Generalized Hyperbolic Regression Modeling for Survival Data

Yuanjing Cai(20549267), Yidan Chen(20568620)

April 21, 2018

Abstract

Survival analysis is a branch of statistics concerning the estimation of the time-to-occurrence of an event of interest, which is used in a variety of fields such as cancer studies, sociology and engineering. And parametric survival regression is a way of linking the expected duration of time until an event happens to several explanatory variables of the event. A Heteroscedastic Linear Regression Model (HLM) is an appropriate candidate to capture the location and scaling of the distribution of the response, but to also capture the skewness and kurtosis, a Generalized Hyperbolic Regression Model (GHR) is a better choice for being more versatile and flexible. In this project, we analyzed the `colon` dataset in R's survival package, and implemented a Markov Chain Monte Carlo (MCMC) algorithm to estimate the parameters of the GHR model. We want to find the most relevant explanatory variables to predict the survival time of patients with Stage B/C colon cancer after adjuvant chemotherapy.

1 Introduction

Let $\boldsymbol{y}$ denote the log-survival time response variable for a given subject, and ($\boldsymbol{x}$, $\boldsymbol{w}$) a pair of covariate vectors with respective dimensions $\boldsymbol{p}$ and $\boldsymbol{q}$. A Heteroscedastic Linear Regression Model (Long and Ervin 2000) is of the form:

$$\boldsymbol{y} = \boldsymbol{x}'\boldsymbol{\beta} + exp(\boldsymbol{w}'\boldsymbol{\gamma})^{1/2} \cdot \epsilon, \epsilon \sim \mathcal{N}(0, 1) \qquad (1.1)$$

where $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are to be estimated from data. A HLM model can capture covariate-dependent location and scaling of the response and it also has good computational properties. Even so, it is not flexible in skewness and kurtosis. Instead, we consider the Generalized Hyperbolic Regression Model, which has extra modeling flexibility and is able to replicate many special cases such as Gaussian distribution, t-distribution and variance-gamma distribution (BROWNE and McNICHOLAS 2015).

Using the `colon` dataset in R's survival package, we first preprocess the raw data, throwing out irrelevant variables and observations with missing covariates (31 observations out of 929). Linear Regression, Cox Proportional Hazard Regression (Fox 2008) and Accelerated Failure Time (Wei 1992) models were used when selecting significant covariates. Then we use Bayesian inference with a flat prior and Markov Chain Monte Carlo (MCMC) to estimate the parameters of the GHR model. In MCMC, we use the concept of data augmentation, where to sample from $p(\theta|y)$, the observed quantity y is augmented by a latent quantity z, so that $p(\theta|y, z)$ is easier to sample from than the original $p(\theta|y)$ (Tanner and Wong 1987). This in our case also effectively accounts for the effect of right-censoring, in which case the event of interest is not observed (Lagakos, 1979). A Gibbs sampler is used to sample every parameter from its conditional posterior given everything else. To ensure the correctness and the effectiveness of the algorithm, before applying it to the `colon` dataset, we first apply it to a simulated dataset to examine the quality of parameter estimation, and the necessity of parameter regularization. After fitting the tested model to the `colon` dataset, we assess the quality of prediction and address possible problems and causes.

2 Methodology

2.1 Model Specification

The Generalized Hyperbolic distribution is parameterized as follows,

$$y \sim GH(\mu, \alpha, \sigma, \psi, \lambda) \iff f(y|\theta) = (\frac{Q_\theta(y)}{R_\theta})^{\frac{\lambda}{2}-\frac{1}{4}} \frac{K_{\lambda-1/2}(\sqrt{Q_\theta(y)R_\theta})}{\sigma\sqrt{2\pi}K_\lambda(\psi)} exp(\frac{y-\mu}{\sigma^2/\alpha})$$

where $Q_\theta(y) = \psi + (y-\mu)^2/\sigma^2$, $R_\theta = \psi + (\alpha/\sigma)^2$, and $K_\alpha(x)$ is the modified Bessel function of the second kind. Specifically, in this parameterization, μ is the location parameter; α is the skewness parameter; σ is the scale parameter; ψ is the shape parameter; λ is the power parameter.

It is essentially the probability density of a random variable Y generated from a normal variance-mean mixture,

$$Y = \mu + \alpha V + \sigma\sqrt{V} \cdot Z$$

where the mixing variable V has a Generalized Inverse Gaussian distribution,

$$V \sim GIG(\psi, \eta, \lambda) \iff f(v|\psi, \eta, \lambda) = \frac{(v/\eta)^{\lambda-1}}{2\eta K_\lambda(\psi)} exp\{(-\frac{\psi}{2}(\frac{v}{\eta} + \frac{\eta}{v}))\},$$
$$\text{and } Z \sim \mathcal{N}(0,1).$$

Hence the data generating process for a GH distribution can be written as

$$V \sim GIG(\psi, \eta, \lambda),\ Y|V \sim \mathcal{N}(\mu + \alpha V, \sigma^2 V) \qquad (2.1)$$
$$\Rightarrow Y \sim GH(\mu, \alpha, \sigma, \psi, \lambda)$$

which will be critical for parameter inference in section 2.2.

We use a Generalized Hyperbolic Regression (GHR) model to link the survival time of the colon cancer patients and the covariates, such that for subject i, the survival time y_i given the covariate vectors $\boldsymbol{x_i}, \boldsymbol{w_i}$ follows a GH distribution,

$$y_i|\boldsymbol{x_i}, \boldsymbol{w_i} \sim GH(\mu_i, \alpha, \sigma_i, \psi, \lambda), \mu_i = \boldsymbol{x_i'\beta}, \sigma_i^2 = \exp(\boldsymbol{w_i'\gamma})$$

That is, in addition to the heteroscedastic linear regression model (1.1), we assume that there is a latent variable $V \sim GIG(\psi, \eta, \lambda)$ such that

$$y = \boldsymbol{x'\beta} + \alpha V + [exp(\boldsymbol{w'\gamma})V]^{1/2} \cdot Z \qquad (2.2)$$

Also, it can be shown that

$$E[y|\boldsymbol{x}, \boldsymbol{w}] = \boldsymbol{x'\beta} + \alpha E[V]$$
$$var(y|\boldsymbol{x}, \boldsymbol{w}) = \alpha^2 var(V) + exp(\boldsymbol{w'\gamma})E(V)$$

which gives point and interval estimates for the response.

2.2 Bayesian Inference for GHR model

Let $\boldsymbol{D = (y, X, W)}$ be the observed data, and $\boldsymbol{\theta = (\beta, \alpha, \gamma, \psi, \lambda)}$ be the parameters of interests in model (2.2). Bayesian inference can be conducted by sampling from the posterior distribution $\boldsymbol{p(\theta|D)} = \prod_{i=1}^n p(y_i|\theta, X, W) \cdot \pi(\boldsymbol{\theta})$ using an MCMC algorithm. However, directly sampling from $\boldsymbol{p(\theta|D)}$ is difficult. Alternatively we use the idea of *data augmentation* by adding the latent variable V to our complete data. That is,

$$y_{comp} = (D, V), \qquad y_{obs} = D, \qquad y_{miss} = V \qquad (2.3)$$

Then based on (2.1), the complete data model is

$$V_i|\boldsymbol{x_i}, \boldsymbol{w_i} \overset{\text{iid}}{\sim} GIG(\psi, \eta, \lambda)$$
$$y_i|V_i, \boldsymbol{x_i}, \boldsymbol{w_i} \overset{\text{ind}}{\sim} \mathcal{N}(\boldsymbol{x_i'\beta} + \alpha V_i, exp(\boldsymbol{w_i'\gamma})V_i)$$

which breaks the posterior distribution down to a mixture of GIG distribution and Normal distribution that are easier to sample from. To avoid parameter identifiability issues, we set η=1. So now we will sample from the augmented posterior distribution

$$p(\theta, V|D) = \prod_{i=1}^n p(y_i, Vi|\theta, X, W) \cdot \pi(\boldsymbol{\theta}) = \prod_{i=1}^n p(y_i|Vi, \theta, X, W) \cdot p(Vi|\theta) \cdot \pi(\theta),$$

and the MCMC samples from this posterior is equivalent to those from $p(\theta|D)$ because, if $\boldsymbol{(\theta^{(1)}, V^{(1)}), \ldots, (\theta^{(m)}, V^{(m)})}$ are MCMC samples with stationary distribution $\boldsymbol{p(\theta, V|D)}$, then marginally $\boldsymbol{\theta^{(1)}, \ldots, \theta^{(m)}}$ have stationary distribution $\boldsymbol{p(\theta|D) = \int p(\theta, V|D)\, dV}$, i.e. sampling $\boldsymbol{V}$ from its conditional distribution in every MCMC iteration is equivalent to integrating it out in $p(\theta, V|D)$.

For now, we assume a flat prior $\pi(\theta) \propto 1$, then the log of augmented posterior is

$$logp(\theta, V|D) = \sum_{i=1}^{n}[\frac{(y_i - x_i'\beta - \alpha V_i)^2}{2exp(w_i'\gamma)V_i} + \frac{w_i'\gamma}{2} + (\frac{3}{2} - \lambda)log\ V_i + \frac{\psi}{2}(V_i + 1/V_i) + log\ 2K_\lambda(\psi)] \tag{2.4}$$

The idea of data augmentation can be applied in a similar fashion to account for the effect of right-censoring. The observed response variable Y_i in the `colon` dataset is either the number of days until death or censoring, i.e. $Y_i = min\{y_i, C_i\}$, where C_i is the censoring time for subject i. In addition to (2.3), we add another indicator variable δ to the framework so that

$$y_{comp} = (D, V, \delta), \qquad y_{obs} = (X, W, \delta, \{y_i : \delta_i = 1\}), \qquad y_{miss} = (V, \{y_i : \delta_i = 0\})$$

where $\delta_i = 1$ if $Y_i = y_i$, i.e. the actual survival time is observed, and 0 otherwise.

Since y and C are conditionally independent given the covariates $\boldsymbol{(x, w)}$, i.e. the censoring mechanism is ignorable, the posterior $p(\theta, V|D, \delta) = p(\theta, V|D)$. Similar to V, in each cycle of MCMC, we will also sample the missing survival time y_i for the censored cases from its conditional distribution, which by (2.1) is a normal distribution, but with a lower bound C_i, as the event is not observed by the time of censoring.

The Gibbs sampler for posterior (2.4) is constructed as follows:

- Conditional update of y_i for censored cases.
  As indicated above the conditional distribution of y_i for the censored cases is a truncated Normal with lower bound C_i, the censoring time. Therefore

  $$y_i\{\delta_i = 0\}|C_i, V_i, x_i', w_i' \sim \mathcal{N}(x_i'\beta + V_i\alpha, exp(w_i'\gamma)V_i) \times \mathbb{1}\{y_i > C_i\}$$

- Conditional update of $(\boldsymbol{\beta}, \alpha)$. By discarding all terms in (2.4) which do not involve (α, β), we obtain the conditional log posterior of (α, β)

  $$log\ p(\beta, \alpha|\mathcal{A}_{-\beta,\alpha}) = -\frac{1}{2}\sum_{i=1}^{n}\frac{(y_i - \boldsymbol{x_i'}\beta - \alpha V_i)^2}{exp(\boldsymbol{w_i'}\gamma)V_i}$$

  which is a quadratic function of (β, α), suggesting that the conditional posterior of (β, α) must be normal. In fact, it is essentially log likelihood function of the ML estimator of the coefficients of a linear regression model

  $$y_i|V_i, \boldsymbol{x_i}, \boldsymbol{w_i} \sim \mathcal{N}(\boldsymbol{x_i'\beta} + \alpha V_i, exp(\boldsymbol{w_i'\gamma})V_i)$$

  Hence, we can do an exact Gibbs update for (β, α)

  $$\begin{bmatrix} \beta \\ \alpha \end{bmatrix}|\mathcal{A}_{-\beta,\alpha} \sim \mathcal{N}(\begin{bmatrix} \hat\beta \\ \hat\alpha \end{bmatrix}, \hat\Sigma),$$

  where $\begin{bmatrix} \hat\beta \\ \hat\alpha \end{bmatrix}$ is the MLE of the coefficients and $\hat\Sigma$ is the covariance matrix of the ML estimator. They can be easily extracted from an `lm` object in R.

- Conditional update of γ. Discarding all terms which don't depend on γ in (2.4) we obtain

  $$log\ p(\gamma|\mathcal{A}_{-\gamma}) = -\frac{1}{2}\sum_{i=1}^{n}[\frac{R_i}{exp(w_i\gamma)} + w_i'\gamma],\ R_i = \frac{(y_i - x_i'\beta - \alpha V_i)^2}{V_i}$$

  which has exactly the same form as the log likelihood function for the coefficients of a Gamma regression model

  $$R_i|w_i \sim Gamma(\frac{1}{2}, 2exp(w_i'\gamma))$$

the mode and covariance matrix can be extracted from a `glm` object in R. Then we have a Metropolized Independence multivariate normal matching the mode and quadrature of $log\ p(\gamma|\mathcal{A}_{-\gamma})$ as our transition density.

- Conditional update of V. Discarding everything which don't involve V in (2.4) we obtain

$$log\ p(V|\mathcal{A}_{-V}) = \textstyle\sum_{i=1}^{n} A_i log\ V_i + B_i V_i + C_i/V_i,$$

$$A_i = \tfrac{3}{2} - \lambda,\ B_i = \tfrac{\psi}{2} + \tfrac{\sigma^2}{2exp(w_i'\gamma)},\ C_i = \tfrac{\psi}{2} + \tfrac{(y_i - x_i'\beta)^2}{2exp(w_i'\gamma)}$$

  which corresponds to the log-PDF of a GIG distribution $GIG(\hat{\psi}_i, \hat{\eta}_i, \hat{\lambda}_i)$ up to its normalizing constant, where

$$\hat{\lambda}_i = 1 - A_i, \qquad \hat{\eta}_i = \sqrt{\tfrac{C_i}{B_i}}, \qquad \hat{\psi}_i = \sqrt{4B_iC_i}$$

  Thus we conditionally update V from $V_i|\mathcal{A}_{-V} \sim GIG(\hat{\psi}_i, \hat{\eta}_i, \hat{\lambda}_i)$.

- Conditional updates of ψ and λ. Discarding everything which don't involve ψ and λ in (2.4) we obtain

$$log\ p(\psi, \lambda) = \lambda S - \tfrac{\psi}{2}T - nlog\ 2K_\lambda(\psi),$$

$$S = \textstyle\sum_{i=1}^{n} log\ V_i, \qquad T = \sum_{i=1}^{n}(V_i + 1/V_i)$$

  While this log-pdf is convex, calculating the mode via Newton-Raphson requires the gradient and Hessian of $K_\lambda(\psi)$, which numerically stable algorithms are not readily available. We instead use Metropolis-within-Gibbs to update each component of (ψ, λ).

3 Data Analysis

3.1 Simulated Dataset

We first applied the Gibbs sampler described in section 2.2 to simulated data to test the correctness and effectiveness of the algorithm. We generated 500 observations of (y, V) from model (2.1) with $\theta_0 = (\beta_0, \alpha_0, \gamma_0, \psi_0, \lambda_0) = (1, 1, 1, 10, 10)$.

In Figure 1, we superimpose the analytic conditional PDF derived from (2.4) onto the histogram of MCMC samples of each parameter from their respective conditional posterior. Their shapes match and this suggests that the conditional posteriors are correctly implemented. Additionally, the bottom right figure suggests that the MVN proposal for updating γ matches the mode and quadrature of the target density.

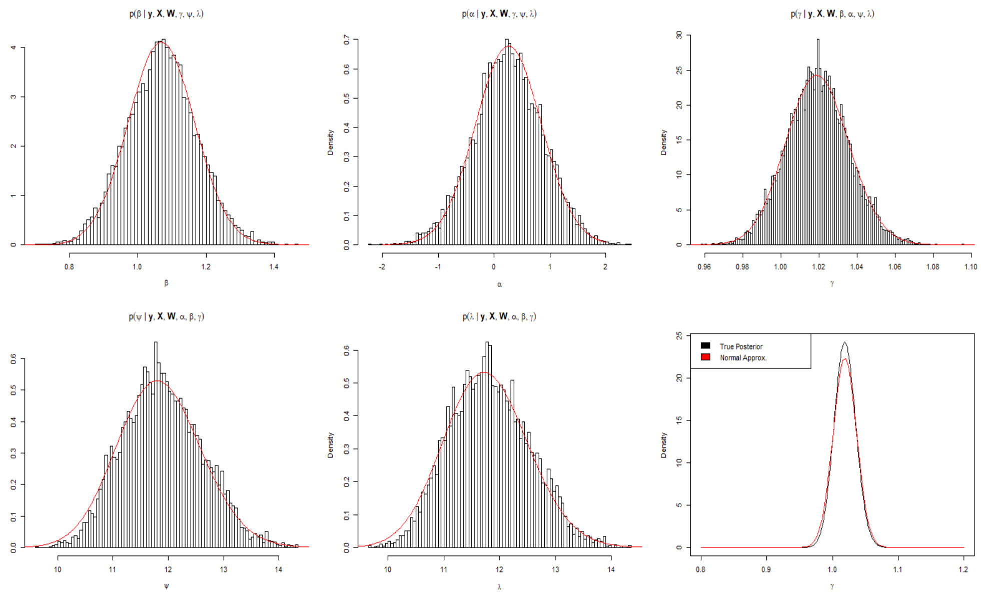

Figure 1: Analytic conditional posteriors vs. conditional MCMC samples

Next, we sample from the joint posterior $p(\theta, V|D)$, and we expect that the posterior mean of the parameters are close to their true values. However, MCMC algorithm is a stochastic process whose convergence to the posterior distribution is unpredictable. In our experiments with the simulated dataset, at times the MCMC samples of (ψ, λ) drifted away from the actual posterior, and the posterior mean gives poor estimate of the parameters. The below table is an example of the parameter estimates given by a "bad" Markov Chain (MCMC2) vs. a "good" one (MCMC1) (both 10000 iterations).

	MCMC1			MCMC2		
	2.5%	97.5%	Point	2.5%	97.5%	Point
β	0.86	1.48	1.17	0.87	1.50	1.19
α	-2.76	1.75	-0.32	-1.86	1.11	-0.32
γ	0.77	1.21	0.99	0.80	1.07	0.93
ψ	3.18	20.91	9.43	8.46	20.45	15.07
λ	1.47	14.48	9.18	7.42	43.46	24.35

Table 1: Point and Interval Estimation of θ given by two different MCMC runs

To solve this problem, we could have used a weakly informative prior on (ψ, λ) such as $\pi(\psi) \propto 1/\psi$ to penalize large values of (ψ, λ). However, experiments show that in the case of the true value of the parameters being large (e.g. $(\psi_0, \lambda_0) = (30, 20)$), imposing such priors will also prevent the Markov Chain from getting there, and we still end up getting poor estimate. In practice, we wouldn't have the information of the true parameter values, so using informative prior could introduce bias. Therefore, we believe that without definite prior knowledge of a parameter, using an uninformative flat prior is a safer choice.

To study the actual effect on prediction caused by the quality of parameter estimates, we plot the estimated $p(y_i|\theta_j, x_i, w_i)$ vs. the true PDF of $y_i|\theta_0, x_i, w_i$ for a randomly selected subject i, where θ_j is the parameter estimate given by $MCMC_j$. Despite the fact that two MCMC samples give seemingly very different point estimate for (ψ, λ), the PDF of $y_i|\theta_j, x_i, w_i$ do not seem to differ by much. Moreover, the 95% credible for (ψ, λ) given by MCMC2 covers the true value after all. This fact is supported by a number of trials with different true parameter values θ_0. Therefore, we claim that this MCMC algorithm still provides reliable estimates.

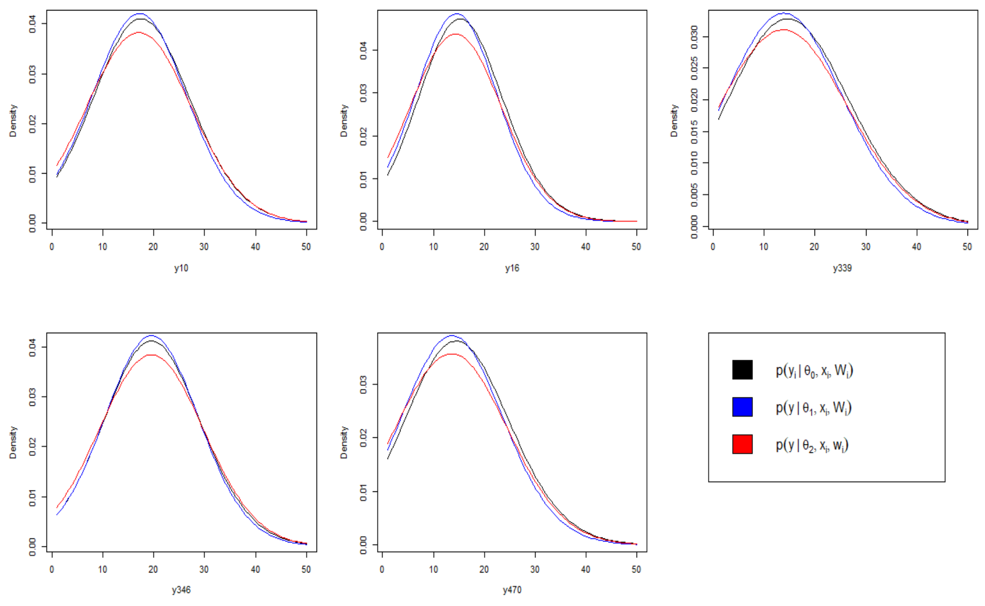

Figure 2: $p(y_i|\theta, x_i, w_i)$

3.2 Colon Dataset

3.2.1 Preprocessing and MCMC Initialization

We applied the tested algorithm to the `colon` dataset in R's `survival` package. The data are from one of the first successful trials of adjuvant chemotherapy for colon cancer. The raw `colon` dataset has 1858 observations and 16 variables. The response variable is `time`, the number of days until recurrence of the cancer, death or censoring.

We start with preprocessing the dataset. The `study` field is ignored since it is 1 for all observations. Also, there are two records for each individual, each for a different `etype` (1=recurrence, 2=death). Since we are only interested in the number of days before the death (`status` = 1) or censoring (`status` = 0) time of the patient, the recurrence of the cancer is not an event of concern. So we only keep the record with `etype` =2 for each individual in the dataset. After the preprocessing, the dataset has 929 subjects (458 censored cases + 430 non-censored cases) and 15 variables which are `id`, `rx`, `sex`, `age`, `perfor`, `adhere`, `nodes`, `obstruct`, `status`, `differ`, `extent`, `surg`, `node4`, `time`, `etype`.

We first use the HLM (1.1) setting to roughly see which covariates are influential on the survival time of patients with colon cancer. That is, we first fit a linear regression model `M<-lm(time∼.-id)` to the subjects who are not censored, and pick the covariates with small p-values (<0.05), which we choose to be the mean covariates x_i. Then we fit a `glm((M$residuals)^2 ∼ .-id, family=Gamma("log"))` to the squared residuals of `M` vs. the covariates, and pick those with small p-values, which we choose to be the log variance covariates w_i.

$\boldsymbol{X}$=`(rx, age, obstruct, differ, node4)`, $\boldsymbol{W}$=`(rx, obstruct, node4)`

Then we fit a HLM (1.1), the MLE of which, $(\hat{\beta}, \hat{\gamma})$ will be the starting value for the MCMC.

3.2.2 Result

We used 10000 MCMC iterations, and the acceptance rate for Metropolis updates for (γ, ψ, λ), (89%, 44%, 41%), are close to the optimal acceptance rate. We extracted the sample mean to be the point estimates for θ, i.e. $\hat{\theta} = (\hat{\beta}, \hat{\alpha}, \hat{\gamma}, \hat{\psi}, \hat{\lambda})$, then $y_i|\boldsymbol{x_i}, \boldsymbol{w_i} \sim GH(\boldsymbol{x_i}'\hat{\beta}, \hat{\alpha}, exp(\boldsymbol{w_i}'\hat{\gamma}/2), \hat{\psi}, \hat{\lambda})$. The below tables show the point and interval estimates for θ.

	Intercept	rxLev	rxLev+5FU	age	obstruct1	differ2	differ3	node41
2.5%	-1095.77	-279.74	-147.39	-12.83	-653.86	-90.93	-589.15	-1129.20
97.5%	-393.53	43.45	680.09	-4.50	-305.84	139.97	-194.27	-836.07
Point	-599.74	-136.09	200.86	-11.19	-590.49	49.74	-270.95	-879.17

Table 2: Point and Interval estimate for β

	Intercept	rxLev	rxLev+5FU	obstruct1	node41	α	ψ	λ
2.5%	3.12	-12.93	-4.05	-5.81	-11.84	1701.58	3.17	2.89
97.5%	12.35	1.03	8.72	1.70	-0.98	1791.47	4.58	4.12
Point	8.52	-6.24	2.77	-1.48	-5.76	1773.38	3.78	3.46

Table 3: Point and Interval estimate for $\gamma, \alpha, \psi, \lambda$

To assess the quality of prediction under this model, we first plot the estimated survival time, $E(y_i) = x_i'\beta + \alpha E(V)$, vs. observed survival time (only non-censored cases where time-to-death is observed) (Figure 3). We can see that the model in general overestimates the response, and the distribution shows a highly right-skewed pattern. This can be seen more clearly in Figure 4, where we randomly pick several non-censored individuals, and plot $p(y_i|\theta, x_i, w_i)$. Moreover, the average of the standard deviation of the survival time for an individual ($s.d.(Y_i|x_i, w_i, \theta) = \sqrt{\hat{\alpha}^2 var(V) + exp(w_i'\hat{\gamma})E(V)}$) is quite large (1685.13).

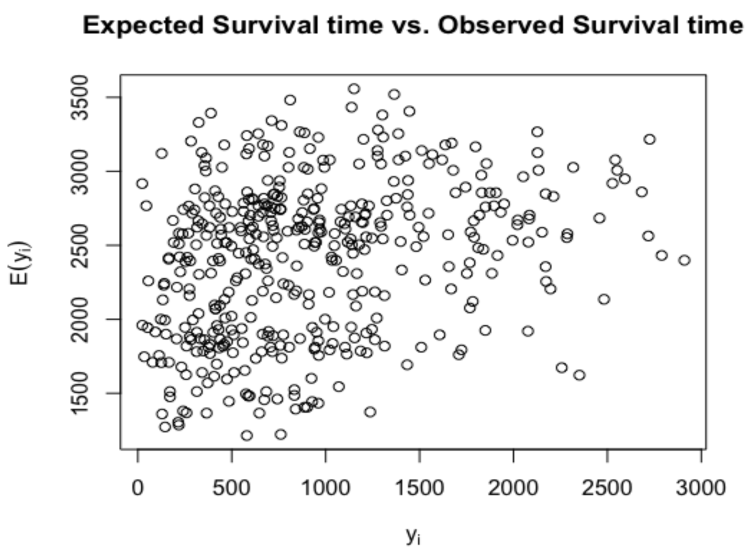

Figure 3

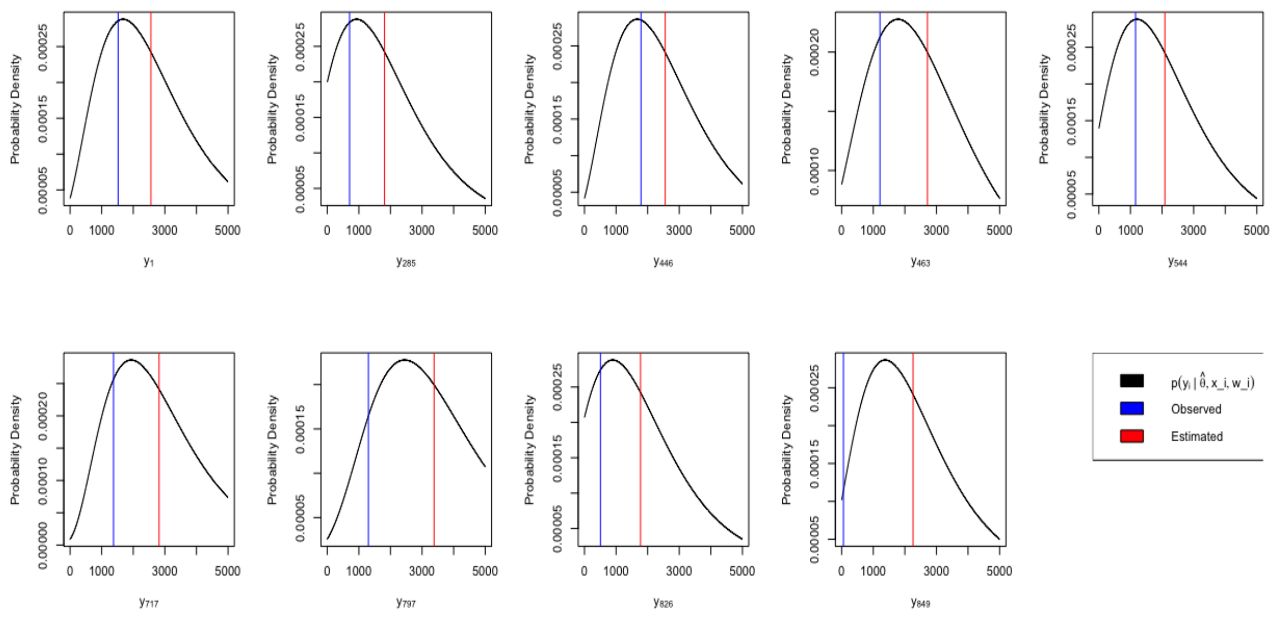

Figure 4. $p(y_i|\theta, x_i, w_i)$

Further investigation suggests that this is probably due to right-censoring effect. We ran another MCMC with only non-censored data. It turns out that the average standard deviation of the survival time is significantly smaller (585.92) compared with using complete data, and the estimated survival time is much closer to the observed survival time.

When we account for right-censoring effect, we sample the unobserved event time from a truncated normal with a lower bound, but without an upper bound. Therefore the sampled time is almost always much greater than most of the event time of non-censored cases. Despite that, it is still necessary for us to account for right-censoring. Although only sampling from $p(\theta|D\{\delta_i = 1\})$ greatly improves in-sample prediction, it can also introduce huge bias (especially in our case where more than 50% of the subjects are right-censored). Sampling the unobserved event time y_i in MCMC is equivalent to integrating it out, and thus gives a more flexible model which has better out-of-sample performance.

We also constructed 95% confidence intervals for $Y_i\{\delta_i = 1\}|x_i, w_i, \theta$ with lower bound being

0 (positive survival time). The rate of the CI's covering the observed response is 100%, but the average length of the CI's (6769.89) is large. To conclude, this model is flexible enough to give reasonable estimate for the time-to-death of colon cancer patients, but due to the large proportion of censored cases in the dataset, and probably also the problem of the MCMC itself described in section 3.1, the accuracy of the prediction is not very satisfactory.

4 Discussion

In this project we did a Generalized Hyperbolic Regression analysis to model survival data. The GHR model is able to capture the covariate-dependent location and scaling of the response like ordinary linear regression model, and it also has the flexibility to model the skewness and kurtosis of the response. However, when it comes to real-world data analysis, its performance can vary depending on the dataset or the quality of parameter inference.

In our project, we conducted Bayesian Inference to estimate parameters. And in this project specifically, it is preferred compared with frequentist's inference techniques like Expectation-Maximization algorithm because of the right-censoring effect. As we have seen in section 3.2.2, simply discarding censored observations is not a good choice. However in Bayesian Inference, the issue of parameter regularization also emerges. We did not use informative priors to prevent the parameters from drifting off because no sufficient information is available for the parameters. Hence, parameter regularization for Bayesian Inference of a normal mean-variance mixture model could be a direction of future research.

5 Appendix

References

Browne, R. P., & McNicholas, P. D. (2015, February 26). A mixture of generalized hyperbolic distributions. Retrieved from https://onlinelibrary.wiley.com/doi/full/10.1002/cjs.11246

Fox, J. (2008). Cox Proportional-Hazards Regression for Survival Data. Retrieved from https://socialsciences.mcmaste
1E/appendix-cox-regression.pdf.

LAGAKOS, S. W. (1979). General Right Censoring and Its Impact on the Analysis of Survival Data. Biometrics, 35, 139-156. Retrieved from http://www.jstor.org/stable/pdf/2529941.pdf

Long, J. S., & Ervin, L. H. (2000). Using Heteroscedasticity Consistent Standard Errors in the Linear Regression Model. The American Statistician, 54(3), 217-224. doi:10.1080/00031305.2000.10474549

Tanner, M. A., & Wong, W. H. (1987). The Calculation of Posterior Distribution by Data Augmentation. Journal of the American Statistical Association, 82, 528-540. Retrieved from https://www.jstor.org/stable/pdf/2289457.pdf.

Wei, L. J. (2007, January 22). The accelerated failure time model: A useful alternative to the cox regression model in survival analysis. Retrieved from https://onlinelibrary.wiley.com/doi/pdf/10.1002/sim.4780111409

R code

```
source("project-functions.R")
# experiment with simulated dataset
# simulate some data
n1<-500
X1<-rtnorm(n1, 10, 10, 5)
W1<-rtnorm(n1, 3, 3, 1)
beta1<-1
alpha1<-1
```

```
gamma1<-1
psi1<-20
lambda1<-20
V1<-rgig(n1, psi1, 1, lambda1) #V1 | X1, W1 ~ rgig(psi1, 1, lambda1)
mu1<-X1*beta1+alpha1*V1
sigma2<-exp(W1*gamma1)*V1
y1<-rnorm(n1, mu1, sqrt(sigma2)) # Y1 | V1, X1, W1 ~ N(mu1, sigma2)

#---------check likelihoods and analytic posteriors vs. mcmc samples----------
# ---------------------psi, lambda-------------------
# check unsimplified vs simplified log likelihoods
ntest<-10
Psi<-psi1+runif(ntest, 5, 10)
Lambda<-lambda1+runif(ntest, 5, 10)

llu<-sapply(1:ntest, function(ii){
  psi<-Psi[ii]
  lambda<-Lambda[ii]
  gh.loglik(y1, V1, mu1, alpha1, sqrt(sigma2), psi, lambda)
})

lls<-sapply(1:ntest, function(ii){
  psi.lam.loglik(V1, Psi[ii], Lambda[ii])
})

llu-lls

# check analytic posterior vs mcmc samples. Prior: psi~N(0,5)
# assuming beta, alpha, gamma, V are fixed
result.pl<-mcmc(20000, y1, X1, W1, beta1, alpha1, gamma1, V1, psi1, lambda1,
                  ba.fixed=TRUE, gam.fixed=TRUE, psl.fixed=FALSE, v.fixed=TRUE, v.out=FALSE, rwsd=
colMeans(cbind(result.pl$psi, result.pl$lambda))

pseq<-seq(5,15, length.out=500)
lseq<-seq(5,15, length.out=500)
psl<-as.matrix(expand.grid(pseq, lseq))
lp.mat<-apply(psl, 1, function(theta){psi.lam.loglik(V1, theta[1], theta[2])}) #+dnorm(theta[1],0,
lp.mat<-matrix(lp.mat, 500, 500)
psi.lam.stat<-grid.plot.stat(pseq, lseq, lp.mat, alpha = .95)

par(mfrow=c(1,2))
hist(result.pl$psi, breaks=100, prob=T, main=expression(p(psi*" | "*bold(y),bold(X),bold(W),alpha,
lines(psi.lam.stat$seq1, psi.lam.stat$dens1, type='l', col="red")
hist(result.pl$lambda, breaks=100, prob=T, main=expression(p(lambda*" | "*bold(y),bold(X),bold(W),
lines(psi.lam.stat$seq2, psi.lam.stat$dens2, type='l', col="red")

#------------------beta, alpha------------------
# check unsimplified vs simplified log likelihoods
Beta<-beta1+runif(ntest, 1, 2)
Alpha<-alpha1+runif(ntest, 1, 2)

llu<-sapply(1:ntest, function(ii){
  beta<-Beta[ii]
  alpha<-Alpha[ii]
  gh.loglik(y1, V1, beta*X1+alpha*V1, alpha, sqrt(sigma2), psi1, lambda1)
})

lls<-sapply(1:ntest, function(ii){
```

```r
  ba.loglik(y1, X1, W1, V1, Beta[ii], Alpha[ii], as.matrix(gamma1))
})

llu-lls

# check analytic posterior vs mcmc samples, assuming V, gamma, psi, lambda are fixed
result.ba<-mcmc(10000, y1, X1, W1, beta1, alpha1, gamma1, V1, psi1, lambda1,
                  ba.fixed=FALSE, gam.fixed=TRUE, psl.fixed=TRUE, v.fixed=TRUE, v.out=FALSE, rwsd=
colMeans(cbind(result.ba$beta, result.ba$alpha))

bseq<-seq(0.1,2, length.out=500)
aseq<-seq(-2,2, length.out=500)
ba<-as.matrix(expand.grid(bseq, aseq))
lp.mat<-apply(ba, 1, function(theta){ba.loglik(y1, X1, W1, V1, theta[1], theta[2], as.matrix(gamma
lp.mat<-matrix(lp.mat, 500, 500)
ba.stat<-grid.plot.stat(bseq, aseq, lp.mat, alpha = .95)

par(mfrow=c(1,2))
hist(result.ba$beta, breaks=100, prob=T, main=expression(p(beta*" | "*bold(y),bold(X),bold(W),gamm
lines(ba.stat$seq1, ba.stat$dens1, type='l', col="red")
hist(result.ba$alpha, breaks=100, prob=T, main=expression(p(alpha*" | "*bold(y),bold(X),bold(W),ga
lines(ba.stat$seq2, ba.stat$dens2, type='l', col="red")

#---------------------gamma------------------------
# check unsimplified vs simplified log likelihoods
Gamma<-gamma1+runif(ntest,1,2)

llu<-sapply(1:ntest, function(ii){
  gamma<-Gamma[ii]
  sigma<-sqrt(exp(W1*gamma)*V1)
  gh.loglik(y1, V1, beta1*X1+alpha1*V1, alpha1, sigma, psi1, lambda1)
})

lls<-sapply(1:ntest, function(ii){
  gamma.loglik(y1, X1, W1, V1, as.matrix(beta1), alpha1, as.matrix(Gamma[ii]))
})

llu-lls

# compare target (true) to proposal (approximation)
par(mfrow=c(1,1))
gmv<-gam.mv(y1, X1, W1, V1, as.matrix(beta1), alpha1)
gseq<-seq(0.8,1.2, length.out=500)
lp.dens<-sapply(1:length(gseq), function(ii){gamma.loglik(y1, X1, W1, V1, as.matrix(beta1), alpha1
gdens<-exp(lp.dens-max(lp.dens))
dg <- gseq[2]-gseq[1]
gdens <- gdens/sum(gdens)/dg
plot(gseq, gdens, type='l', lwd=1.5, xlab=expression(gamma), ylab="Density")
curve(dnorm(x, gmv$mode, sqrt(gmv$v)), add = TRUE, col = "red", lwd=1.5)
legend("topleft", legend = c("True Posterior", "Normal Approx."), fill = c("black", "red"))

# check analytic posterior vs mcmc samples. Assuming V, alpha, beta, psi, lambda fixed
result.gam<-mcmc(10000, y1, X1, W1, beta1, alpha1, gamma1, V1, psi1, lambda1,
                 ba.fixed=TRUE, gam.fixed=FALSE, psl.fixed=TRUE, v.fixed=TRUE, v.out=FALSE, rwsd=c
mean(result.gam$gamma)

hist(result.gam$gamma, breaks=100, prob=T, main=expression(p(gamma*" | "*bold(y),bold(X),bold(W),b
lines(gseq, gdens, col="red")
```

```
#----------------------missing data (V)----------------------------
# check unsimplified vs simplified log likelihoods
# only V varies, everything else fixed
X.sim<-rep(X1[1], ntest)
W.sim<-rep(W1[1], ntest)
V.sim<-rgig(ntest, psi1, 1, lambda1)
mu.sim<-X.sim*beta1+V.sim*alpha1
sig2.sim<-exp(W.sim*gamma1)*V.sim
#y.sim<-rnorm(ntest, mean=mu.sim, sd=sqrt(sig2.sim))
y.sim<-rep(y1[1], ntest)

ldu<-sapply(1:ntest, function(ii){
  gh.loglik(y=y.sim[ii], V.sim[ii], mu=mu.sim[ii], alpha=alpha1, sigma=sqrt(sig2.sim)[ii], psi=psi
})

lds<-sapply(1:ntest, function(ii){v.logpost(y.sim[ii], X.sim[ii], W.sim[ii], V.sim[ii], beta1, alp

ldu-lds


# pick a few Vi's to check mcmc samples matches analytic log posterior
result.v<-mcmc(10000, y1[1:20], X1[1:20], W1[1:20], beta1, alpha1, gamma1, V1, psi1, lambda1,
                  ba.fixed=TRUE, gam.fixed=TRUE, psl.fixed=TRUE, v.fixed=FALSE, v.out=TRUE, rwsd=c


Vind <- sort(sample(1:20, 6)) # pick a few Vi's at random
par(mfrow = c(2,3))
invisible(sapply(Vind, function(vind) {
  hist(result.v$V[,vind], breaks = 100, freq = FALSE,
       xlab = parse(text = paste0("v[", vind, "]")),
       main = parse(text = paste0("p(v[", vind,
                                  "]*\" | \"*theta,bold(D))")))
  vSeq <- range(result.v$V[,vind])
  vSeq <- seq(vSeq[1], vSeq[2], len = 100)
  vPdf <- v.logpost(y = y1[vind], X=X1[vind], W=W1[vind], V = vSeq,
                    beta = beta1, alpha = alpha1, gamma = gamma1, psi = psi1, lambda=lambda1)
  vPdf <- exp(vPdf - max(vPdf))
  vPdf <- vPdf/sum(vPdf)/(vSeq[2]-vSeq[1])
  lines(vSeq, vPdf, col = "red")
}))

#------------------sample from joint posterior------------------------------
result1<-mcmc(10000, y1, X1, W1, beta0=5, alpha0=5, gamma0=gamma1, V0=V1, psi0=5, lambda0=5,
              ba.fixed=FALSE, gam.fixed=FALSE, psl.fixed=FALSE, v.fixed=FALSE, v.out=FALSE, rwsd=c

# point estimates and credible intervals
post.mean<-c(mean(result1$beta), mean(result1$alpha), colMeans(result1$gamma), mean(result1$psi),
names(post.mean)<-c("beta", "alpha", "gamma", "psi", "lambda")
sapply(1:5, function(i){quantile(result1[[i]], c(0.025,0.975))})

# MCMC histogram, true parameter values vs posterior means
theta.true<-c(beta1, alpha1, gamma1, psi1, lambda1)
par(mfrow = c(2,3))
invisible(lapply(1:5, function(i){
  hist(result1[[i]], prob=T, breaks=100, xlab=parse(text=names(result1)[i]),
       main=parse(text = paste0("p(", names(result1)[i], ")")))
  abline(v=theta.true[i], col="blue")
  abline(v=post.mean[i], col="red")
  }))
plot(-10:0, xlim=c(1,10), ylim=c(1,10), xlab="", ylab="", xaxt='n', yaxt='n', axes=FALSE)
legend("topleft", legend = c("True Parameter", "Posterior Mean"), fill = c("blue", "red"), cex=1.5

# another MCMC run, this time with a prior imposed on psi.
result2<-mcmc(10000, y1, X1, W1, beta0=5, alpha0=5, gamma0=gamma1, V0=V1, psi0=5, lambda0=5,
              ba.fixed=FALSE, gam.fixed=FALSE, psl.fixed=FALSE, v.fixed=TRUE, v.out=FALSE, rwsd=c(

# compare the true pdf of y_i|x_i, w_i, theta0 vs the pdf of y_i|x_i, w_i, theta,
# where theta is the point estimates of the previous two mcmc runs
par(mfrow=c(2,3))
invisible(sapply(yind, function(i) {
  plot(x=yseq, y=dghyp(yseq, X1[i]*beta1, alpha1, sqrt(exp(W1[i]*gamma1)), psi1, lambda1),type='l'
       xlab = parse(text = paste0("y", i)), ylab="Density")
  lines(yseq, dghyp(yseq, X1[i]*post.mean["beta"], post.mean["alpha"], sqrt(exp(W1[i]*post.mean["g
                    post.mean["psi"], post.mean["lambda"]), type='l', ylab="Density", ylim=c(0,0.0
  lines(yseq, dghyp(yseq, X1[i]*post.mean2["beta"], post.mean2["alpha"], sqrt(exp(W1[i]*post.mean2
                     post.mean2["psi"], post.mean2["lambda"]),col="red")
}))
plot(-10:0, xlim=c(1,10), ylim=c(1,10), xlab="", ylab="", xaxt='n', yaxt='n', axes=FALSE)
legend("topleft", legend = c(expression(p(y[i]*" | "*theta[0],x[i],W[i])), expression(p(y*" | "*th
       fill = c("black","blue", "red"), cex=1.5)

#--------------------------------------------------------------
require("survival")
source("ghyp-functions.R")
source("hlm-functions.R")

# data analysis with colon dataset
# preprocess dataset
colon1<-colon[colon$etype==2,-2]
rownames(colon1)<-seq(length=nrow(colon1))
colon1$sex<-as.factor(colon1$sex)
colon1$obstruct<-as.factor(colon1$obstruct)
colon1$perfor<-as.factor(colon1$perfor)
colon1$adhere<-as.factor(colon1$adhere)
colon1$status<-as.factor(colon1$status)
colon1$differ<-as.factor(colon1$differ)
colon1$extent<-as.factor(colon1$extent)
colon1$surg<-as.factor(colon1$surg)
colon1$node4<-as.factor(colon1$node4)
colon1$etype<-as.factor(colon1$etype)
colon1$nodes<-as.integer(colon1$nodes)

# identify observations with missing covariates
colon1$miss.nodes<-FALSE
colon1$miss.nodes[is.na(colon1$nodes)]<-TRUE
colon1$miss.differ<-FALSE
colon1$miss.differ[is.na(colon1$differ)]<-TRUE

miss.ids<-colon1[c(colon1$id[colon1$miss.nodes],colon1$id[colon1$miss.differ]),"id"]
non.miss.ids<-colon1$id[-miss.ids]
colon.non.miss<-colon1[non.miss.ids,]
censor.id<-colon.non.miss[colon.non.miss$status==0,]$id
censor.id<-as.character(censor.id)
noncensor.id<-colon.non.miss[colon.non.miss$status==1,]$id

summary(lm(time~rx+sex+age+obstruct+perfor+adhere+nodes+differ+extent+surg+node4-1,data=colon1))
```

```r
M.lm<-lm(time~rx+sex+age+obstruct+perfor+adhere+nodes+differ+extent+surg+node4-1,data=colon1)
summary(glm((M.lm$residuals)^2~model.matrix(M.lm)-1,family=Gamma("log")))

# Determine significant covariates using an HLM setting, and fit an HLM as the
# starting values for beta and hat in MCMC.
X<-model.matrix(lm(time~rx+age+obstruct+differ+node4, data=colon1[non.miss.ids,]))
W<-model.matrix(lm(time~rx+obstruct+node4, data=colon1[non.miss.ids,]))
y<-colon1$time[non.miss.ids]
names(y)<-non.miss.ids
M.hlm<-hlm.fit(y=y,X,W)
beta0<-M.hlm$beta
gamma0<-M.hlm$gamma
alpha0<-10
psi0<-10
lambda0<-10
V0<-rgig(nrow(X),psi=psi0, eta=1, lambda=lambda0)

# run MCMC
result<-mcmc(nsamples=10000, y=y, X=X, W=W, beta0=beta0, alpha0=alpha0, gamma0=gamma0, V0=V0, psi
      censor.id=censor.id, censored=TRUE, rwsd=c("psi"=0.2, "lambda"=0.2))

# posterior sample means - point estimate for theta
beta.hat<-colMeans(result$beta)
alpha.hat<-mean(result$alpha)
gamma.hat<-colMeans(result$gamma)
psi.hat<-mean(result$psi)
lambda.hat<-mean(result$lambda)

mu.hat<-X%*%beta.hat
sigma.hat<-sqrt(exp(W%*%gamma.hat))

# confidence intervals for parameters
beta.CI<-sapply(1:ncol(X), function(i) quantile(x=(result$beta)[,i], probs=c(0.025, 0.975)))
colnames(beta.CI)<-names(beta0)
beta.est<-rbind(beta.CI, beta.hat)

gamma.CI<-sapply(1:ncol(W), function(i) quantile(x=(result$gamma)[,i], probs=c(0.025, 0.975)))
colnames(gamma.CI)<-names(gamma0)
gamma.est<-rbind(gamma.CI, gamma.hat)

agl.CI<-cbind(quantile(result$alpha, probs=c(0.025, 0.975)), quantile(result$psi, probs=c(0.025, 
names(agl.CI)<-c("alpha", "psi", "lambda")
agl.est<-rbind(agl.CI, c(alpha.hat, psi.hat, lambda.hat))

# E(y) and var(y)
# E(y)=X%*%beta+alpha*E(V), only for non-censored cases
EV<-besselK(psi.hat, nu = lambda.hat+1, expon.scaled = TRUE)/besselK(psi.hat, nu = lambda.hat, exp
Ey<-X%*%beta.hat+alpha.hat*EV
names(Ey)<-non.miss.ids
names(mu.hat)<-non.miss.ids
names(sigma.hat)<-non.miss.ids

# var(y)=alpha^2*var(V)+exp(W%*%gamma)*E(V)
varV<-besselK(psi.hat, nu = lambda.hat+2, expon.scaled = TRUE)/besselK(psi.hat, nu = lambda.hat, e
  (besselK(psi.hat, nu = lambda.hat+1, expon.scaled = TRUE)/besselK(psi.hat, nu = lambda.hat, expo
vary<-alpha.hat^2*varV+exp(W[colon.non.miss$status==1, ]%*%gamma.hat)*EV
sqrt(vary)
mean(sqrt(vary))
```

```
# compare observed vs predicted life time
cbind(colon.non.miss$time[colon.non.miss$status==1], Ey[colon.non.miss$status==1, ])
plot(colon.non.miss$time[colon.non.miss$status==1], Ey[colon.non.miss$status==1, ], xlab=expressio
     ylab=expression(E(y[i])), main="Expected Survival time vs. Observed Survival time")

# visual check: p(y_i|x_i, w_i, theta.hat)
yind<-sort(sample(noncensor.id, 9))
par(mfrow=c(2,5))
invisible(
  sapply(yind, function(i){
    yseq<-seq(1,5000,length.out = 2000)
    plot(yseq, dghyp(yseq, mu.hat[names(mu.hat)==i], alpha.hat, sigma.hat[names(sigma.hat)==i], ps
        type='l', xlab=parse(text=paste0("y[",i,"]")), ylab="Probability Density", cex=1.5)
    abline(v=c(y[names(y)==i], Ey[names(Ey)==i]),col=c("blue","red"))
}))
plot(-1, xlim=c(0,1), ylim=c(0,1),xlab="",ylab="",xaxt='n',yaxt='n', axes=FALSE)
par(cex=0.7)
legend("topleft", legend=c(expression(p(y[i]*" | "*hat(theta), x_i, w_i)), "Observed", "Estimated"

# check coverage of confidence interval (0, Upper bound)
require(ghyp)
CI<-sapply(noncensor.id, function(i){
  GH.hat<-ghyp(lambda=lambda.hat, chi=psi.hat, psi=psi.hat, mu=as.numeric(mu.hat[row.names(mu.hat)
               sigma=as.numeric(sigma.hat[rownames(sigma.hat)==i]), gamma=alpha.hat)
  lower.p<-round(ghyp::pghyp(0, GH.hat),8)
  ghyp::qghyp(1-lower.p, GH.hat)
})
```