cumulative probit link model was applied to a methylation dataset to identify methylation patterns that are associated with anxiety, depression and stress. After fitting separate GMIFS extended cumulative probit models, a large number of CpG sites were found to be associated with anxiety and depression based on the AIC selected model (67 CpG sites were associated with anxiety; 19 CpG sites were associated with depression; and 10 CpG sites were associated with stress). BIC selected model can be used to avoid overfitting.

The GMIFS method was also extended to the stereotype logit model to cope with situations when the proportional odds assumption does not hold. The method was applied to a gene expression dataset to predict the severity of Alzheimer's disease (AD), and the resulting misclassification rate based on the AIC selected model was 45%. The poor performance for AIC is probably due to the fact that our AD dataset only included 31 subjects.

Chapter 3 focused on the penalized Bayesian cumulative logit model. We developed an innovative Bayesian ordinal response model that incorporates a penalty term so that a sparse model is obtained. Our Bayesian method includes the likelihood of the cumulative logit model combined with a LaPlace prior. The feature selection property is achieved by utilizing the binary variable inclusion indicator method. The proposed model was first examined using two simulation studies. It was shown that, when the data are not high-dimensional (Simulation study I), the penalized Bayesian cumulative logit model produced similar estimates to other existing methods and were close to the true underlying values. The proposed model was also able to perform feature selection by assigning important covariates with much higher posterior inclusion probabilities. If the data are high-dimensional (Simulation study II), the proposed model performs accurate feature selection, since it has good ability to correctly incorporate true predictors from noise predictors. We also applied our proposed model to a methylation dataset to demonstrate its usage in analysis of a real high-dimensional dataset. The penalized Bayesian cumulative logit model was first fit to perform feature selection. To predict ordinal responses, we

proposed to fit predictive models using only the selected features. The performance of this two step prediction method was assessed using both misclassification rate and cross-validation error. In conclusion, our proposed model provided good prediction (small misclassification rates and CV errors).

In chapter 4, filtering, a competing feature selection method that differs from penalization was addressed. We first stated the question of interest, that is, which methods can predict grouped survival more accurately using the high-dimensional acute myeloid leukemia (AML) dataset. We described our proposed methods: four filtering methods (univariate Cox proportional hazards model, Spearman's s rank correlation test, I_c and I_d) and two penalization methods (Coxpath and GMIFS-FCR). After fitting all these competing methods, results were presented and compared. In conclusion, penalized methods and filtering methods have similar performance in terms of prediction accuracy.

5.2 Future work

5.2.1 Assessing generalization errors for GMIFS extended probit model and stereotype logit model

Prediction performance of the GMIFS extended cumulative probit model and stereotype logit model were examined by assessing the misclassification rates in section 2.2.3 and section 2.3.1; however, misclassification rate cannot be used to evaluate the model performance when applying to the future data. A better way of estimating generalization error should be introduced. In section 2.2.3, we can assess the

cross-validation error. Although cross-validation procedure is straightforward, the GMIFS algorithm needs a long time to fit a model due to the fact that the dataset contains 285,173 covariates. One way to speed up the process is by parallel computation. In the stereotype logit model example in section 2.3.1, since the dataset is too small for a cross-validation, we need to find an independent testing dataset to assess the generalization error. Unfortunately, finding an independent dataset can be challenging, because the response variable in the testing dataset also needs to be the severity of Alzheimer's disease categorized based on the MiniMental Status Examination criteria. Moreover, the gene expression levels in the new testing dataset also need to be assayed by Affymetrix HG-U133A. One challenge in genomic data analysis is that since the techniques advanced rapidly, one platform may soon be out of date, it is difficult to find an independent testing dataset assayed by the same platform.

5.2.2 Selecting optimal priors for λ and $\boldsymbol{\alpha}$

In Chapter 3, we assigned the shrinkage parameter λ with a $Gamma(a, b)$ prior distribution, where $a = 1$, and $b = 1$. Here, inspired by Huang et al (2013), we proposed to obtain the optimal values of hyperparameters a and b with cross-validation by three steps. In the first step, we examined $a = b = 0.01, 1$, and 2, and a pair (a_1, b_1) corresponding to the smallest cross-validation error was obtain. In the second step, we treated b as fixed at b_1, and examined $a = -0.5, -0.3, -0.1, -0.01, 0.01, 0.05, 0.1, 0.5$, and 1, and kept an a_2 corresponding to the smallest cross-validation error. In the

third step, a was fixed at a_2, and b was chosen from the set $[0.01, 0.1, 0, 1, 2, 3, 4, 5]$, which yielded our optimal b_2 [Huang et al., 2013]. Note that this process can result in a long computation time, we propose to introduce parallel computing to speed up the MCMC convergence.

Additionally, we can check the results of the Bayesian LASSO when an improper flat prior is used for intercepts.

5.2.3 Bayesian variable selection with consideration of the correlations between features

In our proposed penalized Bayesian cumulative logit model, we made a strong assumption that all covariates are independent and we imposed i.i.d Bernoulli priors to each of the binary indicate variables, γ_i. However, the independence assumption is almost always violated when we have a high-dimensional data, especially, genomic data. For example, in section 3.3, we demonstrated the application of our proposed Bayesian method using a methylation data assayed by the Illumina GoldenGate Cancer Panel I (Illumina, San Diego, CA). This platform interrogates 1,505 CpG sites, selected from 807 genes. As a consequence, many of the CpG sites are correlated since they are located on the same gene. For instance, 463 genes are represented on the assay by two CpG sites, and 114 genes are represented on the assay by more than three CpG sites. Not only are CpG sites located on the same gene are correlated, many genes are correlated, too. For example, among these 807 interrogated genes, the OCT, NBC, MDR1, ABCG5, and ABCB4 genes are presented on the same Bile secretion pathway [Klaassen and Aleksunes, 2010]. The highly correlated nature

of genomic data has deleterious effects on the performance of our methods [Clarke et al., 2008]. Although normalization during data filtering process and standardization before statistical analysis can remove correlation, the normalization itself can make different results [Qiu et al., 2005].

When there is a known biological structure among the predictor spaces, the Bayesian framework provides a very natural setting for incorporating pre-existing correlation structures in the covariate matrix. Li and Zhang (2010) proposed theoretical and computational schemes to incorporating prior structural information in linear regression setting. They proposed to give general Ising prior for $\boldsymbol{\gamma}$ and present Gibbs Sampling scheme of $f(\gamma|\boldsymbol{Y})$. The feature selection are then performed based on the posterior inclusion probabilities of $\boldsymbol{\gamma}$.

5.2.4 Extensions of Bayesian LASSO to other cumulative logit models

In the near future, we also propose to investigate the extensions of the Bayesian LASSO for other ordinal response models. For example, probit link models, adjacent category model, forward and backward continuation-ratio models.

5.2.5 Inclusion of unpenalized predictors

It is sometimes desirable to include relevant predictors based on the prior knowledge. For example, methylation is known to be age-related; therefore, the predictive model including age as a non-penalized covariate can adjust for confounding. In the future,

we would include no penalty subset for predictors, such as age in our Bayesian LASSO framework.

Bibliography

Agresti, A. (2010). *Analysis of Ordinal Categorical Data*. John Wiley & Sons, Inc.

Anderson, J. (1984). Regression and ordered categorical variables. *Journal of the Royal Statistical Society. Series B*, 46(1):1–30.

Archer, K. (2010). rpartordinal: An R package for deriving a classification tree for predicting an ordinal response. *J Stat Softw*, 34(07).

Archer, K., Hou, J., and Williams, A. (2014a). *New Frontiers of Multidisciplinary Research in STEAM-H (Science, Technology, Engineering, Agriculture, Mathematics and Health)*, chapter Classifying Normal, Nevus, and Malignant Melanoma Skin Samples using Penalized Ordinal Regression, pages 111–133. Springer Proceedings in Mathematics and Statistics.

Archer, K., Hou, J., Zhou, Q., Ferber, K., Layne, J., and Gentry, A. (2014b). ordinalgmifs: An R package for ordinal regression in high-dimensional data settings. *Cancer Informatics*, 13:187–95.

Archer, K., Mas, V., Maluf, D., and Fisher, R. (2010). High-throughput assessment

of CpG site methylation for distinguishing between HCV-cirrhosis and HCV-associated hepatocellular carcinoma. *Mol Genet Genomics*, 293(4):341–9.

Archer, K. and Williams, A. (2012). L1 penalized continuation ratio models for ordinal response prediction using high-dimensional datasets. *Statistics in Medicine*, 31(14):1464–74.

Bibikova, M., Barnes, B., Tsan, C., Ho, V., Klotzle, B., Le, J., and Shen, R. (2011). High density DNA methylation array with single CpG site resolution. *Genomics*, 98(4):288–295.

Blalock, E., Geddes, J., Chen, K., Porter, N., Markesbery, WR, and PW, L. (2004). Incipient Alzheimer's disease: microarray correlation analyses reveal major transcriptional and tumor suppressor responses. *Proc Natl Acad Sci U S A*, 101(7):2173–8.

Bliss, C. (1934). The method of probits. *Science*, 79(2037):38–9.

Chen, L., Li, M., Li, Q., Wang, C., and Xie, S. (2013). DKK1 promotes hepatocellular carcinoma cell migration and invasion through β-catenin/MMP7 signaling pathway. *Mol Cancer*, 12:157.

Chipman, H. and Hamada, M. (1996). Bayesian analysis of ordered categorical data from industrial experiments. *Technometrics*, 38:1–10.

Clarke, R., Ressom, H., Wang, A., Xuan, J., Liu, M., Gehan, E., and Wang, Y. (2008). The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. *Nat Rev Cancer*, 8:37–49.

Cohen, S. and Williamson, G. (1988). Perceived stress in a probability sample of the united states. In Spacapam, S. and Oskamp, S., editors, *The Social Psychology of Health: Claremont Symposium on Applied Social Psychology*, Newbury Park, CA. Sage.

Dellaportas, P., Forster, J., and Ntzoufras, J. (2002). On Bayesian model and variable selection using MCMC. *Statistics and Computing*, 12(1):27–36.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *Ann. Statist.*, 32(2):407–499.

Ferber, K. and Archer, K. (2015). Modeling discrete survival time using genomic feature data. *Cancer Informatics*, Suppl.2:37–43.

George, E. and McCulloch, R. (1993). Variable selection via Gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889.

Grimwade, D., Walker, H., Oliver, F., and Goldstone, A. (1998). The importance of diagnostic cytogenetics on outcome in AML: analysis of 1,612 patients entered into the MRC AML 10 trial. the Medical Research Council Adult and Children's Leukaemia Working Parties. *Blood*, 92(7):2322–33.

Hans, C. (2009). Bayesian lasso regression. *Biometrika*, 96(4):835–845.

Hastie, T., Taylor, J., Tibshirani, R., and Walther, G. (2007). Forward stagewise regression and the monotone lasso. *Electron J Stat*, 1:1–29.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer.

Henderson, R., Jones, M., and Stare, J. (2001). Accuracy of point predictions in survival analysis. *Statistics in Medicine*, 20(20):3083–3096.

Hoerl, A. and Kennard, R. (1970). Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Huang, A., Xu, S., and Cai, X. (2013). Empirical bayesian lasso-logistic regression for multiple binary trait locus mapping. *BMC Genetics*, 14:5.

Kanai, M., Hamada, J., Takada, M., Asano, T., Murakawa, K., and Moriuchi, T. (2010). Aberrant expressions of HOX genes in colorectal and hepatocellular carcinomas. *Oncol Rep*, 23(3):843–51.

Klaassen, C. and Aleksunes, L. (2010). Xenobiotic, bile acid, and cholesterol transporters: function and regulation. *Pharmacol Rev*, 62(1):1–96.

Kornblau, S., Qiu, Y., Zhang, N., Singh, N., Faderl, S., Ferrajoli, A., York, H., Qutub, A., Coombes, K., and Watson, D. (2011). Abnormal expression of FLI1 protein is an adverse prognostic factor in acute myeloid leukemia. *Blood*, 118(20):5604–12.

Kornblau, S., Qutub, A., Yao, H., York, H., Qiu, Y., Graber, D., Ravandi, F., Cortes, J., Andreeff, M., Zhang, N., and Coombes, K. (2013). Proteomic profiling iden-

tifies distinct protein patterns in acute myelogenous leukemia CD34+CD38- stem-like cells. *PloS one*, 8(10):pp.e78453.

Kornblau, S., Tibes, R., Qiu, Y., Chen, W., Kantarjian, H Andreeff, M., Zhang, N., Coombes, K., and Gordon B, M. (2009). Functional proteomic profiling of AML predicts response and survival. *Blood*, 113:154–164.

Kuo, L. and Mallick, B. (1998). Variable selection for regression models. *Sankhya: The Indian Journal of Statistics, Series B*, 60(1):65–81.

Lambert, S., Pallant, J., Boyes, A., King, M., Britton, B., and Girgis, A. (2013). A rasch analysis of the hospital anxiety and depression scale (HADS) among cancer survivors. *Psychological Assessment*, 25(2)(2):379–390.

Lykou, A. and Ntzoufras, I. (2013). On Bayesian lasso variable selection and the specification of the shrinkage parameter. *Statistics and Computing*, 23(3):361–390.

Mardis, E., Ding, L., Dooling, D., Larson, D., and Timothy, J. (2009). Recurring mutations found by sequencing an acute myeloid leukemia genome. *The New England Journal of Medicine*, 361(11):1058–1066.

Ntzoufras, I. (2008). *Bayesian Modeling Using WinBUGS*. John Wiley and Sons, Inc.

Park, M. and Hastie, T. (2007). L1-regularization path algorithm for generalized linear models. *J. R. Statis. Soc. B*, 69(4):659–677.

Park, T. and Casella, G. (2008). The Bayesian lasso. *Journal of the American Statistical Association*, 103(485):681–686.

Piccarreta, R. (2001). A new measure of nominal-ordinal association. *Journal of Applied Statistics*, 28:107–120.

Piccarreta, R. (2008). Classification trees for ordinal variables. *Computational Statistics*, 23:407–427.

Qiu, X., Brooks, A., Klebanov, L., and Yakovlev, A. (2005). The effects of normalization on the correlation structure of microarray data. *BMC Bioinformatics*, 6:120.

Rana, A., Ali, G., Ali, S., Khan, A., Sabiha, B., Malik, S., Riaz, A., and Farooqi, A. (2013). Breakpoint cluster region-c-abl oncogene 1, non-receptor tyrosine kinase signaling: current patterns of the versatile regulator revisited. *J Cancer Res Ther*, 9(1):3–5.

Rao, J., Zhang, C., Wang, P., Lu, L., Qian, X., Qin, J., Pan, X., Li, G., Wang, X., and Zhang, F. (2015). C/EBP homologous protein (CHOP) contributes to hepatocyte death via the promotion of ERO1 signalling in acute liver failure. *Biochem J*, 466(2):369–78.

Sato, Y., Harada, K., Sasaki, M., and Nakanuma, Y. (2013). Clinicopathological significance of S100 protein expression in cholangiocarcinoma. *J Gastroenterol Hepatol*, 28(8):1422–9.

Society, A. C. (2014). Cancer facts and figures 2014. *Altanta: American Cancer Society*.

Tibes, R., Qiu, Y., Lu, Y., Hennessy, B., Andreeff, M., Mills, G., and Kornblau, S. (2006). Reverse phase protein array: validation of a novel proteomic technology and utility for analysis of primary leukemia specimens and hematopoietic stem cells. *Molecular Cancer Therapeutics*, 5(10):2512–21.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J R stat Soc B*, 58(267-88).

Wang, X., Chen, L., and Zeng, Y. (2001). Calcium-binding protein S100A2 in the retardation of growth and proliferation of hepatocellular carcinoma cells. *Zhonghua Zhong Liu Za Zhi*, 23(5):363–5.

Yamauchi, S., Takeuchi, K., Chihara, K., Sun, X., Honjoh, C., Yoshiki, H., Hotta, H., and Sada, K. (2015). Hepatitis C virus particle assembly involves phosphorylation of NS5A by the c-Abl tyrosine kinase. *J Biol Chem*, 290(36):21857–64.

Yan, X. and Zheng, T. (2008). Selecting informative genes for discriminant analysis using multigene expression profiles. *BMC Genomics*, 9(Suppl+2)(S14).

York, H., Kornblau, S., and Qutub, A. (2012). Network analysis of reverse phase protein expression data: Characterizing protein signatures in acute myeloid leukemia cytogenetic categories t(8;21) and inv(16). *Proteomics*, 12(13):2084–2093.

Zhang, C., Fan, L., Fan, T., Wu, D., Gao, L., Ling, Y., Zhu, J., Li, R., and Wei, L. (2013). Decreased PADI4 mRNA association with global hypomethylation in hepatocellular carcinoma during HBV exposure. *Cell Biochem Biophys*, 65(2):187–95.

Zhou, T., Lv, X., Guo, X., Ruan, B., Liu, D., Ding, R., Gao, Y., Ding, J., Dou, K., and Chen, Y. (2015). RACK1 modulates apoptosis induced by sorafenib in HCC cells by interfering with the IRE1/XBP1 axis. *Oncol Rep*, 33(6):3006–14.

Zigmond, A. and Snaith, R. (1983). The hospital anxiety and depression scale. *Acta Psychiatrica Scandinavica*, 67(6):361–370.

Appendix A

Chapter 2 appendix

A.1 GMIFS cumulative probit model with example code

```
1 ############################----------------------------------
2 The code for fitting GMIFS cumulative probit models code in ordinal.
    gmifs function
3 ############################----------------------------------
4 cumprobit.likelihood<-function (par, xmatrix, y)
5 {
6     k <- length(unique(y))
7     levels <- sort(unique(y))
8     Ymat <- matrix(0, nrow = length(y), ncol = k)
9     for (i in levels) {
```

```
10        Ymat[which(y == i), which(levels == i)] <- 1
11    }
12    alpha <- numeric()
13    alpha[1] <- -Inf
14    alpha[k + 1] <- 400 ###change to 100 to match Jiayi
15    alpha[2:k] <- par[1:(k-1)]
16    beta<- par[k:length(par)]
17    Xb <- xmatrix %*% beta
18    G.mat <- matrix(0, nrow = length(y), ncol = k+1)
19  G.mat[,1]<-0
20  G.mat[,k+1]<-1
21  for (i in 2:k) { G.mat[,i]<-pnorm(alpha[i]+Xb) }
22    pi <- matrix(0, nrow = length(y), ncol = k)
23    for (i in 2:(k + 1)) {
24        pi[, i - 1] <- G.mat[,i] - G.mat[,i-1]
25    }
26    pi <- apply(pi * Ymat, 1, sum)
27    loglik <- sum(log(pi))
28    -loglik
29 }
30
31 cumprobit.stepwise<-function (x, y, epsilon = 0.0001, tol = 1e-05,
     scale = FALSE)
```

```
{
    levels <- sort(unique(y))
    k <- length(unique(y))
    x <- as.matrix(x)
    vars <- dim(x)[2]
  oldx <- x
    if (scale) {
        x <- scale(x, center = TRUE, scale = TRUE)
    }
    x <- cbind(x, -1 * x)
    Ymat <- matrix(0, nrow = length(y), ncol = k)
    for (i in levels) {
        Ymat[which(y == i), which(levels == i)] <- 1
    }

    # initiating alpha
    pi.0 <- table(y)/length(y)
    alpha <- qnorm(cumsum(pi.0))[1:(k - 1)]
    beta <- rep(0, dim(x)[2])
    names(beta) <- dimnames(x)[[2]]
    step <- 0
    Estimates<-matrix(0,ncol=dim(oldx)[[2]])
    alpha.update <- matrix(alpha, ncol = k-1)
```

```
55    Likelihood<-numeric()
56    AIC<-numeric()
57
58  ui<-matrix(0,nr=length(alpha)-1,nc=length(alpha)) #constrain matrix
59  for (i in 1:dim(ui)[1]){
60  ui[i,i]<- -1
61  ui[i,i+1]<- 1
62  }
63
64     ci<-rep(0,dim(ui)[1])
65    repeat {
66        z <- matrix(ncol = k - 1, nrow = length(y))
67        for (i in 1:(k - 1)) {
68            z[, i] <- alpha[i] + x %*% beta
69        }
70
71
72 u1<-matrix(nr=dim(x)[1],nc=k)
73 for (j in 1:k){
74  if (j==1){
75  u1[,j]<-Ymat[,1]*dnorm(z[,1])/pnorm(z[,1])
76  }
77  else if (j <= k-1 ){
```

```
78  u1[,j]<-Ymat[,j]*(dnorm(z[,j])-dnorm(z[,j-1]))/(pnorm(z[,j])-pnorm(
       z[,j-1]))
79  }
80  else if (j == k) {
81  u1[,j]<- -Ymat[,k]*dnorm(z[,k-1])/(1-pnorm(z[,k-1])+1e-16)
82  }
83 }
84
85 u<- -t(x) %*% apply(u1,1,sum)
86
87
88
89        update.value <- min(u)
90        update.j <- which.min(u)
91        if (update.value < 0) {
92            beta[update.j] <- beta[update.j] + epsilon
93        }
94        Estimates<-rbind(Estimates,beta[1:vars]-beta[(vars+1):length(
             beta)])
95
96
97     out<-constrOptim(alpha.update[step+1,],fn.cumprobit,grad=gr.
          probit, ui=ui,ci=ci,x=x[,1:vars], y=y,
```

```
98  beta=beta[1:vars]-beta[(vars+1):length(beta)],method="BFGS")
99     #out<-optim(alpha.update[step+1,], fn.cumprobit, x=x[,1:vars], y=
           y, beta=beta[1:vars]-beta[(vars+1):length(beta)], method="
           BFGS")
100   alpha.update <- rbind(alpha.update, out$par)
101   alpha <- out$par
102      p <- sum(Estimates[step+2,]!=0) + length(alpha)
103
104      Likelihood[step+1]<- LL1<- -out$value
105      AIC[step+1]<-2*p-2*Likelihood[step+1]
106      print(step)
107
108      #cat("update.value=",update.value,"\n")
109
110      if (step >= 1 && LL1 - LL0 < tol) {
111         break
112      }
113      LL0 <- LL1
114      step <- 1 + step
115   }
116   beta <- Estimates[-1,]
117 alpha<-alpha.update[-1,]
118 model.select<-which.min(AIC)
```

```
119     list(beta = beta, alpha = alpha, x=oldx, y=y, scale=scale,
           Likelihood=Likelihood, AIC=AIC, model.select=model.select)
120 }
121
122 fn.cumprobit<-function (par,beta,x, y)
123 {
124     k <- length(unique(y))
125     levels <- sort(unique(y))
126     Ymat <- matrix(0, nrow = length(y), ncol = k)
127     for (i in levels) {
128         Ymat[which(y == i), which(levels == i)] <- 1
129     }
130     alpha <- numeric()
131     alpha[1] <- -Inf
132     alpha[k + 1] <- 400 ###change to 100 to match Jiayi
133     alpha[2:k] <- par
134     xmatrix<-as.matrix(x)
135     Xb <- xmatrix %*% beta
136     G.mat <- matrix(0, nrow = length(y), ncol = k+1)
137   G.mat[,1]<-0
138   G.mat[,k+1]<-1
139   for (i in 2:k) { G.mat[,i]<-pnorm(alpha[i]+Xb) }
140     pi <- matrix(0, nrow = length(y), ncol = k)
```

```
141     for (i in 2:(k + 1)) {
142         pi[, i - 1] <- G.mat[,i] - G.mat[,i-1]
143     }
144     pi <- apply(pi * Ymat, 1, sum)
145     loglik <- sum(log(pi))
146   #cat(par,"\n")
147   -loglik
148 
149 }
150 
151 #gradient function that used in constrOptim function
152 gr.probit<-function(par,beta,x, y){
153     k <- length(unique(y))
154     levels <- sort(unique(y))
155     Ymat <- matrix(0, nrow = length(y), ncol = k)
156     for (i in levels) {
157         Ymat[which(y == i), which(levels == i)] <- 1
158     }
159 
160     alpha <- par
161     xmatrix<-as.matrix(x)
162     Xb <- xmatrix %*% beta
163     z <- matrix(ncol = k - 1, nrow = length(y))
```

```
164         for (i in 1:(k - 1)) {
165             z[, i] <- alpha[i] + Xb
166         }
167
168 grad<-matrix(nr=dim(xmatrix)[1],nc=k-1)
169 for (j in 1:(k-1)){
170   if (j==1){
171   grad[,j]<-Ymat[,1]*dnorm(z[,1])/pnorm(z[,1])- Ymat[,2]*(dnorm(z
        [,1])/(pnorm(z[,2])-pnorm(z[,1])))
172
173   }
174   else if (j < k-1 ){
175   grad[,j]<-Ymat[,j]*(dnorm(z[,j])/(pnorm(z[,j])-pnorm(z[,j-1])))-
        Ymat[,j+1]*(dnorm(z[,j])/(pnorm(z[,j+1])-pnorm(z[,j])))
176   }
177   else if (j == k-1) {
178   grad[,j]<- Ymat[,k-1]*(dnorm(z[,k-1])/(pnorm(z[,k-1])-pnorm(z[,k
        -2])))-Ymat[,k]*dnorm(z[,k-1])/(1-pnorm(z[,k-1]))
179   }
180 }
181
182  c(-apply(grad,2,sum))
183 }
```

```
184
185
186
187 # Predict outcome for a given vector of fit=c(beta,alpha) estimates
188 predict.forward.cumprobit<-function(fit,newx,scale=TRUE,model.select=
      NA) {
189   y<-fit$y
190   x<-fit$x
191   if (is.na(model.select)) model.select=dim(fit$beta)[1]
192   beta<-fit$beta[model.select,]
193   alpha<-fit$alpha[model.select,]
194   k<-length(unique(y))
195   if (identical(newx,x)) {
196     if (scale) {
197       newx<-scale(newx,center=TRUE,scale=TRUE)
198     }
199   } else if (scale) {
200     newx<-rbind(x,newx)
201     newx<-scale(newx,center=TRUE,scale=TRUE)
202     newx<-matrix(newx[-(1:dim(x)[1]),],ncol=dim(x)[2])
203   }
204   levels<-sort(unique(y))
205   z<-matrix(ncol=k-1,nrow=dim(newx)[1])
```

```
206    for (i in 1:(k-1)) {
207      z[,i]<-alpha[i]+newx%*%beta
208   }
209    pi.z<-matrix(ncol=k,nrow=dim(newx)[1])
210    for (i in 1:k) {
211    if (i==1) {
212        pi.z[,i]<-pnorm(z[,i])
213    } else if (i <= k-1) {
214        pi.z[,i]<-pnorm(z[,i]) - pnorm(z[,i-1])
215    } else if (i==k) {
216      pi.z[,i]<-1 - pnorm(z[,i-1])
217    }
218      }
219   class<-levels[apply(pi.z,1,which.max)]
220   class}
221
222
223 #
       #-------------------------------------------------------------------------

224 Code for performing example in chapter 2.2.3
225 #
       #-------------------------------------------------------------------------
```

```
##09/15/2014###
rm(list=ls())
load("phenoinfo.RData") #pheno info for baseline data
load("qz_bcworkspace.RData") # geno info for baseline data
#library(gtools)
#library(reshape2)
#library(lme4)
library(ordinalgmifs)

#get rid of duplicated rows
pheno.info2<-pheno.info[!duplicated(pheno.info$Sid),] #I keep pheno.
    info2 for numeric outcomes
dim(pheno.info2)

pheno.info3<-transform(pheno.info2, Stress =quantcut(pheno.info2$
    TotalPSS, q=seq(0,1,by=0.25),labels=FALSE,na.rm=TRUE),
Anxiety=ifelse(pheno.info2$HadsAxiety<=7,1,ifelse(pheno.info2$
    HadsAxiety <=10 ,2,3)),Depress=ifelse(pheno.info2$HadsDepress
    <=7,1,ifelse(pheno.info2$HadsDepress <=10 ,2,3))
)

```

```
#Filtering process
    ------------------------------------------------------------------------------


#filtering out CpG sites that fully methylated (beta > 0.9) and not
    methylated(beta <0.1) for all samples
ind.full_unmethy<-numeric()
for (i in 1:dim(beta.corrected)[1]){
ind.full_unmethy[i]<-ifelse(sum(beta.corrected[i,]<=0.1 | beta.
    corrected[i,]>=0.9)==73,1,0)
}
table(ind.full_unmethy)
beta.filtered<-beta.corrected[ind.full_unmethy==0,]
#methy.M<-log(beta.filtered/(1-beta.filtered))
methy.M<-beta.filtered
dim(methy.M)

x.genes<-t(methy.M)

#Seperated model
    ------------------------------------------------------------------------------


Anxiety.model<-ordinal.gmifs(Anxiety ~1,x=x.genes,data=pheno.info3,
    epsilon=0.01,probability.model="Cumulative",link="probit",scale=
```

```
      TRUE)
259 Depress.model<-ordinal.gmifs(Depress ~1,x=x.genes,data=pheno.info3,
      epsilon=0.01,probability.model="Cumulative",link="probit",scale=
      TRUE)
260 Stress.model<-ordinal.gmifs(Stress ~1,x=x.genes,data=pheno.info3,
      epsilon=0.01,probability.model="Cumulative",link="probit",scale=
      TRUE)
261
262 save.image("ADS_gmifs1.RData")
263 #-----------------------------------------------
264 load("ADS_gmifs1.RData")
265 ls()
266 library(ordinalgmifs)
267 #AIC model misclassification--------------------
268 pred.an<-predict(Anxiety.model)
269 table(pheno.info3$Anxiety, pred.an$class)
270 pred.dep<-predict(Depress.model)
271 table(pheno.info3$Depress, pred.dep$class)
272 pred.st<-predict(Stress.model)
273 out<-table(pheno.info3$Stress, pred.st$class)
274 (sum(out)-sum(diag(out)))/sum(out)
275
276 #obtain BIC models-------------------------------------------------
```

```
277 an.b<-coef(Anxiety.model, model.select=which.min(Anxiety.model$BIC))
278 dep.b<-coef(Depress.model, model.select=which.min(Depress.model$BIC))
279 st.b<-coef(Stress.model, model.select=which.min(Stress.model$BIC))
280
281 pred.an.bic<-predict(Anxiety.model,model.select=which.min(Anxiety.
      model$BIC))
282 out<-table(pheno.info3$Anxiety, pred.an.bic$class)
283 out
284 pred.dep.bic<-predict(Depress.model,model.select=which.min(Depress.
      model$BIC))
285 out<-table(pheno.info3$Depress, pred.dep.bic$class)
286 out
287 pred.st.bic<-predict(Stress.model,model.select=which.min(Stress.model
      $BIC))
288 out<-table(pheno.info3$Stress, pred.st.bic$class)
289 out
290 (sum(out)-sum(diag(out)))/sum(out)
291
292 Sig.B<-list(Anxiety.BIC=an.b[an.b!=0][-c(1:2)],Depress.BIC=dep.b[dep.
      b!=0][-c(1,2)],Stress.BIC=st.b[st.b!=0][-c(1,2,3)])
293 savehistory(file = "Probit analysis.Rhistory")
```

A.2 GMIFS continuation ratio model code

```
####-------------------------------------------------
Penalized forward continuation ratio model using probit link code
####-------------------------------------------------
FCR_probit.fn<-function(par, x, y, beta) {
  x<-as.matrix(x)
    k <- length(unique(y))
    levels <- sort(unique(y))
    Ymat <- matrix(0, nrow = length(y), ncol = k)
    for (i in levels) {
        Ymat[which(y == i), which(levels == i)] <- 1
  }
    Xb<-x%*%beta
    G.mat <- matrix(0, nrow = length(y), ncol = k)
    G.mat[,1]<-pnorm(par[1]+Xb)
    G.mat[,2]<-pnorm(par[2]+Xb)*(1-G.mat[,1])
    if (k>3) {
    for (i in 3:(k-1)) {
      G.mat[,i]<-pnorm(par[i]+Xb)*(1-matrix(apply(G.mat[,1:(i-1)],1,
          sum),nrow=nrow(G.mat),byrow=T))
    }
  }
  G.mat[,k]<-1-matrix(apply(G.mat[,1:(k-1)],1,sum),nrow=nrow(G.mat),
      byrow=T)
```

```
22     pi <- Ymat*G.mat
23     pi <- apply(pi,1,sum)
24     loglik <- sum(log(pi))
25     -loglik
26 }
27
28
29 ### Forward Continuation Ratio GMIFS function ###
30 fcr_probit.stepwise<-
31 function(x,y,tol=1e-5, epsilon=0.0001, scale=FALSE, step=TRUE) {
32   levels<-sort(unique(y))
33   k<-length(unique(y))
34   x<-as.matrix(x)
35     vars<-dim(x)[2]
36   oldx<-x
37   if (scale) {
38     x<-scale(x,center=TRUE,scale=TRUE)
39   }
40   x<-cbind(x,-1*x)
41   Ymat<-matrix(0,nrow=length(y),ncol=k)
42   for (i in levels){
43     Ymat[which(y==i),which(levels==i)]<-1
44   }
```

```
45    beta <- rep(0, dim(x)[2])
46    names(beta) <- dimnames(x)[[2]]
47
48    alpha<-numeric()
49    tab<-table(y)
50
51  Cum.Ymat<-matrix(0,nrow=nrow(Ymat),ncol=k-1)
52    for(i in 1:(k-1)) {
53        alpha[i]<- dnorm(tab[i]/sum(tab[i:k]))
54    Cum.Ymat[,i]<-apply(matrix(Ymat[,i:k],nrow=dim(Ymat)[1]),1,sum)
55    }
56    names(alpha)<-paste("alpha",1:(k-1),sep=".")
57    step<-0
58    Estimates<-matrix(0,ncol=dim(oldx)[[2]])
59    alpha.update <- matrix(alpha, ncol = k-1)
60    Likelihood<-numeric()
61    AIC<-numeric()
62    repeat {
63        u <- rep(0,dim(x)[[2]])
64        Xb<-x%*%beta
65        eta<-matrix(0,ncol=k-1,nrow=dim(x)[1])
66        for (i in 1:(k-1)) {
67            eta[,i]<-alpha[i] + Xb
```

```
68        }
69            z1<- dnorm(eta)/pnorm(eta)*Ymat[,1:(k-1)] + dnorm(eta)/(1-
                 pnorm(eta))*(Ymat[,1:(k-1)]-Cum.Ymat)
70    u<- -apply(t(x)%*%z1,1,sum)
71        update.value<-min(u)
72        if (update.value<0) {
73            beta[which.min(u)]<- beta[which.min(u)]+epsilon
74        }
75        Estimates<-rbind(Estimates,beta[1:vars]-beta[(vars+1):length(
             beta)])
76        out<-optim(fn=FCR_probit.fn, par=alpha.update[step+1,], x=x
             [,1:vars], y=y, beta=beta[1:vars]-beta[(vars+1):length(beta
             )], method="BFGS")
77    alpha.update <- rbind(alpha.update, out$par)
78            alpha <- out$par
79        p <- sum(Estimates[step+2,]!=0) + length(alpha)
80        Likelihood[step+1]<- LL1<- -out$value
81        AIC[step+1]<-2*p-2*Likelihood[step+1]
82        if (step){
83        print(step)}
84        if ( (step>=1 && LL1-LL0<tol) ) {
85            break
86        }
```

```
87        LL0<-LL1
88        step<-1+step
89     }
90     beta<-Estimates[-1,]
91     alpha<-alpha.update[-1,]
92   model.select<-which.min(AIC)
93     list(beta = beta, alpha = alpha, x=oldx, y=y, scale=scale,
          Likelihood=Likelihood, AIC=AIC, model.select=model.select)
94     }
95
96
97 ### Function to predict class ###
98 predict.FCR_probit<-function(fit,newx,model.select=NA) {
99   x<-fit$x
100  y<-fit$y
101  if (is.na(model.select)) model.select=dim(fit$beta)[1]
102  beta<-fit$beta[model.select,]
103  alpha<-fit$alpha[model.select,]
104  k<-length(unique(y))
105  newx<-as.matrix(newx)
106  if (identical(newx,x)) {
107    if (fit$scale) {
108      newx<-scale(newx,center=TRUE,scale=TRUE)
```

```
109     }
110   } else if (fit$scale) {
111     newx<-rbind(x,newx)
112     newx<-scale(newx,center=TRUE,scale=TRUE)
113     newx<-matrix(newx[-(1:dim(x)[1]),],ncol=dim(x)[2])
114   }
115   levels<-sort(unique(y))
116     Xb<-newx%*%beta
117     pi <- matrix(0, nrow = dim(newx)[1], ncol = k)
118     pi[,1]<-pnorm(alpha[1]+Xb)
119     pi[,2]<-pnorm(alpha[2]+Xb)*(1-pi[,1])
120     if (k>3) {
121     for (i in 3:(k-1)) {
122       pi[,i]<-pnorm(alpha[i]+Xb)*(1-matrix(apply(pi[,1:(i-1)],1,sum),
            nrow=nrow(pi),byrow=T))
123     }
124   }
125   pi[,k]<-1-matrix(apply(pi[,1:(k-1)],1,sum),nrow=nrow(pi),byrow=T)
126   class<-levels[apply(pi,1,which.max)]
127   list(predicted=pi,class=class)
128 }
129
130 ####--------------------------------------------------
```

```
Penalized back continuation ratio model using probit link code
####-------------------------------------------------
backcr_probit.stepwise<-
function(x,y,tol=1e-5, epsilon=0.0001, scale=FALSE,step=TRUE) {
  levels<-sort(unique(y))
  k<-length(unique(y))
  x<-as.matrix(x)
    vars<-dim(x)[2]
  oldx<-x
  if (scale) {
    x<-scale(x,center=TRUE,scale=TRUE)
  }
  x<-cbind(x,-1*x)
  Ymat<-matrix(0,nrow=length(y),ncol=k)
  for (i in levels){
    Ymat[which(y==i),which(levels==i)]<-1
  }
    beta <- rep(0, dim(x)[2])
    names(beta) <- dimnames(x)[[2]]

    alpha<-numeric()
    tab<-table(y)

```

```
Cum.Ymat<-matrix(0,nrow=nrow(Ymat),ncol=k-1)
  for(i in 1:(k-1)) {
      alpha[i]<- dnorm(tab[i+1]/sum(tab[1:(i+1)]))
  Cum.Ymat[,i]<-apply(matrix(Ymat[,(i+1):k],nrow=dim(Ymat)[1]),1,sum
     )
  }
  names(alpha)<-paste("alpha",1:(k-1),sep=".")
  step<-0
  Estimates<-matrix(0,ncol=dim(oldx)[[2]])
  alpha.update <- matrix(alpha, ncol = k-1)
  Likelihood<-numeric()
  AIC<-numeric()
  repeat {
      u <- rep(0,dim(x)[[2]])
      Xb<-x%*%beta
      eta<-matrix(0,ncol=k-1,nrow=dim(x)[1])
      for (i in 1:(k-1)) {
          eta[,i]<-alpha[i] + Xb
      }
          z1<- dnorm(eta)/pnorm(eta)*Ymat[,2:k] + dnorm(eta)/(1-
             pnorm(eta))*(Cum.Ymat-1)
  u<- -apply(t(x)%*%z1,1,sum)
      update.value<-min(u)
```

```
175         if (update.value<0) {
176             beta[which.min(u)]<- beta[which.min(u)]+epsilon
177         }
178         Estimates<-rbind(Estimates,beta[1:vars]-beta[(vars+1):length(
                beta)])
179         out<-optim(fn=BackCR_probit.fn, par=alpha.update[step+1,], x=x
                [,1:vars], y=y, beta=beta[1:vars]-beta[(vars+1):length(beta
                )],method="BFGS")
180     alpha.update <- rbind(alpha.update, out$par)
181             alpha <- out$par
182         p <- sum(Estimates[step+2,]!=0) + length(alpha)
183         Likelihood[step+1]<- LL1<- -out$value
184         AIC[step+1]<-2*p-2*Likelihood[step+1]
185         if (step){
186         print(step)}
187         if ( (step>=1 && LL1-LL0<tol) ) {
188             break
189         }
190         LL0<-LL1
191         step<-1+step
192     }
193     beta<-Estimates[-1,]
194     alpha<-alpha.update[-1,]
```

```
195   model.select<-which.min(AIC)
196     list(beta = beta, alpha = alpha, x=oldx, y=y, scale=scale,
          Likelihood=Likelihood, AIC=AIC, model.select=model.select)
197     }
198
199
200 ### Function to update alpha ###
201 BackCR_probit.fn<-function(par, x, y, beta) {
202   x<-as.matrix(x)
203     k <- length(unique(y))
204     levels <- sort(unique(y))
205     Ymat <- matrix(0, nrow = length(y), ncol = k)
206     for (i in levels) {
207         Ymat[which(y == i), which(levels == i)] <- 1
208   }
209     Xb<-x%*%beta
210     G.mat <- matrix(0, nrow = length(y), ncol = k)
211     G.mat[,k]<-pnorm(par[k-1]+Xb)
212     G.mat[,k-1]<-pnorm(par[k-2]+Xb)*(1-G.mat[,k])
213     if (k>3) {
214     for (i in (k-2):2) {
215       G.mat[,i]<-pnorm(par[i-1]+Xb)*(1-matrix(apply(G.mat[,k:(i+1)],1,
            sum),nrow=nrow(G.mat),byrow=T))
```

```
    }
  }
  G.mat[,1]<-1-matrix(apply(G.mat[,k:2],1,sum),nrow=nrow(G.mat),byrow
      =T)
    pi <- Ymat*G.mat
    pi <- apply(pi,1,sum)
    loglik <- sum(log(pi))
    -loglik
}


### Function to predict class ###
predict.backCR_probit<-function(fit,newx,model.select=NA) {
  x<-fit$x
  y<-fit$y
  if (is.na(model.select)) model.select=dim(fit$beta)[1]
  beta<-fit$beta[model.select,]
  alpha<-fit$alpha[model.select,]
  k<-length(unique(y))
  newx<-as.matrix(newx)
  if (identical(newx,x)) {
    if (fit$scale) {
```

```
238       newx<-scale(newx,center=TRUE,scale=TRUE)
239     }
240   } else if (fit$scale) {
241     newx<-rbind(x,newx)
242     newx<-scale(newx,center=TRUE,scale=TRUE)
243     newx<-matrix(newx[-(1:dim(x)[1]),],ncol=dim(x)[2])
244   }
245   levels<-sort(unique(y))
246     Xb<-newx%*%beta
247     pi <- matrix(0, nrow = dim(newx)[1], ncol = k)
248     pi[,k]<-pnorm(alpha[k-1]+Xb)
249     pi[,k-1]<-pnorm(alpha[k-2]+Xb)*(1-pi[,k])
250     if (k>3) {
251     for (i in (k-2):2) {
252       pi[,i]<-pnorm(alpha[i-1]+Xb)*(1-matrix(apply(pi[,k:(i+1)],1,sum)
              ,nrow=nrow(pi),byrow=T))
253     }
254   }
255   pi[,1]<-1-matrix(apply(pi[,k:2],1,sum),nrow=nrow(pi),byrow=T)
256   class<-levels[apply(pi,1,which.max)]
257   list(predicted=pi,class=class)
258 }
```

AIC model : CpG sites that were significantly associated with Anxiety (Illumina loci ID) : cg22417589 cg07424927 cg17129821 cg10313047 cg15060599 cg18005693 cg20449692 cg03827835 cg13619597 cg22056094 cg26630791 cg15262505 cg13679303 cg27615378 cg27332938 cg02072400 cg16094511 cg12485185 cg00049664 cg09955084 cg08900396 cg15250633 cg08888354 cg09759458 cg17399362 cg01943289 cg00375105 cg01044189 cg11152528 cg18412777 cg21471515 cg25832529 cg07501029 cg16298457 cg17443007 cg01267068 cg02058002 cg16277214 cg17336044 cg22532079 cg05350396 cg05237015 cg02734955 cg19685567 cg23169584 cg23684218 cg05483021 cg05738743 cg17795240 cg19049724 cg13717350 cg14090916 cg14223966 cg12566078 cg16871435 cg23260525 cg06730161 cg15001406 cg06722407 cg17809365 cg26657240 cg14556515 cg20985587 cg03234186 cg03832839 cg18917378 cg00192046

CpG sites that were significantly associated with Depression (Illumina loci ID) : cg00378717 cg00147788 cg20399616 cg24394624 cg10043663 cg17336044 cg03049125 cg03091070 cg19748937 cg18873166 cg05453820 cg13781956 cg19683821 cg25542438 cg20418308 cg14516632 cg06771839 cg04932840 cg19913465

CpG sites that were significantly associated with Stress (Illumina loci ID) : cg21566642 cg22040631 cg13619597 cg22307444 cg10758057 cg10174864 cg00324161 cg21121843 cg19755435 cg26889118

BIC model : CpG sites that were significantly associated with Anxiety (Illumina loci ID) : cg15060599

CpG sites that were significantly associated with Depression (Illumina loci ID) : cg24394624 cg03091070 cg13781956 cg19683821 cg25542438 cg20418308 cg19913465

CpG sites that were significantly associated with Stress (Illumina loci ID) : cg21566642 cg22040631 cg13619597 cg22307444 cg10758057 cg10174864 cg00324161 cg21121843 cg19755435 cg26889118

A.3 GMIFS stereotype logit model with example code

```
### sterotype model with fixed beta
stereo.fn<-function (par, xmatrix, y,beta)
{
    k <- length(unique(y)) # there are k levels
    levels <- sort(unique(y))
    Ymat <- matrix(0, nrow = length(y), ncol = k) # create a matrix of
        i * k, yi1=1 if control else 0, yi2=1 if incipient else 0...
    for (i in levels) {
        Ymat[which(y == i), which(levels == i)] <- 1
    }
    Xb<-xmatrix%*%beta
    eta<-matrix(0,ncol=k-1,nrow=dim(xmatrix)[1])
    eta[,1] <- exp(par[1] + Xb)
  for (i in 2:(k-1)) {
      eta[,i]<- exp(par[i] + par[i+k-2]*Xb)
    }
```

```
16   pik<- 1- apply(eta,1,sum)/(1+apply(eta,1,sum))
17
18
19   pi<-matrix(0,ncol=k,nrow=dim(xmatrix)[1])
20   pi[,k]<- pik
21   pi[,1:(k-1)]<-eta*pik
22    loglik<-sum(apply(Ymat*log(pi),1,sum))
23   -loglik
24 }
25
26 stereo.stepwise<-function(xmatrix,y,tol=1e-5,epsilon=0.001, scale=
       FALSE) {
27    levels<-sort(unique(y))
28    k<-length(unique(y))
29    x<-as.matrix(xmatrix)
30    vars<-dim(x)[2]
31    oldx<-x
32    if (scale) {
33        x<-scale(x,center=TRUE,scale=TRUE)
34    }
35    x<-cbind(x,-1*x)
36    Ymat<-matrix(0,nrow=length(y),ncol=k)
37    for (i in levels){
```

```
38        Ymat[which(y==i),which(levels==i)]<-1
39    }
40
41    #initialize beta
42    beta <- rep(0, dim(x)[2])
43    names(beta) <- dimnames(x)[[2]]
44
45    #initialize alpha
46    alpha<-numeric()
47    pi.0 <- table(y)/length(y)
48    for(i in 1:(k-1)) {
49      alpha[i]<- log(pi.0[i]/pi.0[k])
50    }
51    names(alpha)<-paste("alpha",1:(k-1),sep=".")
52
53    #initialize phi
54    phi <- c(1,rep(0.1,k-2))
55    names(phi) <- paste("phi",1:(k-1),sep=".")
56
57    step<-0
58    Estimates<-matrix(0,ncol=dim(oldx)[[2]])
59    alpha.update <- matrix(alpha, ncol = k-1) # 1*3 matrix for updated
          alpha
```

```
60    phi.update <- matrix(phi,ncol=k-1)
61
62    Likelihood<-numeric()
63    AIC<-numeric()
64    repeat {
65      u <- rep(0,dim(x)[[2]])
66        Xb<-x%*%beta
67
68      eta<-matrix(0,ncol=k-1,nrow=dim(x)[1])
69      for (i in 1:(k-1)) {
70           eta[,i]<- exp(alpha[i] + phi[i]*Xb)
71        }
72        denom<- 1+apply(eta,1,sum)
73        numer<-matrix(0,ncol=k-1,nrow=dim(x)[1])
74        for (i in 1:(k-1)){
75          numer[,i]<-eta[,i]*phi[i]
76        }
77        numer2<- -apply(numer,1,sum)*Ymat[,k]/denom #contribution to
             log-like for class K
78        numer1<-matrix(0,ncol=k-1,nrow=dim(x)[1]) # contribution to
             log-like for classes 1 to K-1
79        for (i in 1:(k-1)){
80          numer1[,i]<-Ymat[,i]*(phi[i]-apply(numer,1,sum)/denom)
```

```
81        }
82        numer1f<- apply(numer1,1,sum)
83        dll<-x*(numer1f + numer2)
84        u<-apply(-dll,2,sum) # u is a 2p*1 vector of -dlogL/dbeta
85    update.value<-min(u)
86        if (update.value<0) {
87            beta[which.min(u)]<- beta[which.min(u)]+epsilon
88        }
89        Estimates<-rbind(Estimates,beta[1:vars]-beta[(vars+1):length(
              beta)]) #?
90        #out<-optim(par=c(alpha.update[step+1,],phi.update[step+1,2:
              dim(phi.update)[2]]), stereo.fn, x=x[,1:vars], y=y, beta=
              beta[1:vars]-beta[(vars+1):length(beta)], method="BFGS")
91          out<-optim(par=c(alpha.update[step+1,],phi.update[step+1,2:
                dim(phi.update)[2]]), stereo.fn, x=x[,1:vars], y=y, beta=
                beta[1:vars]-beta[(vars+1):length(beta)],
92        method="L-BFGS-B",upper=c(rep(Inf,k-1),rep(1,k-2)),lower=c(rep
              (-Inf,k-1),rep(0,k-2)))
93        alpha.update <- rbind(alpha.update, (out$par)[1:k-1])
94      phi.update <- rbind(phi.update, c(1,out$par[k:length(out$par)]))
95        alpha <- out$par[1:k-1]
96    phi <- c(1,out$par[k:length(out$par)])
```

```
97        p <- sum(Estimates[step+2,]!=0) + length(alpha) + length(phi)
              # ?
98        Likelihood[step+1]<- LL1<- -out$value
99        AIC[step+1]<-2*p-2*Likelihood[step+1]
100       print(step)
101       if (step >= 1 && LL1 - LL0 < tol) {
102           break
103       }
104       LL0<-LL1
105       step<-1+step
106   }
107   beta<-Estimates[-1,]
108   alpha<-alpha.update[-1,]
109   phi<-phi.update[-1,]
110       model.select<-which.min(AIC)
111   list(beta = beta, alpha = alpha, phi = phi, x=oldx, y=y, scale=
          scale, Likelihood=Likelihood, AIC=AIC, model.select=model.
          select)
112   }
113
114
115
116 ### Function to predict class ###
```

```
117 predict.stereo<-function(fit,newx,model.select=NA) {
118   x<-fit$x
119   y<-fit$y
120   if (is.na(model.select)) model.select=dim(fit$beta)[1] # if no
        model.select, using the last iteration
121   beta<-fit$beta[model.select,]
122   alpha<-fit$alpha[model.select,]
123      phi<-fit$phi[model.select,]
124   k<-length(unique(y))
125   newx<-as.matrix(newx)
126   if (identical(newx,x)) {
127     if (fit$scale) {
128       newx<-scale(newx,center=TRUE,scale=TRUE)
129     }
130   } else if (fit$scale) {
131     newx<-rbind(x,newx)
132     newx<-scale(newx,center=TRUE,scale=TRUE)
133     newx<-matrix(newx[-(1:dim(x)[1]),],ncol=dim(x)[2])
134   }
135   levels<-sort(unique(y))
136   eta<-matrix(0,ncol=k-1,nrow=dim(newx)[1])
137   Xb <- newx%*%beta
138   for (i in 1:(k-1)) {
```

```
139     eta[,i]<-exp(alpha[i] + phi[i]*Xb)
140   }
141 
142       pik<- 1- apply(eta,1,sum)/(1+apply(eta,1,sum))
143 
144 
145   pi<-matrix(0,ncol=k,nrow=dim(newx)[1])
146   pi[,k]<- pik
147   pi[,1:(k-1)]<-eta*pik
148 
149     class<-levels[apply(pi,1,which.max)]
150     list(predicted=pi,class=class)
151 }
152 
153 
154 load("GSE1297.RData")
155 class<-sorted.pheno$y
156 x<-log(x,2)
157 x<-scale(x,center=TRUE,scale=TRUE)
158 
159 
160 Alz.stereo<-stereo.stepwise(xmatrix=x,y=class)
```

```
161 fit.class<-predict.stereo(Alz.stereo,newx=x,model.select=Alz.stereo$
       model.select)
162 table(fit.class$class,class)
163 full.class<-predict.stereo(Alz.stereo,newx=x,model.select=dim(Alz.
       stereo$beta)[1])
164 table(full.class$class,class)
165 save.image("stereo1297.RData")
```

Appendix B

Chapter 3 appendix

B.1 Simulation code

```
#---------------------------------------
#Simulation 1 code-----------------------
#---------------------------------------
n.sim<-200
set.seed(125)
x.var<-matrix(nr=n.sim,ncol=5)
for (i in 1:5){
x.var[,i]<-rnorm(n.sim,0,1)
}


```

```
alpha1=-1
alpha2=2

beta1<-3
beta2<-1

logit1<-alpha1+beta1*x.var[,1]+beta2*x.var[,2]
logit2<-alpha2+beta1*x.var[,1]+beta2*x.var[,2]

G<-function(z){
 exp(z)/(1+exp(z))
}

p1<-G(logit1)
p2<-G(logit2)

tmp <- runif(n.sim)
y <- 4-((tmp < p1) + (tmp < p2) + (tmp<1))
y
table(y)

simu.x<-scale(x.var,center=TRUE,scale=TRUE)
simu.y<-y
```

```
35
36 ##vglm
37 library(VGAM)
38 vglm.simu<-vglm(simu.y~ simu.x[,1]+simu.x[,2] + simu.x[,3]+simu.x[,4]
       + simu.x[,5], family=cumulative(parallel=T,reverse=F))
39 vglm.pi<-predict(vglm.simu,type="response")
40 vglm.class<-apply(vglm.pi,1,which.max)
41 out<-table(simu.y, vglm.class)
42 1-sum(diag(out))/sum(out)
43 summary(vglm.simu)
44
45 #ordinalgmifs
46 library(ordinalgmifs)
47 penal.ord.simu<-ordinal.gmifs(simu.y~1, x=simu.x, data=data.frame(
      simu.x))
48 summary(penal.ord.simu)
49
50
51 #Bayesian
52 library(rjags)
53 library(R2jags)
54 k<-length(unique(simu.y))
55 pi.0 <- table(simu.y)/length(simu.y)
```

```
alpha.0 <- log(cumsum(pi.0)/(1 - cumsum(pi.0)))[1:(k - 1)]
simu.inits1<-list(tau=alpha.0,beta=c(rep(0,5)))
simu.inits2<-list(tau=c(-1,1),beta=c(rnorm(5,0,0.01)))
simu.inits3<-list(tau=c(-2,2),beta=c(rnorm(5,0,0.001)))
simu.inits<-list(simu.inits1,simu.inits2,simu.inits3)
simu.data<-list(Y=simu.y,X=as.matrix(simu.x),N=length(simu.y),nb=5,k=
    k)
simu.params<-c("tau","beta","gamma","lambda","bgamma")

simujags <- jags(data=simu.data, inits=simu.inits, simu.params,
n.iter=10000, model.file="Non_info_model.txt",n.chains=3,n.thin=2)
simujags.upd <-autojags(simujags,n.thin=2,n.iter=5000,Rhat=1.1,n.
    update=4)
print(simujags.upd)
range(simujags.upd$BUGSoutput$summary[,"Rhat"])


#####traceplot#####

traceplot.qz<-function(x,param,v.name){
x <- x$BUGSoutput
n.chain <- x$n.chains
n.keep <- x$n.keep
```

```
77 bugs.array <- x$sims.array
78 range.x <- c(1, n.keep)
79
80 range.y <- range(bugs.array[, , param])
81 plot(range.x, range.y, type = "n", main = v.name, xlab = "iteration",
       ylab = v.name, xaxt = "n",
82               xaxs = "i")
83
84 col = rainbow(x$n.chains)
85 for (i in 1:n.chain) {
86               x.cord <- 1:n.keep
87                y.cord <- bugs.array[, i, param]
88                lines(x.cord, y.cord, col = col[i], lty = 1,
89                lwd = 1)
90            }
91 abline(h=x$summary[param,"mean"],lwd=2)
92 mtext(paste("Mean=",round(x$summary[param,"mean"],2)),side=4,at=x$
      summary[param,"mean"],line=0.5)
93 axis(1, at = seq(0, n.keep, n.keep * 0.1), tick = TRUE)
94 }
95
96
97 par(mfrow=c(2,1))
```

```
98 traceplot.qz(x=simujags.upd,param="tau[1]",v.name=expression(alpha
       [1]))
99 traceplot.qz(x=simujags.upd,param="tau[2]",v.name=expression(alpha
       [2]))
100
101
102 par(mfrow=c(3,2))
103 traceplot.qz(x=simujags.upd,param="beta[1]",v.name=expression(beta
       [1]))
104 traceplot.qz(x=simujags.upd,param="beta[2]",v.name=expression(beta
       [2]))
105 traceplot.qz(x=simujags.upd,param="beta[3]",v.name=expression(beta
       [3]))
106 traceplot.qz(x=simujags.upd,param="beta[4]",v.name=expression(beta
       [4]))
107 traceplot.qz(x=simujags.upd,param="beta[5]",v.name=expression(beta
       [5]))
108
109 par(mfrow=c(3,2))
110 traceplot.qz(x=simujags.upd,param="bgamma[1]",v.name=expression(gamma
       [1]*beta[1]))
111 traceplot.qz(x=simujags.upd,param="bgamma[2]",v.name=expression(gamma
       [2]*beta[2]))
```

```
112 traceplot.qz(x=simujags.upd,param="bgamma[3]",v.name=expression(gamma
      [3]*beta[3]))
113 traceplot.qz(x=simujags.upd,param="bgamma[4]",v.name=expression(gamma
      [4]*beta[4]))
114 traceplot.qz(x=simujags.upd,param="bgamma[5]",v.name=expression(gamma
      [5]*beta[5]))
115
116
117
118
119 rm(list=ls())
120
121 #-----------------------------------------------
122 #Simulation 2 codes-------------------------------
123 #-----------------------------------------------
124
125 #Simulation for frequentist GMIFS
      ------------------------------------
126 library(ordinalgmifs)
127 n.sim<-90
128 p.sim<-100
129
130 gmifs.coef<-list()
```

```
seed.se<-seq(1234,123456,by=120)
for (j in 1:100){
      set.seed(seed.se[[j]])
  x.var<-matrix(nr=n.sim,ncol=p.sim)
      for (i in 1:5){
  x.var[,i]<-c(rnorm(30,0,0.4),rnorm(30,1,0.4),rnorm(30,2,0.4))}


      for (i in 6:p.sim){
  x.var[,i]<-rnorm(n.sim,0,0.4)
  }


      colnames(x.var)<-paste0("x",seq(1,p.sim))


  simu.x<-scale(x.var,center=TRUE,scale=TRUE)
  simu.y<-c(rep(1,30),rep(2,30),rep(3,30))
      table(simu.y)
      head(simu.x)

  penal.ord.simu<-ordinal.gmifs(simu.y~1, x=simu.x, data=data.frame(
     simu.x),
```

```
153   epsilon = 0.001,verbose=TRUE)
154   summary(penal.ord.simu)
155   coefficients<-coef(penal.ord.simu)
156   gmifs.coef[[j]]<-coefficients[coefficients!=0]
157 }
158
159 save.image("gmifs100_01132016.RData")
160
161 #Simulation for Bayesian with non-info \alpha ----------------------
162 library(rjags)
163 library(R2jags)
164 n.sim<-90
165 p.sim<-100
166 num<-100
167 simujags.upd<-list()
168 seed.se<-seq(1234,123456,by=120)
169 for (j in 1:100){
170         set.seed(seed.se[j])
171   x.var<-matrix(nr=n.sim,ncol=p.sim)
172         for (i in 1:5){
173   x.var[,i]<-c(rnorm(30,0,0.4),rnorm(30,1,0.4),rnorm(30,2,0.4))}
174
175
```

```
176         for (i in 6:p.sim){
177   x.var[,i]<-rnorm(n.sim,0,0.4)
178   }
179
180
181         colnames(x.var)<-paste0("x",seq(1,p.sim))
182
183
184   simu.x<-scale(x.var,center=TRUE,scale=TRUE)
185   simu.y<-c(rep(1,30),rep(2,30),rep(3,30))
186         table(simu.y)
187         head(simu.x)
188
189 k<-length(unique(simu.y))
190 pi.0 <- table(simu.y)/length(simu.y)
191 alpha.0 <- log(cumsum(pi.0)/(1 - cumsum(pi.0)))[1:(k - 1)]
192 simu.inits1<-list(tau=alpha.0,beta=c(rep(0,num)))
193 simu.inits2<-list(tau=c(-1,1),beta=c(rep(0,num)))
194 simu.inits<-list(simu.inits1,simu.inits2)
195 simu.data<-list(Y=simu.y,X=as.matrix(simu.x),N=length(simu.y),nb=num,
      k=k)
196 simu.params<-c("tau","beta","gamma")
197
```

```
198 simujags <- jags(data=simu.data, inits=simu.inits, simu.params,
199 n.iter=10000, model.file="Non_info_model.txt",n.chains=2,n.thin=16)
200 simujags.upd[[j]] <-autojags(simujags,n.thin=16,n.iter=5000,Rhat=1,n.
       update=4)$BUGSoutput$summary
201
202 }
203
204 save.image("bayes100_flat.RData")
205
206
207 #Simulation with shrink \lambda, non-info \alpha
       ------------------------------------------
208 library(rjags)
209 library(R2jags)
210 n.sim<-90
211 p.sim<-100
212 num<-100
213 simujags.upd<-list()
214 seed.se<-seq(1234,123456,by=120)
215 for (j in 1:100){
216         set.seed(seed.se[j])
217   x.var<-matrix(nr=n.sim,ncol=p.sim)
218         for (i in 1:5){
```

```
  x.var[,i]<-c(rnorm(30,0,0.4),rnorm(30,1,0.4),rnorm(30,2,0.4))}


      for (i in 6:p.sim){
  x.var[,i]<-rnorm(n.sim,0,0.4)
  }


      colnames(x.var)<-paste0("x",seq(1,p.sim))


  simu.x<-scale(x.var,center=TRUE,scale=TRUE)
  simu.y<-c(rep(1,30),rep(2,30),rep(3,30))
      table(simu.y)
      head(simu.x)

k<-length(unique(simu.y))
pi.0 <- table(simu.y)/length(simu.y)
alpha.0 <- log(cumsum(pi.0)/(1 - cumsum(pi.0)))[1:(k - 1)]
simu.inits1<-list(tau=alpha.0,beta=c(rep(0,num)))
simu.inits2<-list(tau=c(-1,1),beta=c(rep(0,num)))
simu.inits<-list(simu.inits1,simu.inits2)
```

```
241 simu.data<-list(Y=simu.y,X=as.matrix(simu.x),N=length(simu.y),nb=num,
      k=k)
242 simu.params<-c("tau","beta","gamma")
243
244 simujags <- jags(data=simu.data, inits=simu.inits, simu.params,
245 n.iter=10000, model.file="Non_info_model2.txt",n.chains=2,n.thin=16)
246 simujags.upd[[j]] <-autojags(simujags,n.thin=16,n.iter=5000,Rhat=1,n.
      update=4)$BUGSoutput$summary
247
248 }
249
250 save.image("bayes100_shrink.RData")
251
252 #Simulation with informative \alphas ----------------------
253 library(rjags)
254 library(R2jags)
255 n.sim<-90
256 p.sim<-100
257 num<-100
258 simujags.upd<-list()
259 seed.se<-seq(1234,123456,by=120)
260 for (j in 1:100){
261       set.seed(seed.se[j])
```

```
262  x.var<-matrix(nr=n.sim,ncol=p.sim)
263       for (i in 1:5){
264  x.var[,i]<-c(rnorm(30,0,0.4),rnorm(30,1,0.4),rnorm(30,2,0.4))}
265
266
267       for (i in 6:p.sim){
268  x.var[,i]<-rnorm(n.sim,0,0.4)
269  }
270
271
272       colnames(x.var)<-paste0("x",seq(1,p.sim))
273
274
275  simu.x<-scale(x.var,center=TRUE,scale=TRUE)
276  simu.y<-c(rep(1,30),rep(2,30),rep(3,30))
277       table(simu.y)
278       head(simu.x)
279
280 k<-length(unique(simu.y))
281 pi.0 <- table(simu.y)/length(simu.y)
282 alpha.0 <- log(cumsum(pi.0)/(1 - cumsum(pi.0)))[1:(k - 1)]
283
```

```
284 simu.inits1<-list(tau1=alpha.0[1],beta=c(rep(0,100)),Diff=alpha.0[2]-
      alpha.0[1])
285 simu.inits2<-list(tau1=-1,beta=c(rep(0,100)),Diff=2)
286 simu.inits3<-list(tau1=-2,beta=c(rep(0,100)),Diff=4)
287 simu.inits<-list(simu.inits1,simu.inits2,simu.inits3)
288 simu.data<-list(Y=simu.y,X=as.matrix(simu.x),N=length(simu.y),nb=num,
      k=k,a=alpha.0[1])
289 simu.params<-c("tau1","Diff","beta","gamma")
290
291
292 simujags <- jags(data=simu.data, inits=simu.inits, simu.params,
293 n.iter=5000, model.file="Info_model.txt",n.chains=3,n.thin=16)
294 simujags.upd[[j]] <-autojags(simujags,n.thin=16,n.iter=2000,Rhat=1,n.
      update=4)$BUGSoutput$summary
295
296 }
297
298 save.image("bayes100_info.RData")
299
300
301 #Data loading-------------------------------------
302 #load bayesian model with noninformative alpha prior and gamma (1,1)
      lambda prior
```

```
303 load("bayes100_flat.RData")
304 post.flat<-simujags.upd
305 rm(list=ls()[ls()!="post.flat"])
306
307 #load bayesian model with informative alpha prior and gamma (1,1)
      lambda prior
308 load("bayes100_info.RData")
309 post.info<-simujags.upd
310 rm(list=ls()[!ls()%in%c("post.flat","post.info")])
311
312 #load bayesian model with flat alpha prior but different gamma
      (0.025,0.05) lambda prior
313 load("bayes100_shrink.RData")
314 post.shrink<-simujags.upd
315 rm(list=ls()[!ls()%in%c("post.flat","post.info","post.shrink")])
316
317 #load gmifs model
318 load("gmifs100_01132016.RData")
319 rm(list=ls()[!ls()%in%c("post.flat","post.info","post.shrink","gmifs.
      coef")])
320
321
322 #analysis-------------------------------------
```

```
323 select.threshold<-0.5
324 freq.alpha<-matrix(0,nc=2,nr=100)
325 freq.beta<-matrix(0,nc=100,nr=100)
326 freq.table<-matrix(0,nc=100,nr=100)
327
328
329 flat.alpha<-matrix(nr=100,nc=2)
330 flat.beta<-matrix(nr=100,nc=100)
331 flat.table<-matrix(0,nr=100,nc=100)
332
333 info.alpha<-matrix(nr=100,nc=2)
334 info.beta<-matrix(nr=100,nc=100)
335 info.table<-matrix(0,nr=100,nc=100)
336
337 shrink.alpha<-matrix(nr=100,nc=2)
338 shrink.beta<-matrix(nr=100,nc=100)
339 shrink.table<-matrix(0,nr=100,nc=100)
340
341
342 alpha.ind<-grep("tau",names(post.flat[[1]][,"mean"]))
343 beta.ind<-grep("beta",names(post.flat[[1]][,"mean"]))
344 gamma.ind<-grep("gamma",names(post.flat[[1]][,"mean"]))
345
```

```
alpha.ind2<-c(grep("Diff",names(post.info[[1]][,"mean"])),grep("tau1"
    ,names(post.info[[1]][,"mean"])))
beta.ind2<-grep("beta",names(post.info[[1]][,"mean"]))
gamma.ind2<-grep("gamma",names(post.info[[1]][,"mean"]))


for(i in 1:100){
flat.alpha[i,]<-post.flat[[i]][,"mean"][alpha.ind]
flat.beta[i,]<-post.flat[[i]][,"mean"][beta.ind]
flat.table[i,which(post.flat[[i]][,"mean"][gamma.ind]>=select.
    threshold)]<-1

shrink.alpha[i,]<-post.shrink[[i]][,"mean"][alpha.ind]
shrink.beta[i,]<-post.shrink[[i]][,"mean"][beta.ind]
shrink.table[i,which(post.shrink[[i]][,"mean"][gamma.ind]>=select.
    threshold)]<-1

info.alpha[i,]<-post.info[[i]][,"mean"][alpha.ind2]
info.beta[i,]<-post.info[[i]][,"mean"][beta.ind2]
info.table[i,which(post.info[[i]][,"mean"][gamma.ind2]>=select.
    threshold)]<-1

freq.alpha[i,]<-gmifs.coef[[i]][1:2]
```

```
freq.beta.ind<-paste("x",seq(1:100),sep="") %in% names(gmifs.coef[[i
    ]][-(1:2)])
freq.beta[i,freq.beta.ind]<- gmifs.coef[[i]][-c(1:2)]
freq.table[i,freq.beta.ind]<-1
}

flat.true<-apply(flat.table[,c(1,2,3,4,5)],1,sum)
info.true<-apply(info.table[,c(1,2,3,4,5)],1,sum)
shrink.true<-apply(shrink.table[,c(1,2,3,4,5)],1,sum)
freq.true<-apply(freq.table[,c(1,2,3,4,5)],1,sum)
summary(flat.true)
summary(info.true)
summary(shrink.true)
summary(freq.true)

flat.false<-apply(flat.table[,-c(1,2,3,4,5)],1,sum)
info.false<-apply(info.table[,-c(1,2,3,4,5)],1,sum)
shrink.false<-apply(shrink.table[,-c(1,2,3,4,5)],1,sum)
freq.false<-apply(freq.table[,-c(1,2,3,4,5)],1,sum)
summary(flat.false)
summary(info.false)
summary(shrink.false)
summary(freq.false)
```

B.2 Application code

```
#-----------------------------------------------------
#HCC data-----------------------------------------------
#-----------------------------------------------------
load("hccCancerPanel.RData")
library(rjags)
library(R2jags)

k<-length(unique(hccCancerPanel$Tissue))
pi.0 <- table(hccCancerPanel$Tissue)/length(hccCancerPanel$Tissue)
alpha.0 <- log(cumsum(pi.0)/(1 - cumsum(pi.0)))[1:(k - 1)]
hcc.inits1<-list(tau=alpha.0,beta=c(rnorm(1469,0.01)))
hcc.inits2<-list(tau=c(-1,1),beta=c(rnorm(1469,0.0001)))
hcc.inits3<-list(tau=c(-2,2),beta=c(rnorm(1469,0.1)))
hcc.inits<-list(hcc.inits1,hcc.inits2,hcc.inits3)
hcc.params<-c("tau","beta","gamma")
hcc.data<-list(Y=hccCancerPanel$Tissue,X=scale(as.matrix(
    hccCancerPanel[,-1]),center=TRUE,scale=TRUE),N=length(
    hccCancerPanel$Tissue),nb=1469,k=k)
hcc.jags <- jags(data=hcc.data, inits=hcc.inits, hcc.params,
n.iter=100000, model.file="Non_info_model.txt",n.chains=3)
hcc.upd <- autojags(hcc.jags,n.iter=10000)
```

```
20
21 save.image("hcc_bay0_5_03232016.RData")
22
23 load("hcc_bay0_5_03232016.RData")
24 result<-hcc.jags$BUGSoutput$summary
25
26 #Obtain posterior slopes
27 beta.ind<-grep("bgamma",names(result[,"mean"]))
28 beta.value<-result[,"mean"][beta.ind]
29
30 beta.ind2<-grep("beta",names(result[,"mean"]))
31 beta.value2<-result[,"mean"][beta.ind2]
32
33 #Obtain posterior binary indicator
34 gamma.ind0<-grep("gamma",names(result[,"mean"]))
35 gamma.ind<-gamma.ind0[gamma.ind0%in%beta.ind==FALSE]
36 gamma.value<-result[,"mean"][gamma.ind]
37 hist(gamma.value,breaks=100, xlab=expression(paste(gamma, " value")),
       main=expression(paste("Histogram of ", gamma, " value")))
38
39 #Obtain posterior intercepts
40 alpha.ind<-grep("tau1",names(result[,"mean"]))
41 diff.ind<-grep("Diff",names(result[,"mean"]))
```

```
alpha.value<-c(result[,"50%"][alpha.ind],result[,"50%"][alpha.ind]+
    result[,"50%"][diff.ind])


genename<-colnames(hccCancerPanel[,-1])
threshold<-mean(result[,"mean"][gamma.ind])+3*sd(result[,"mean"][
    gamma.ind])
sig.ind<-which(result[,"mean"][gamma.ind]>=threshold)


slope.coef<-beta.value[sig.ind]
names(slope.coef)<-genename[sig.ind]

freq.sig<-c("CDKN2B_seq_50_S294_F","DDIT3_P1313_R","ERN1_P809_R","GML
    _E144_F","HDAC9_P137_R","HLA.DPA1_P205_R","HOXB2_P488_R","IL16_
    P226_F",
"IL16_P93_R","IL8_P83_F","MPO_E302_R","MPO_P883_R","PADI4_P1158_R","
    SOX17_P287_R","TJP2_P518_F")
table(genename[sig.ind]%in%freq.sig)

#Prediction for misclassification error
    ---------------------------------------
```

```
#refit a non-penalized cumulative logit model with only selected features
x.select<-scale(as.matrix(hccCancerPanel[,-1]),center=TRUE,scale=TRUE)[,sig.ind]

library(rjags)
library(R2jags)
#with non-informative priors
k<-length(unique(hccCancerPanel$Tissue))
pi.0 <- table(hccCancerPanel$Tissue)/length(hccCancerPanel$Tissue)
alpha.0 <- log(cumsum(pi.0)/(1 - cumsum(pi.0)))[1:(k - 1)]
pred.inits1<-list(tau=alpha.0,beta=c(rep(0,dim(x.select)[2])))
pred.inits2<-list(tau=c(-1,1),beta=c(rnorm(dim(x.select)[2],0,0.001)))
pred.inits3<-list(tau=c(-2,2),beta=c(rnorm(dim(x.select)[2],0,0.01)))
pred.inits<-list(pred.inits1,pred.inits2,pred.inits3)
pred.params<-c("tau","beta","Y.pred")
pred.data<-list(Y=hccCancerPanel$Tissue,X=x.select,N=length(hccCancerPanel$Tissue),nb=dim(x.select)[2],k=k)
pred.jags <- jags(data=pred.data, inits=pred.inits, pred.params,
n.iter=2000, model.file="Reg_pred.txt",n.chains=3)
pred.upd <- autojags(pred.jags)$BUGSoutput$summary

```

```
y.ind<-grep("Y.pred",names(pred.upd[,"50%"]))
y.pred<-pred.upd[y.ind,"50%"]
out<-table(hccCancerPanel$Tissue,y.pred)
1-sum(diag(out))/sum(out)

#with informative priors
rm(list=ls()[ls()=="out"])
info.inits1<-list(tau1=alpha.0[1],beta=c(rep(0,dim(x.select)[2])),
    Diff=alpha.0[2]-alpha.0[1])
info.inits2<-list(tau1=-1,beta=c(rnorm(dim(x.select)[2],0,0.01)),Diff
    =2)
info.inits3<-list(tau1=-2,beta=c(rnorm(dim(x.select)[2],0,0.001)),
    Diff=4)
info.inits<-list(info.inits1,info.inits2,info.inits3)
info.data<-list(Y=hccCancerPanel$Tissue,X=x.select,N=length(
    hccCancerPanel$Tissue),nb=dim(x.select)[2],k=k,a=alpha.0[1])
info.params<-c("tau1","Diff","beta","Y.pred")
info.jags <- jags(data=info.data, inits=info.inits, info.params,
n.iter=2000, model.file="Reg_pred_info.txt",n.chains=3)
info.upd <- autojags(info.jags)
y.ind<-grep("Y.pred",names(info.upd$BUGSoutput$summary[,"50%"]))
y.pred<-info.upd$BUGSoutput$summary[y.ind,"50%"]
out<-table(hccCancerPanel$Tissue,y.pred)
```

```
1-sum(diag(out))/sum(out)


#refit a penalized cumulative logit model with only selected features
    --------------------------------
x.select<-scale(as.matrix(hccCancerPanel[,-1]),center=TRUE,scale=TRUE
    )[,sig.ind]

library(rjags)
library(R2jags)
#non-informative priors
k<-length(unique(hccCancerPanel$Tissue))
pi.0 <- table(hccCancerPanel$Tissue)/length(hccCancerPanel$Tissue)
alpha.0 <- log(cumsum(pi.0)/(1 - cumsum(pi.0)))[1:(k - 1)]
pred.inits1<-list(tau=alpha.0,beta=c(rep(0,dim(x.select)[2])))
pred.inits2<-list(tau=c(-1,1),beta=c(rnorm(dim(x.select)[2],0,0.001))
    )
pred.inits3<-list(tau=c(-2,2),beta=c(rnorm(dim(x.select)[2],0,0.01)))
pred.inits<-list(pred.inits1,pred.inits2,pred.inits3)
pred.params<-c("tau","beta","Y.pred")
pred.data<-list(Y=hccCancerPanel$Tissue,X=x.select,N=length(
    hccCancerPanel$Tissue),nb=dim(x.select)[2],k=k)
pred.jags <- jags(data=pred.data, inits=pred.inits, pred.params,
```

```
n.iter=2000, model.file="DE_pred.txt",n.chains=3)
pred.upd <- autojags(pred.jags)$BUGSoutput$summary

y.ind<-grep("Y.pred",names(pred.upd[,"50%"]))
y.pred<-pred.upd[y.ind,"50%"]
out<-table(hccCancerPanel$Tissue,y.pred)
1-sum(diag(out))/sum(out)

#informative priors
rm(list=ls()[ls()=="out"])
info.inits1<-list(tau1=alpha.0[1],beta=c(rep(0,dim(x.select)[2])),
    Diff=alpha.0[2]-alpha.0[1])
info.inits2<-list(tau1=-1,beta=c(rnorm(dim(x.select)[2],0,0.01)),Diff
    =2)
info.inits3<-list(tau1=-2,beta=c(rnorm(dim(x.select)[2],0,0.001)),
    Diff=4)
info.inits<-list(info.inits1,info.inits2,info.inits3)
info.data<-list(Y=hccCancerPanel$Tissue,X=x.select,N=length(
    hccCancerPanel$Tissue),nb=dim(x.select)[2],k=k,a=alpha.0[1])
info.params<-c("tau1","Diff","beta","Y.pred")
info.jags <- jags(data=info.data, inits=info.inits, info.params,
n.iter=2000, model.file="DE_pred_info.txt",n.chains=3)
info.upd <- autojags(info.jags)
```

```
y.ind<-grep("Y.pred",names(info.upd$BUGSoutput$summary[,"50%"]))
y.pred<-info.upd$BUGSoutput$summary[y.ind,"50%"]
out<-table(hccCancerPanel$Tissue,y.pred)
1-sum(diag(out))/sum(out)


#Cross-validation
    ----------------------------------------------------------
#noninf0
pred.noninfo<- function(x, y, nit,penal){
k<-length(unique(y[!is.na(y)]))
p<-dim(x)[2]
pi.0 <- table(y[!is.na(y)])/length(y[!is.na(y)])
alpha.0 <- log(cumsum(pi.0)/(1 - cumsum(pi.0)))[1:(k - 1)]
inits1<-list(tau=alpha.0,beta=c(rep(0,p)))
inits2<-list(tau=c(-1,1),beta=c(rnorm(p,0,0.001)))
inits3<-list(tau=c(-2,2),beta=c(rnorm(p,0,0.01)))
inits<-list(inits1,inits2,inits3)
params<-c("tau","beta","Y.pred","Y")
pred.data<-list(Y=y,X=x,N=length(y),nb=p,k=k)
pred.jags <- jags(data=pred.data, inits=inits, params,
n.iter=nit, model.file=penal,n.chains=3)
pred.upd <- autojags(pred.jags)$BUGSoutput$summary
```

```
156 return(pred.upd)
157 }
158
159 y.cv<-numeric()
160 for (i in 1:56){
161 x<-x.select
162 y<-hccCancerPanel$Tissue
163 y[i]<-NA
164 pred.upd<-pred.noninfo(x=x.select,y=y,nit=1000,penal="DE_pred.txt") #
      change to Reg_red to let beta ~ N
165 y.cv[i]<-pred.upd[i,"50%"]
166 }
167 out<-table(hccCancerPanel$Tissue,y.cv)
168 1-sum(diag(out))/sum(out)
169 rm(list=ls()[ls()=="out"])
170 rm(list=ls()[ls()=="y.cv"])
171 #Cross-validation
      ----------------------------------------------------
172 #inf0
173 pred.info<- function(x, y, nit,penal){
174 k<-length(unique(y[!is.na(y)]))
175 p<-dim(x)[2]
176 pi.0 <- table(y[!is.na(y)])/length(y[!is.na(y)])
```

```
alpha.0 <- log(cumsum(pi.0)/(1 - cumsum(pi.0)))[1:(k - 1)]
inits1<-list(tau1=alpha.0[1],beta=c(rep(0,p)),Diff=alpha.0[2]-alpha
    .0[1])
inits2<-list(tau1=-1,beta=c(rnorm(p,0,0.01)),Diff=2)
inits3<-list(tau1=-2,beta=c(rnorm(p,0,0.001)),Diff=4)
inits<-list(inits1,inits2,inits3)
info.data<-list(Y=y,X=x,N=length(y),nb=p,k=k,a=alpha.0[1])
params<-c("tau1","Diff","beta","Y.pred","Y")
info.jags <- jags(data=info.data, inits=inits, params,
n.iter=nit, model.file=penal,n.chains=3)
info.upd <- autojags(info.jags)$BUGSoutput$summary
return(info.upd)
}

y.cv<-numeric()
for (i in 1:56){
x<-x.select
y<-hccCancerPanel$Tissue
y[i]<-NA
pred.upd<-pred.info(x=x.select,y=y,nit=1000,penal="DE_pred_info.txt")
    #change to Reg_pred_info to let beta ~ N
y.cv[i]<-pred.upd[i+1,"50%"]
}
```

```
198 out<-table(hccCancerPanel$Tissue,y.cv)
199 1-sum(diag(out))/sum(out)
```

B.3 Bayesian model files

Penalized Bayesian cumulative logit model with non-informative priors, corresponding to equation (3.2) (Non_info_model.txt)

```
1  model{
2    for(i in 1:N){
3      mu[i] <- inprod(X[i,],bgamma[])
4      logit(Q[i,1]) <- tau[1]+mu[i]
5      p[i,1] <- Q[i,1]
6      for(j in 2:(k-1)){
7      logit(Q[i,j]) <- tau[j]+mu[i]
8      p[i,j] <- Q[i,j] - Q[i,j-1]
9      }
10     p[i,k] <- 1 - Q[i,(k-1)]
11     Y[i] ~ dcat(p[i,1:k])
12          }
13                tt <- lambda * v
14                for (b in 1:nb){
15     beta[b] ~ ddexp(0,tt)}
16       lambda ~ dgamma(1,1)
```

```
17                v ~ dgamma (0.0001,0.0001)
18                for (j in 1:nb){
19                 bgamma[j] <- beta[j] * gamma[j]
20                 gamma[j] ~ dbern(0.5)
21                                 }
22       tau[1] ~ dnorm(0,0.001)T(-6.9,6.9)
23         tau[2] ~ dnorm(0,0.001)T(tau[1],6.9)
24    }
```

Non_info_model2.txt

```
1 model{
2
3  for(i in 1:N){
4    mu[i] <- inprod(X[i,],bgamma[])
5    logit(Q[i,1]) <- tau[1]+mu[i]
6    p[i,1] <- Q[i,1]
7    for(j in 2:(k-1)){
8    logit(Q[i,j]) <- tau[j]+mu[i]
9    p[i,j] <- Q[i,j] - Q[i,j-1]
10   }
11
12   p[i,k] <- 1 - Q[i,(k-1)]
13
```

```
14    Y[i] ~ dcat(p[i,1:k])
15          }
16
17
18                tt <- lambda * v
19                for (b in 1:nb){
20    beta[b] ~ ddexp(0,tt)}
21
22    lambda ~ dgamma(0.025,0.05)
23                v ~ dgamma (0.0001,0.0001)
24
25                for (j in 1:nb){
26
27                 bgamma[j] <- beta[j] * gamma[j]
28                 gamma[j] ~ dbern(0.5)
29                                }
30
31               tau[1] ~ dnorm(0,0.001)T(-6.9,6.9)
32    tau[2] ~ dnorm(0,0.001)T(tau[1],6.9)
33
34
35  }
```

Penalized Bayesian cumulative logit model with informative priors, corresponding to equation (3.3) (info_model.txt)

```
model{

 for(i in 1:N){
 mu[i] <- inprod(X[i,],bgamma[])
   logit(Q[i,1]) <- tau1+mu[i]
   p[i,1] <- Q[i,1]
   for(j in 2:(k-1)){
   logit(Q[i,j]) <- tau2+mu[i]
   p[i,j] <- Q[i,j] - Q[i,j-1]
   }

   p[i,k] <- 1 - Q[i,(k-1)]

   Y[i] ~ dcat(p[i,1:k])
        }

            tt <- lambda * v
              for (b in 1:nb){
   beta[b] ~ ddexp(0,tt)}

   lambda ~ dgamma(1,1)
```

```
22              v ~ dgamma (0.0001,0.0001)
23
24              for (j in 1:nb){
25
26               bgamma[j] <- beta[j] * gamma[j]
27               gamma[j] ~ dbern(0.5)
28
29              }
30
31             tau1 ~ dnorm(a,0.001)T(-6.9,6.9)
32        Diff ~ dgamma(1.4,1)
33             tau2 <- tau1 + Diff
34
35
36  }
```

Penalized Bayesian cumulative logit model with non-informative priors(DE_pred.txt)

```
1 model{
2
3  for(i in 1:N){
4
5    mu[i] <- inprod(X[i,],beta[])
6
```

```
7     logit(Q[i,1]) <- tau[1]+mu[i]
8     p[i,1] <- Q[i,1]
9     for(j in 2:(k-1)){
10    logit(Q[i,j]) <- tau[j]+mu[i]
11    p[i,j] <- Q[i,j] - Q[i,j-1]
12    }
13
14    p[i,k] <- 1 - Q[i,(k-1)]
15
16    Y[i] ~ dcat(p[i,1:k])
17    Y.pred[i] ~ dcat(p[i,1:k])
18     }
19
20              tt <- lambda * v
21              for (b in 1:nb){
22    beta[b] ~ ddexp(0,tt)}
23
24    lambda ~ dgamma(1,1)
25              v ~ dgamma (0.0001,0.0001)
26
27             tau[1] ~ dnorm(0,0.001)T(-6.9,6.9)
28    tau[2] ~ dnorm(0,0.001)T(tau[1],6.9)
29
```

```
30
31  }
```

Penalized Bayesian cumulative logit model with informative priors (DE_pred_info.txt)

```
1 model{
2
3  for(i in 1:N){
4
5    mu[i] <- inprod(X[i,],beta[])
6
7
8    logit(Q[i,1]) <- tau1+mu[i]
9    p[i,1] <- Q[i,1]
10   for(j in 2:(k-1)){
11   logit(Q[i,j]) <- tau2+mu[i]
12   p[i,j] <- Q[i,j] - Q[i,j-1]
13   }
14
15   p[i,k] <- 1 - Q[i,(k-1)]
16
17   Y[i] ~ dcat(p[i,1:k])
18   Y.pred[i] ~ dcat(p[i,1:k])
19    }
```

```
20
21
22              tt <- lambda * v
23              for (b in 1:nb){
24   beta[b] ~ ddexp(0,tt)}
25
26   lambda ~ dgamma(1,1)
27              v ~ dgamma (0.0001,0.0001)
28
29
30         tau1 ~ dnorm(a,0.001)T(-6.9,6.9)
31        Diff ~ dgamma(1.2,1)
32             tau2 <- tau1 + Diff
33
34
35  }
```

Non-penalized Bayesian cumulative logit model with non-informative priors(Reg_pred.txt)

```
1 model{
2  for(i in 1:N){
3  mu[i] <- inprod(X[i,],beta[])
4       logit(Q[i,1]) <- tau[1]+mu[i]
```

```
    p[i,1] <- Q[i,1]
    for(j in 2:(k-1)){
    logit(Q[i,j]) <- tau[j]+mu[i]
    p[i,j] <- Q[i,j] - Q[i,j-1]
    }

    p[i,k] <- 1 - Q[i,(k-1)]

    Y[i] ~ dcat(p[i,1:k])
    Y.pred[i] ~ dcat(p[i,1:k])
        }

    for (b in 1:nb){
    beta[b] ~ dnorm(0,0.001)}

     tau[1] ~ dnorm(0,0.001)T(-6.9,6.9)
    tau[2] ~ dnorm(0,0.001)T(tau[1],6.9)

  }
```

Non-penalized Bayesian cumulative logit model with informative priors(Reg_pred_info.txt)

```
model{
  for(i in 1:N){
```

```
3     mu[i] <- inprod(X[i,],beta[])
4         logit(Q[i,1]) <- tau1+mu[i]
5     p[i,1] <- Q[i,1]
6     for(j in 2:(k-1)){
7     logit(Q[i,j]) <- tau2+mu[i]
8     p[i,j] <- Q[i,j] - Q[i,j-1]
9     }
10
11    p[i,k] <- 1 - Q[i,(k-1)]
12
13    Y[i] ~ dcat(p[i,1:k])
14    Y.pred[i] ~ dcat(p[i,1:k])
15        }
16
17
18    for (b in 1:nb){
19    beta[b] ~ dnorm(0,0.001)}
20
21    tau1 ~ dnorm(a,0.001)T(-6.9,6.9)
22         Diff ~ dgamma(1.2,1)
23               tau2 <- tau1 + Diff
24
25  }
```

Appendix C

Chapter 3 appendix

C.1 Filtering code

```
1 ##############----------------
2 #Four univariate feature selection methods
3 ##############----------------
4 library("survival")
5 library(bootstrap)
6 library(VGAM)
7 AML<-read.csv("AMLdata.csv")
8 attach(AML)
9 censor<-ifelse(vital.status=="A",1,0)
10 AML.feature<-AML[,4:252]
11 delta<-1-censor
```

```
12
13
14 #Cox PH-------------------------------------------------
15 AML.feature<-AML[,4:252]
16 p1a.pvalue<-numeric()
17 for (i in 1:dim(AML.feature)[2]){
18 fit<-coxph(Surv(Overall_Survival,delta)~AML.feature[,i])
19 #fit<-coxph(Surv(Overall_Survival)~AML.feature[,i])
20 p1a.pvalue[i]<-summary(fit)$waldtest["pvalue"]
21 }
22 sig.a<-colnames(AML.feature)[which(p1a.pvalue<=0.1)]
23
24
25
26
27 #Spearman rank test--------------------------------------
28 p1b.pvalue<-numeric()
29 for (i in 1:dim(AML.feature)[2]){
30 p1b.pvalue[i]<-cor.test(Overall_Survival,AML.feature[,i],method="
       spearman")$p.value
31 }
32 sig.b<-colnames(AML.feature)[which(p1b.pvalue<=0.1)]
33
```

```
34
35 #IC----------------------------------------------------
36 AML.feature$AHD<-as.numeric(AML.feature$AHD)
37 var.class<-lapply(AML.feature,class)
38 cat.feature<-AML.feature[,var.class=="integer"]
39 numeric.feature<-AML.feature[,var.class=="numeric"]
40 kmean.feature<-data.frame(cat.feature,apply(numeric.feature,2,
      function(x) kmeans(x,3)$cluster))
41
42 #function that caculates important score
43 Ic.fun<-function(var,surv){
44 Yk_hat<-aggregate(surv, by=list(var),FUN=mean)$x
45 n_k<-aggregate(surv, by=list(var),FUN=length)$x
46 Ic<-sum((n_k^2)*((Yk_hat-mean(surv))^2))/TSS
47 return(Ic)
48 }
49 TSS<-sum((Overall_Survival-mean(Overall_Survival))^2)
50
51 Ic.obs<-numeric()
52 for (i in 1:dim(kmean.feature)[2]){
53 Ic.obs[i]<-Ic.fun(kmean.feature[,i],surv=Overall_Survival)
54 }
55
```

```
#bootstrap
subject<-1:dim(kmean.feature)[1]
Ic.matrix<-matrix(nr=dim(kmean.feature)[2],nc=1000)
for (i in 1:dim(kmean.feature)[2]){
  var<-data.frame(Overall_Survival,kmean.feature[,i])
  var2<-var[order(var[,2]),]
  set.seed(1234)
for (b in 1:1000) {
      bootsample<-sample(subject, replace=TRUE)
      Surb<-var2[,1][bootsample]
      #slope[b]<-lm(Surb ~ var2[,2])$coef[2] #mean=0 :)
      #test.p[b]<-anova(lm(Surb ~ var2[,2]))[1,5] #uniform dist :)
      Ic.matrix[i,b]<-Ic.fun(var2[,2],surv=Surb)
  }
}


p1c.pvalue<-numeric()
for (i in 1:dim(kmean.feature)[2]){
            Ic.ind<-ifelse(Ic.matrix[i,] > Ic.obs[i], 1, 0)
p1c.pvalue[i]<-sum(Ic.ind)/length(Ic.ind)
}
sig.c<-colnames(kmean.feature)[which(p1c.pvalue<=0.1)]
```

```
79
80 hist(Ic.matrix[,2],breaks=100,main="Prior AML")
81 abline(v=Ic.obs[2],col="red")
82 kmean.feature[,1:5]
83
84 #Id-------------------------------------------------------------
85
86 #function that calculate Id
87 Id_fun<-function(surv,var){
88 y<-ifelse(surv<=52,1,ifelse(surv<=104,2,3))
89 k<-length(unique(var))
90 Fmatrix<-matrix(nr=k,nc=3)
91 sumG<-numeric()
92 for (i in 1:k){
93 Fmatrix[i,]<-cumsum(table(factor(y[var==i],levels = c(1:3))))/sum(
      table(y[var==i]))
94 sumG[i]<-sum(Fmatrix[i,]*(1-Fmatrix[i,]))
95 }
96 sum(sumG)
97 }
98
99 #change binary variable from 0,1 to 1,2 to enable the function
100 kmean.feature2<-kmean.feature
```

```
kmean.feature2[,1:6][kmean.feature2[,1:6]==1]<-2
kmean.feature2[,1:6][kmean.feature2[,1:6]==0]<-1

#observed Id for each x
Id.obs<-numeric()
for (i in 1:dim(kmean.feature2)[2]){
Id.obs[i]<-Id_fun(var=kmean.feature2[,i],surv=Overall_Survival)
}

#bootstrap
subject<-1:dim(kmean.feature2)[1]
Id.matrix<-matrix(nr=dim(kmean.feature2)[2],nc=1000)
for (i in 1:dim(kmean.feature2)[2]){
  var<-data.frame(Overall_Survival,kmean.feature2[,i])
  var2<-var[order(var[,2]),]
  set.seed(1234)
for (b in 1:1000) {
      bootsample<-sample(subject, replace=TRUE)
      Surb<-var2[,1][bootsample]
      Id.matrix[i,b]<-Id_fun(surv=Surb,var=var2[,2])
  }
}

```

```
p1d.pvalue<-numeric()
for (i in 1:dim(kmean.feature)[2]){
    Id.ind<-ifelse(Id.matrix[i,] < Id.obs[i], 1 , 0)
    p1d.pvalue[i]<-sum(Id.ind)/length(Id.ind)
}
sig.d<-colnames(kmean.feature2)[which(p1d.pvalue<=0.1)]

table(sig.c%in%sig.d)


##############-------------------------------------------
Multivariable predictive model using only significant features after
    filtering
##############-------------------------------------------
y<-ifelse(Overall_Survival<=52,1,ifelse(Overall_Survival<=104,2,3))
method1.data<-data.frame(AML.feature[,which(p1a.pvalue<=0.1)])
method2.data<-data.frame(AML.feature[,which(p1b.pvalue<=0.1)])
method3.data<-data.frame(AML.feature[,colnames(AML.feature)%in%sig.c
    ==TRUE])
method4.data<-data.frame(AML.feature[,colnames(AML.feature)%in%sig.d
    ==TRUE])

#FCR-----------------------------------------------------
```

```
#forward continuation ratio model is used for prediction
fit.vglm1<-vglm(y~.,data=method1.data, family=sratio(parallel=T,
    reverse=F),link=cloglog)
vglm.pi<-predict(fit.vglm1,type="response")
apply(vglm.pi,2,range)
vglm.class<-apply(vglm.pi,1,which.max)

#resubsitution error
out1<-table(factor(y,level=c(1:3)), factor(vglm.class,level=c(1:3)))
(sum(out1)-sum(diag(out1)))/sum(out1)

#leave one cross validation
#the function below taking predictors x data.frame, and will return
    misclassification rate (CV_error)
LOOC.vglm<-function(x){
cv.fit<-function(x,y){vglm(y~.,data=data.frame(x), family=sratio(
    parallel=T,reverse=F),link=cloglog)}
cv.predict<-function(fit,x){apply(predict(fit,newdata=x,type="
    response"),1,which.max)}
u <- vector("list", length(y))
cv.result <- rep(NA,length(y) )
for (j in 1:length(y)) {
       u <- cv.fit(x[-j, ], y[-j])
```

```
163         cv.result[j] <- cv.predict(u, x[j,])
164     }
165 out.cv<-table(factor(cv.result,level=c(1:3)),factor(y,level=c(1:3)))
166 (sum(out.cv)-sum(diag(out.cv)))/sum(out.cv)
167 }
168 LOOC.vglm(method1.data)
169
170 #
      ------------------------------------------------------------------------------

171 fit.vglm2<-vglm(y~.,data=method2.data,family=sratio(parallel=T,
      reverse=F),link=cloglog)
172 vglm.pi<-predict(fit.vglm2,type="response")
173 vglm.class<-apply(vglm.pi,1,which.max)
174
175 #resubsitution error
176 out2<-table(factor(y,level=c(1:3)), factor(vglm.class,level=c(1:3)))
177 (sum(out2)-sum(diag(out2)))/sum(out2)
178
179 #leave one cross validation
180 LOOC.vglm(method2.data)
181
```

```
#
    ------------------------------------------------------------------------

fit.vglm3<-vglm(y~.,data=method3.data,family=sratio(parallel=T,
    reverse=F),link=cloglog)
vglm.pi<-predict(fit.vglm3,type="response")
vglm.class<-apply(vglm.pi,1,which.max)

#resubsitution error
out3<-table(factor(y,level=c(1:3)), factor(vglm.class,level=c(1:3)))
(sum(out3)-sum(diag(out3)))/sum(out3)
#CV
LOOC.vglm(method3.data)

#
    ------------------------------------------------------------------------

fit.vglm4<-vglm(y~.,data=method4.data,family=sratio(parallel=T,
    reverse=F),link=cloglog)
vglm.pi<-predict(fit.vglm4,type="response")
vglm.class<-apply(vglm.pi,1,which.max)

#resubsitution error
```

```
out4<-table(factor(y,level=c(1:3)), factor(vglm.class,level=c(1:3)))
(sum(out4)-sum(diag(out4)))/sum(out4)
#CV
LOOC.vglm(method4.data)


#Cox PH multivariable----------------------------------------
#Misclassification rate
#
      ------------------------------------------------------------------------------


#Fit cox model
fit.cox1<-coxph(Surv(Overall_Survival,delta)~.,data=method1.data,ties
      ="breslow")

#survival.rate is a matrix contains the probability of people die
      within three intervals (<55, 55-144,>144)
# Col1=dealth rate for <=55, Col2=55-144, ...
# Row1=patient 1,...

survival.rate<-matrix(nc=3,nr=dim(method1.data)[1])
for (i in 1:dim(method1.data)[1]){
```

```
summary.cox1<-summary(survfit(fit.cox1,newdata=method1.data[i,])) #
    survfit function will create predicted survival curve for ith
    observation
survival.rate[i,1]<-ifelse(length(summary.cox1$surv[summary.cox1$time
    <=52])>0,diff(range(summary.cox1$surv[summary.cox1$time<=52])),0)
survival.rate[i,2]<-ifelse(length(summary.cox1$surv[summary.cox1$time
    <=104 & summary.cox1$time >52])>0,diff(range(summary.cox1$surv[
    summary.cox1$time<=104 & summary.cox1$time >52])),0)
survival.rate[i,3]<-1-survival.rate[i,1]-survival.rate[i,2]
}
cox.y1<-apply(survival.rate,1,which.max)
cox1.table<-table(factor(cox.y1,level=c(1,2,3)),factor(y,level=c
    (1,2,3))) #nice
cox1.table
(sum(cox1.table)-sum(diag(cox1.table)))/sum(cox1.table)


fit.cox2<-coxph(Surv(Overall_Survival,delta)~.,data=method2.data,ties
    ="breslow")
survival.rate2<-matrix(nc=3,nr=dim(method1.data)[1])
for (i in 1:dim(method1.data)[1]){
summary.cox2<-summary(survfit(fit.cox2,newdata=method2.data[i,]))
```

```
survival.rate2[i,1]<-ifelse(length(summary.cox2$surv[summary.cox2$
    time<=52])>0,diff(range(summary.cox2$surv[summary.cox2$time<=52]))
    ,0)
survival.rate2[i,2]<-ifelse(length(summary.cox2$surv[summary.cox2$
    time<=104 & summary.cox2$time >52])>0,diff(range(summary.cox2$surv
    [summary.cox2$time<=104 & summary.cox2$time >52])),0)
survival.rate2[i,3]<-1-survival.rate2[i,1]-survival.rate2[i,2]
}
cox.y2<-apply(survival.rate2,1,which.max)
cox2.table<-table(factor(cox.y2,level=c(1,2,3)),factor(y,level=c
    (1,2,3)))
cox2.table
(sum(cox2.table)-sum(diag(cox2.table)))/sum(cox2.table)


fit.cox3<-coxph(Surv(Overall_Survival,delta)~.,data=method3.data,ties
    ="breslow")
survival.rate3<-matrix(nc=3,nr=dim(method1.data)[1])
for (i in 1:dim(method1.data)[1]){
summary.cox3<-summary(survfit(fit.cox3,newdata=method3.data[i,]))
survival.rate3[i,1]<-ifelse(length(summary.cox3$surv[summary.cox3$
    time<=52])>0,diff(range(summary.cox3$surv[summary.cox3$time<=52]))
    ,0)
```

```
survival.rate3[i,2]<-ifelse(length(summary.cox3$surv[summary.cox3$
    time<=104 & summary.cox3$time >52])>0,diff(range(summary.cox3$surv
    [summary.cox3$time<=104 & summary.cox3$time >52])),0)
survival.rate3[i,3]<-1-survival.rate3[i,1]-survival.rate3[i,2]
}
cox.y3<-apply(survival.rate3,1,which.max)
cox3.table<-table(factor(cox.y3,level=c(1,2,3)),factor(y,level=c
    (1,2,3)))
cox3.table
(sum(cox3.table)-sum(diag(cox3.table)))/sum(cox3.table)


fit.cox4<-coxph(Surv(Overall_Survival,delta)~.,data=method4.data,ties
    ="breslow")
survival.rate4<-matrix(nc=3,nr=dim(method1.data)[1])
for (i in 1:dim(method1.data)[1]){
summary.cox4<-summary(survfit(fit.cox4,newdata=method4.data[i,]))
survival.rate4[i,1]<-ifelse(length(summary.cox4$surv[summary.cox4$
    time<=52])>0,diff(range(summary.cox4$surv[summary.cox4$time<=52]))
    ,0)
survival.rate4[i,2]<-ifelse(length(summary.cox4$surv[summary.cox4$
    time<=104 & summary.cox4$time >52])>0,diff(range(summary.cox4$surv
    [summary.cox4$time<=104 & summary.cox4$time >52])),0)
```

```
survival.rate4[i,3]<-1-survival.rate4[i,1]-survival.rate4[i,2]
}
cox.y4<-apply(survival.rate4,1,which.max)
cox4.table<-table(factor(cox.y4,level=c(1,2,3)),factor(y,level=c
    (1,2,3)))
cox4.table
(sum(cox4.table)-sum(diag(cox4.table)))/sum(cox4.table)

#cross validation
#
    ------------------------------------------------------------------------------

#leave one cross validation
#the function below taking data need to be predicted, and will return
     misclassification rate (CV_error)

 LOOC.cox<-function(data){

 #cox.pr function take x=coxph object, and y=new data and produce
     predicted
 cox.pr<-function(x,y){
 summ.cox<-summary(survfit(x,newdata=y))
 surv.vec<-numeric()
```

```
280 surv.vec[1]<-ifelse(length(summ.cox$surv[summ.cox$time<=52])>0,diff(
       range(summ.cox$surv[summ.cox$time<=52])),0)
281 surv.vec[2]<-ifelse(length(summ.cox$surv[summ.cox$time<=104 & summ.
       cox$time >52])>0,diff(range(summ.cox$surv[summ.cox$time<=104 &
       summ.cox$time >52])),0)
282 surv.vec[3]<-1-surv.vec[1]-surv.vec[2]
283 cox.y<-which.max(surv.vec)
284 cox.y
285 }
286
287 u<-numeric()
288 for (i in 1:dim(data)[1]){
289 fit.cox1<-coxph(Surv(Overall_Survival[-i],delta[-i])~.,data=data[-i
       ,])
290 u[i]<-cox.pr(fit.cox1,data[i,])
291 }
292
293 out.cv<-table(factor(u,level=c(1:3)),factor(y,level=c(1:3)))
294 (sum(out.cv)-sum(diag(out.cv)))/sum(out.cv)
295 }
296
297 LOOC.cox(method1.data)
298 LOOC.cox(method2.data)
```

```
299 LOOC.cox(method3.data)
300 LOOC.cox(method4.data)
```

C.2 Coxpath and GMIFS-FCR code

```
1 ##########-----------------------------------------------
2 ##Coxpath
3 ##########-----------------------------------------
4 AML<-read.csv("AMLdata.csv")
5 attach(AML)
6 censor<-ifelse(vital.status=="A",1,0)
7 delta<-1-censor
8 AML.feature<-AML[,4:252]
9
10 ### RE-SUBSTITUTION ERROR
     ---------------------------------------------------------
11 #fit coxpath LASSO regression
12 library(glmpath)
13 fit.path1<-coxpath(list(x=as.matrix(AML.feature),time=Overall_
     Survival,status=delta))
14 summary.coxpath1<-summary(fit.path1)
15 step1<-as.numeric(gsub("Step ","", rownames(summary.coxpath1)[which.
     min(summary.coxpath1$AIC)] ) )
16 step1
```

```
#obtain coefficient
#fit.path1$b.corrector will return the a matrix of coefficient
    obtained in corrector step
#The coef obtained from $b.corrector is same as obtained from predict
    .path
path.coef<-fit.path1$b.corrector[step1,]

#obtain cumulative hazards
#since predict.coxpath(type="coxph") did not converge, I hardcode
    hazard function and predicted survival curve
#aalen will return Aalen's estimates of the cumulative hazard

aalen <- function(x,time,delta,beta){
event <- delta == 1
new.x<-scale(x,center=TRUE,scale=FALSE)
path.coef<-beta
risk<-exp(new.x%*%path.coef)
dt <- unique(time[event])
ct <- unique(time[event==FALSE])
ct.dt<-data.frame()
for(i in 1:length(ct)){
ct.dt[i,1]<-ct[i]
diff<-ct[i]-dt
```

```
ct.dt[i,2]<-(dt[diff>=0])[which.min(diff[diff>=0])]
}
colnames(ct.dt)<-c("cen.time","corres.time")
k <- length(dt)
lambda <- rep(0,k)
for(i in 1:k) {
lambda[i] <- sum(event[time==dt[i]])/sum(risk[time >= dt[i]])
}
result<-data.frame(time=dt, lambda=lambda)
result2<-result[order(result$time),]
new.result<-transform(result2,lambda.cum=cumsum(lambda))

#data.frame contained censored time and its lambda.cum
censor.haz<-merge(new.result,ct.dt,by.x="time",by.y="corres.time",all
    .y=TRUE)
new.censor.haz<-transform(censor.haz,time=cen.time)
cum.haz<-rbind(new.result,new.censor.haz[,-4])
z0<-colMeans(x)
bz0<-sum(z0*path.coef)
cum.haz.data<-transform(cum.haz,lambda.cum.base=lambda.cum*exp(-bz0))
cum.haz.data[order(cum.haz$time),]
}

```

```
x<-as.matrix(AML.feature)
cum.haz<-aalen(x,time=Overall_Survival,delta=delta,beta=path.coef)
exp.xb<-exp(as.matrix(AML.feature)%*%path.coef)
survival.rate.path<-matrix(nc=3,nr=dim(AML.feature)[1])
for(i in 1:length(exp.xb)){
pred.surv<-data.frame(surv=exp(-cum.haz$lambda.cum.base*exp.xb[i]),
    time=cum.haz$time)
survival.rate.path[i,1]<-ifelse(length(pred.surv$surv[pred.surv$time
    <=52])>0,diff(range(pred.surv$surv[pred.surv$time<=52])),0)
survival.rate.path[i,2]<-ifelse(length(pred.surv$surv[pred.surv$time
    <=104 & pred.surv$time >52])>0,diff(range(pred.surv$surv[pred.surv
    $time<=104 & pred.surv$time >52])),0)
survival.rate.path[i,3]<-1-survival.rate.path[i,1]-survival.rate.path
    [i,2]
 }

cox.path.y<-apply(survival.rate.path,1,which.max)
y<-ifelse(Overall_Survival<=52,1,ifelse(Overall_Survival<=104,2,3))
cox.path.table<-table(factor(cox.path.y,level=c(1,2,3)),factor(y,
    level=c(1,2,3)))
(sum(cox.path.table)-sum(diag(cox.path.table)))/sum(cox.path.table)

```

```
75 ### CV ERROR
     ------------------------------------------------------------
76 ## Parallel coding was used to speed up the CV process
77 ## Parallel version using doSNOW, foreach, and itertools packages
78 library(doSNOW)
79 library(itertools)
80 machines <- rep("localhost", each=4)
81 cl <- makeCluster(machines, type="SOCK", outfile="test.txt")
82 registerDoSNOW(cl)
83
84 system.time({
85   iter <- isplitIndices(nrow(AML.feature), chunks=nrow(AML.feature))
86   nfold.class <- foreach(i=iter,
87                 .combine=c, .packages="glmpath") %dopar% {
88                             fit <- coxpath(list(x=as.matrix(AML.
                                feature[-i,]),time=AML$Overall_Survival
                                [-i],status=delta[-i]))
89          summary.fit<-summary(fit)
90          model.select<-as.numeric(gsub("Step ","",rownames(summary.
               fit)[which.min(summary.fit$AIC)]))
91
92
93          path.coef.cv<-fit$b.corrector[model.select,]
```

```
94          cum.haz<-aalen(x=as.matrix(AML.feature[-i,]),time=AML$
               Overall_Survival[-i],delta=delta[-i],path.coef.cv)
95                          exp.xb<-exp(sum(AML.feature[i,]*path.coef.
                               cv))
96                          pred.surv<-data.frame(surv=exp(-cum.haz$
                               lambda.cum*exp.xb),time=cum.haz$time)
97
98          survival.rate.path<-numeric()
99          survival.rate.path[1]<-ifelse(length(pred.surv$surv[pred.
               surv$time<=52])>0,diff(range(pred.surv$surv[pred.surv$
               time<=52])),0)
100         survival.rate.path[2]<-ifelse(length(pred.surv$surv[pred.
               surv$time<=104 & pred.surv$time >52])>0,diff(range(pred.
               surv$surv[pred.surv$time<=104 & pred.surv$time >52])),0)
101         survival.rate.path[3]<-1-survival.rate.path[1]-survival.rate
               .path[2]
102         cox.path.y<-which.max(survival.rate.path)
103         return(cox.path.y)
104       }
105 })
106
107 stopCluster(cl)
108 y<-ifelse(Overall_Survival<=52,1,ifelse(Overall_Survival<=104,2,3))
```

```
cox.path.table<-table(factor(nfold.class,level=c(1,2,3)),factor(y,
    level=c(1,2,3)))
(sum(cox.path.table)-sum(diag(cox.path.table)))/sum(cox.path.table) #
    CV Error


###############---------------------------------------
##ordinalgmifs
###############---------------------------------------
#Kyle Ferber codes for censoring-----------------
G<-function(z){
 1-exp(-exp(z))
}
### Forward Continuation Ratio GMIFS function ###
forwardcr.stepwise<-function(x,y,censor=NULL,tol=1e-5, epsilon=0.001,
     scale=FALSE) {
  Detail content is written by Kyle Ferber, not disclosed in this
      thesis
    }

### Function to update alpha ###
ForwardCR.fn<-function(par, x, y, censor=NULL, beta) {
```

```
128   Detail content is written by Kyle Ferber, not disclosed in this
          thesis
129 }
130
131 ### Function to predict class ###
132 predict.forwardCR<-function(fit,newx,model.select=NA) {
133   Detail content is written by Kyle Ferber, not disclosed in this
          thesis
134 }
135
136
137 ### Function to predict cv class ###
138 predict.forwardCR.cv<-function(fit,newx,model.select=NA) {
139   x<-fit$x
140   y<-fit$y
141   if (is.na(model.select)) model.select=dim(fit$beta)[1]
142   beta<-fit$beta[model.select,]
143   alpha<-fit$alpha[model.select,]
144   k<-length(unique(y))
145   newx<-as.matrix(newx)
146   if (identical(newx,x)) {
147     if (fit$scale) {
148       newx<-scale(newx,center=TRUE,scale=TRUE)
```

```
149     }
150   } else if (fit$scale) {
151     newx<-rbind(x,newx)
152     newx<-scale(newx,center=TRUE,scale=TRUE)
153     newx<-matrix(newx[-(1:dim(x)[1]),],ncol=dim(x)[2])
154   }
155   levels<-sort(unique(y))
156     Xb<-newx%*%beta
157     pi <- matrix(0, nrow = dim(newx)[1], ncol = k)
158     pi[,1]<-G(alpha[1]+Xb)
159     pi[,2]<-G(alpha[2]+Xb)*(1-pi[,1])
160     if (k>3) {
161     for (i in 3:(k-1)) {
162       pi[,i]<-G(alpha[i]+Xb)*(1-matrix(apply(pi[,1:(i-1)],1,sum),nrow=
              nrow(pi),byrow=T))
163     }
164   }
165   pi[,k]<-1-matrix(sum(pi[,1:(k-1)]),nrow=nrow(pi),byrow=T)
166   class<-levels[apply(pi,1,which.max)]
167   list(predicted=pi,class=class)
168 }
169
```

```
#
    ------------------------------------------------------------------------------

y<-ifelse(Overall_Survival<=52,1,ifelse(Overall_Survival<=104,2,3))

fit.fcr.survival.over<-forwardcr.stepwise(x=AML.feature, censor=
    censor, y=y, scale=TRUE, epsilon=0.01)
#fit.class.over<-predict.forwardCR(fit.fcr.survival.over,newx=AML.
    feature)
fit.class.over<-predict.forwardCR(fit.fcr.survival.over,newx=AML.
    feature,model.select=245)
censor.table.over<-table(factor(fit.class.over$class,level=c(1:3)),
    factor(y,level=c(1:3)))
(sum(censor.table.over)-sum(diag(censor.table.over)))/sum(censor.
    table.over)
AIC.model<-fit.fcr.survival.over$beta[fit.fcr.survival.over$model.
    select,]
length(AIC.model[AIC.model!=0])
AIC.model[AIC.model!=0]

fit.class.cv<-numeric()
for (i in 1:dim(AML.feature)[1]){
```

```
fit<-forwardcr.stepwise(x=AML.feature[-i,], censor=censor[-i], y=y[-i
    ], scale=TRUE, epsilon=0.01)
fit.class.cv[i]<-predict.forwardCR.cv(fit,newx=AML.feature[i,],model.
    select=fit$model.select)$class
}

out.cv<-table(factor(fit.class.cv,level=c(1:3)),factor(y,level=c(1:3)
    ))
(sum(out.cv)-sum(diag(out.cv)))/sum(out.cv)
```

Vita

Qing Zhou was born October 26, 1986 in Huainan, Anhui province, People's Republic of China. She received her Bachelor's of microbiology from University of Wisconsin-Madison in 2009. In August of 2011, she moved to Virginia to pursue her PhD in Biostatistics at Virginia Commonwealth University in Richmond, VA. During her time at Virginia Commonwealth University, she became an American citizen through naturalization.