Lung Cancer Survival Prediction with Bayesian Generalised Linear Models

Yann McLatchie and Arina Odnoblyudova

Contents

1 Introduction

Lung cancer is one of the most common types of cancer in both men and women. The analysis of survival time of patients with lung cancer is crucial for controlling the disease's development, determining the optimal course of treatment, and understanding what influences the disease's progression. To accomplish these aims, accurate survival time analysis methods are crucial.

Survival analysis is the combination of different statistical methods for analysing time-to-event data. Executing a survival analysis can be challenging, since both non-parametric and parametric models can be used, and various distributions such as exponential, Weibull, and log-normal are often all applicable. The most common model is the Cox hazard model which proposes a constant effect of predictor variables on survival duration (known as the hazard function) throughout time, which we know to be unrealistic, and thus consider to be too simple for this case.

This study employs a Bayesian Weibull Generalised Linear model (GLM) of the survival time of patients with advanced lung cancer. The Weibull approach is more flexible than Cox, and the hazard rate is not assumed to be constant with time. For simulation Bayesian inference with Markov Chain Monte Carlo (MCMC) is used, providing us with a satisfying approximation of uncertainty and ability to use priors from domain knowledge. The model is implemented and tested with the help of the R and Stan packages.

We will investigate the efficacy of three models: a model with data pooled across insitutions and trained on only observed data, a similarly pooled model using censored data, and a hierarchical model using only observed data. The respective models will eventually be compared with regards to their predictive power, and we will conclude that our hierarchical model is the most performant and conclude with a discussion on its implications.

The code for the report, complete with model implementations in R and Stan, is provided in McLatchie and Odnoblyudova [2021].

1.1 Data description

1.1.1 General description

The data used in the study include the survival time of patients with advanced lung cancer from the North Central Cancer Treatment Group. It is provided in the `survival` R package from Therneau [2021]. We will aim to predict the survival time of patients with lung cancer given some covariates.

Our dataset contains 9 features and 149 observations, which we divide into 7 institutional groups, and which we assume to be independent and identically distributed. The target variable is the survival time in days. The covariates are of both categorical and numerical data types.

Special attention has to be paid to the "censored status" indicator. It indicates if the patient had some terminal event (in which case it is equal to 1) or not (equal to 0). If the data for a patient are censored, their true survival time is not known. A right censoring approach is used, meaning incompleteness of survival time at the right side of the follow-up period.

Our variables, and their descriptions adapted from Therneau [2021], are as follows:

Variable	Description
`inst`	Institution code
`time`	Survival time in days
`status`	censoring status 1=censored, 2=dead
`age`	Age in years
`sex`	Male=1 Female=2
`ph.ecog`	ECOG performance score (0-4). 0-good condition, 4-the worst condition.
`ph.karno`	Karnofsky performance score (bad=0-good=100). Provided by physician.
`pat.karno`	KKarnofsky performance score. Provided by patient.
`meal.cal`	Calories consumed at meals
`wt.loss`	Weight loss in last six months (pounds)

1.1.2 Exploratory data analysis

We provide some descriptive statistics to familiarise ourselves with the data, beginning by importing the necessary packages.

```
library(dplyr)
library(ggplot2)
data("cancer", package = "survival")
```

There were 44 observations with missed values, which have been removed from dataset, leaving 105 observations.

```
data = cancer %>%
  filter(inst %in% c(1, 12, 13, 3, 11, 22, 16)) %>%
  na.omit()
```

Institutions are considered as grouping variables, used later in the hierarchical model. Note that to make this analysis more manageable, we are only considering data from the top 7 institutions in terms of number of observations for all models.

```
table(data$inst)
```

```
##
##  1  3 11 12 13 16 22
```

```
## 28 12 13 16 13 10 13
```

Moving to categorical variables below in Figure 1, we have more men than women in our data, and note that the majority of patients are ambulatory with symptoms.

```
par(mfrow=c(1,2))
barplot(table(data$sex), main="Sex statistics", names.arg=c("male", "female"),
        col=c("steelblue","cornflowerblue"))
barplot(table(data$ph.ecog), main="ECOG score statistics",
        legend = c("asymptom.", "ambulatory", "in bed <50% of t", "in bed >50% of t"),
        args.legend = list(x = "topright",inset = c(- 0.15, 0)),
        col=c("steelblue","cornflowerblue","blue","darkblue"))
```

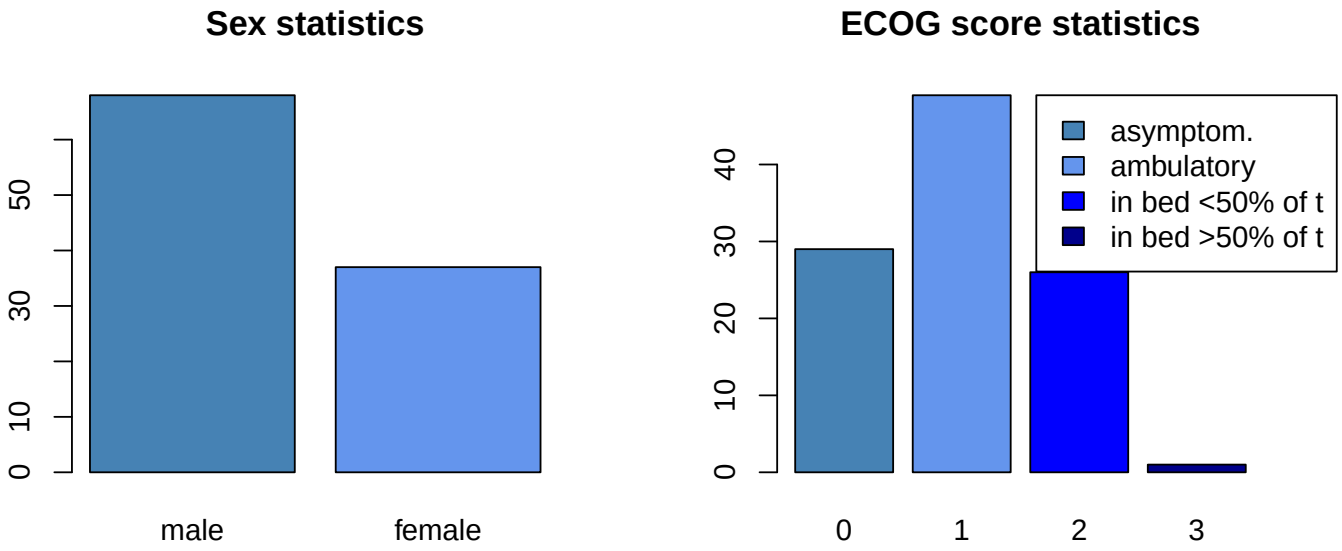


Figure 1: Categorical variables

Investigating the distribution of continuous variables in Figure 2, we find that the features do not follow a normal distribution.

```
par(mfrow=c(3,2))
hist(data$age, freq=FALSE, col="cornflowerblue", main="Histogram of age",xlab="")
hist(data$ph.karno, freq=FALSE, col="cornflowerblue", main="Histogram of ph.karno",xlab="")
hist(data$pat.karno, freq=FALSE, col="cornflowerblue", main="Histogram of pat.karno",xlab="")
hist(data$meal.cal, freq=FALSE, col="cornflowerblue", main="Histogram of meal.cal",xlab="")
hist(data$wt.loss, freq=FALSE, col="cornflowerblue", main="Histogram of wt.loss",xlab="")
```

It is also useful to identify any linear correlations between variables. The correlation is not that high across the board, and only the pair `ph.karno` and `pat.karno` are highly correlated (0.525), but this is intuitively reasonable given the previous description of these covariates, since patients and doctors may measure the same quantity, or agree on the quantity measured.

```
cor(data[c(4,7,8,9,10)], method=c("pearson"))
```

```
##                 age    ph.karno  pat.karno    meal.cal     wt.loss
## age        1.0000000 -0.27886148 -0.3516302 -0.24502394  0.10546384
## ph.karno  -0.2788615  1.00000000  0.6000046  0.03206143 -0.13154644
## pat.karno -0.3516302  0.60000461  1.0000000  0.21500299 -0.22162023
## meal.cal  -0.2450239  0.03206143  0.2150030  1.00000000 -0.03409461
## wt.loss    0.1054638 -0.13154644 -0.2216202 -0.03409461  1.00000000
```

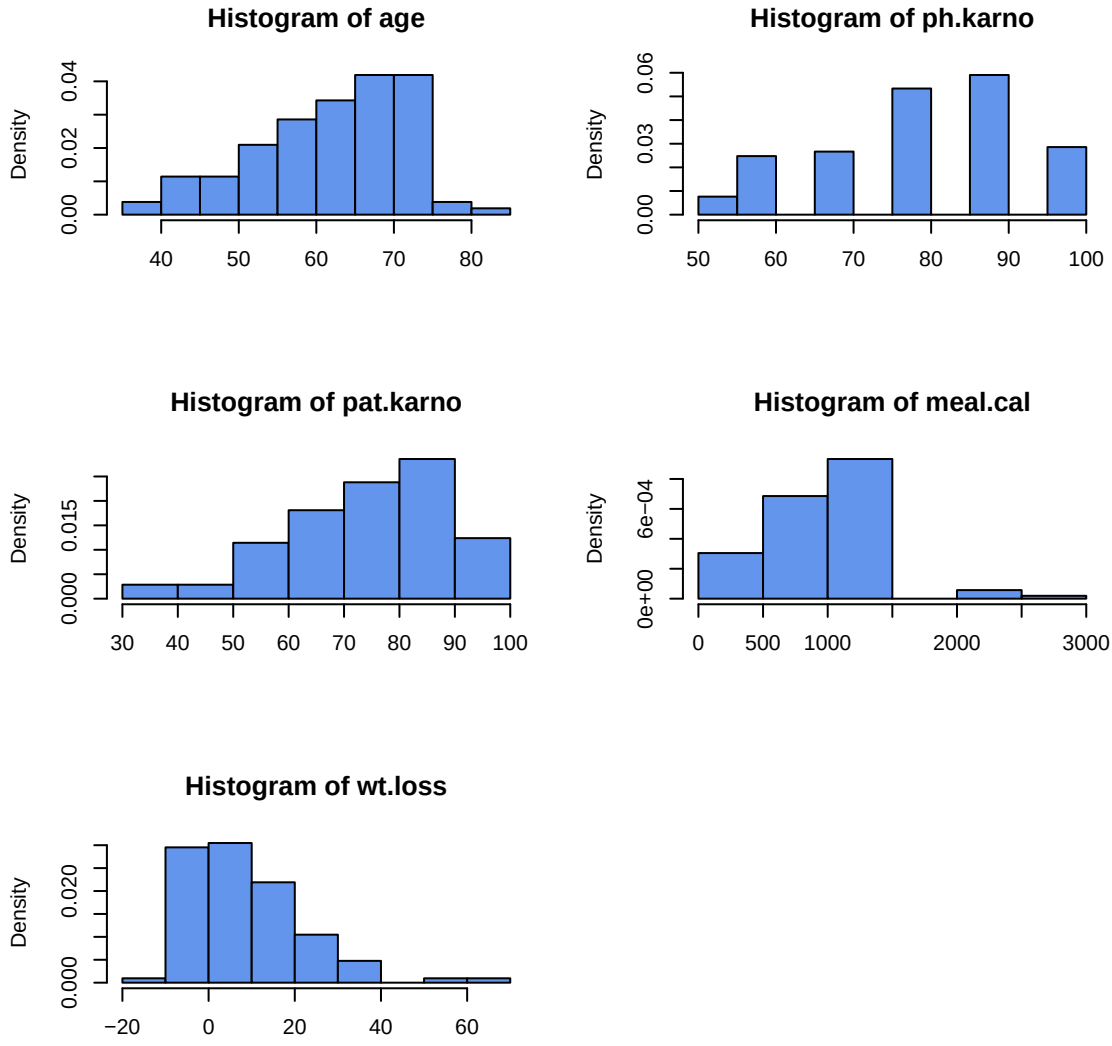


Figure 2: Continuous variables

1.2 Related studies

The original data were presented in Loprinzi et al. [1994], where the authors just provided descriptive information from a lung patient-completed questionnaire, which was aggregated into the given dataset. In Kumar and Sonker [2020] the comparison of semi-parametric and non-parametric models for survival analysis was presented, but a different dataset was used, and there was no serious description of Weibull model implementation. Instead, more attention was payed to Cox regression. Abujarad and Khan [2018] provided a description of exponential models applied to a lung cancer survival time analysis with Stan code, but the Weibull distribution was only mentioned in passing and no models or conclusions were derived for it. That is why in our work three Weibull Survival models are built: one hierarchical and two pooled which will all be described in the following section. The main approaches and survival analysis theory were inspired from the studies above, but the Stan implementation, data preprocessing, choice of priors, and interpretation were all performed by the authors.

2 Description of models

In the following section, we will motivate and define mathematically our three GLMs implemented in Stan for survival time prediction. We begin by importing the necessary packages and setting a random seed for reproducibility.

```
# install libraries
library(survival)
library(tidyverse)
library(rstan)
library(bayesplot)
library(loo)
library(ggplot2)
library(data.table)
# set number of cores
options(mc.cores = parallel::detectCores())
# read lung cancer data from `survival` library
data("cancer", package = "survival")
# set random seed for reproducibility
set.seed(2021)
```

2.1 Weibull without censored data

Our first model will be a pooled model, not considering any censored data points. Let $y \sim \text{Weibull}(\alpha, \sigma)$, we define the probability density function or our Weibull-distributed y as

$$\text{Weibull}(y|\alpha,\sigma) = \frac{\alpha}{\sigma}\left(\frac{y}{\sigma}\right)^{\alpha-1}\exp\left(-\left(\frac{y}{\sigma}\right)^{\alpha}\right),$$

for $y \in [0,\infty), \alpha \in \mathbb{R}^+$, and $\sigma \in \mathbb{R}^+$.

2.1.1 Motivating the distribution

The Weibull distribution is often used as a more flexible and complex alternative to the semi-parametric proportional hazard Cox model for modelling time to failure events, since the hazard rate is not taken to be constant with time. We will use this as our base for all models in this analysis to differentiate ourselves from previous analyses, and hopefully outperform them.

2.1.2 The Weibull distribution as a member of the Exponential family

A probability distribution $f(y|\vartheta)$ is a member of the Exponential family if it can be written in the form

$$f(y|\vartheta) = h(y)\exp[\eta(\vartheta)\cdot T(y) - A(\vartheta)],$$

for some arbitrary parameter functions $h(\cdot), T(\cdot), A(\cdot)$, and canonical parameter $\eta(\vartheta)$ as a function of the parameters $\vartheta = (\sigma, \alpha)^T$. Now take α fixed and finite, then it can be shown that the Weibull distribution belongs to the exponential family since we can write it's probability density function as

$$\text{Weibull}(y|\sigma) = \alpha y^{\alpha-1}\exp(-y^{\alpha}\sigma^{-\alpha} - \alpha\log\sigma),$$

with

$$\begin{aligned} b(y) &= \alpha y^{\alpha-1} \\ \eta(\vartheta) &= \sigma^{-\alpha} \\ T(y) &= -y^{\alpha} \\ a(\eta) &= \alpha\log\sigma. \end{aligned}$$

This will make inference in the future easier for us.

2.1.3 Defining the link function

Looking at our canonical statistic $\eta = \sigma^{-\alpha}$, it can be shown that

$$\sigma = \exp\left(\frac{\log\eta}{-\alpha}\right)$$

where we construct $\eta = \exp(\boldsymbol{X}\beta)$ so that η is always strictly positive given any linear combination of the covariates, and the logarithm is then defined. Thus we choose a log link function for our GLM such that

$$\sigma = \exp\left(-\frac{\boldsymbol{X}\beta}{\alpha}\right),$$

which resembles a vanilla logarithmic inverse link function, with the additional α term and the negation.

2.1.4 Priors

In our Stan model, we will enforce priors over each of the regressors in the linear model, β, and the shape parameter α of our Weibull distribution. Mathematically, where we have N data points and M covariates, the model is defined as

$$\begin{aligned} y_i &\sim \text{Weibull}(\sigma, \alpha)\,, \quad i = 1, \ldots, N, \\ \alpha &\sim \text{Half-Cauchy}(5), \\ \sigma &= \exp\left(-\frac{\boldsymbol{X}\beta}{\alpha}\right), \\ \beta_j &\sim N(0, 1)\,, \quad j = 1, \ldots, M. \end{aligned}$$

The choice of a Half-Cauchy prior is motivated by Gelman [2006], so that the posterior is more flexible than the Gamma alternative, and the data can express themselves more freely. Note that this is a specific case of the of the conditionally-conjugate folded-noncentral-t family of prior distributions. In short, it acts as an appropriate weakly-informative prior satisfying the positive constraint, and given that we expect the value of α to be below 10 but allow for some deviance from this range. Our Gaussian prior over the regressors is easily motivated by the

parameterisation of the data, and is standard in literature. We will mean-center the design matrix before performing our regression in a `transformed data` block. This standardisation will hopefully reduce variance in our regressors and make learning their weights easier. We expect the weights of the regressors to be close to zero, and we allow them to be negative, making this an appropriate prior. In fact, the fact that the variance is not very large should aid convergence later, since our priors are stronger than for example a $N(0, 100)$ prior.

In this pooled model, we model these parameters using data from all institutions. Intuitively, this means that we expect the hazard to be equivalent regardless of which institution a patient is in. This is seen visually in the directed acyclic graph in Figure 3 below.

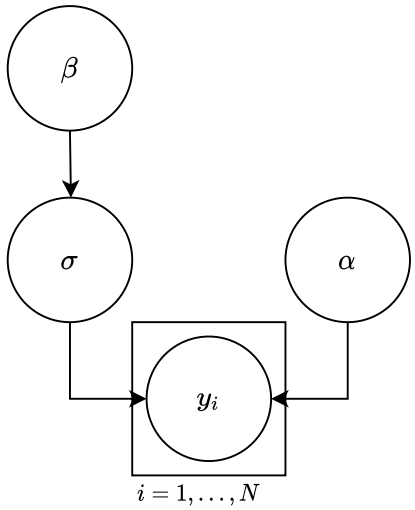


Figure 3: Pooled model DAG

2.1.5 Implemented in Stan

Below, we fit the model in Stan and output its code for the reader. Convergence and other diagnostics will be presented in the following section for all models.

```
# build dataset from only those non-censored data points
uncensored_data <- cancer %>%
  filter(status == 2) %>%
  filter(inst %in% c(1, 12, 13, 3, 11, 22, 16)) %>%
  drop_na()
# identify covariate labels and build design matrix
cov_labels <- uncensored_data %>%
  dplyr::select(-status,-time,-inst) %>%
  colnames()
# build design matrix
X <- as.matrix(uncensored_data[cov_labels])
# print(dim(X))
# [1] 120   7
y <- uncensored_data$time
# build data list for Stan model
weibull_data = list(
  y = y, X = X, N = length(y), M = ncol(X)
)
# compile and run seperate model
wm <- rstan::stan_model(file = "../stan/weibull_survival.stan")
```

```
## Running /anaconda3/envs/seurat4/lib/R/bin/R CMD SHLIB foo.c
## x86_64-apple-darwin13.4.0-clang -I"/anaconda3/envs/seurat4/lib/R/include" -DNDEBUG   -I"/anaconda3/envs
## In file included from <built-in>:1:
## In file included from /anaconda3/envs/seurat4/lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/
## In file included from /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Dense:1:
## In file included from /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Core:88:
## /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unkn
## namespace Eigen {
## ^
## /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error: exp
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /anaconda3/envs/seurat4/lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/
## In file included from /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Dense:1:
## /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file n
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [/anaconda3/envs/seurat4/lib/R/etc/Makeconf:174: foo.o] Error 1
```

```
# print out Stan code
print(wm)
```

```
## S4 class stanmodel 'weibull_survival' coded as follows:
## data {
##   int<lower=0> N; // number of data realisations
##   int<lower=0> M; // feature dimensionality
##   vector<lower=0>[N] y; // survival time
##   matrix[N, M] X; // design matrix
## }
##
## transformed data {
##   matrix[N, M] Xc;  // centered version of X without an intercept
##   vector[M] means_X;  // column means of X before centering
##
##   // column-center the design matrix for fitting the model
##   for (m in 1:M) {
##     means_X[m] = mean(X[, m]);
##     Xc[, m] = X[, m] - means_X[m];
##   }
## }
##
## parameters {
##   // GLM parameters
##   vector[M] beta;      // regressors
##   real<lower=0> alpha; // shape parameter
## }
##
## transformed parameters {
##   // compute latent predictor term
##   vector[N] eta = Xc * beta;
##   // apply the log inverse link function
##   vector<lower=0>[N] sigma = exp(-eta / alpha);
```

```
## }
##
## model {
##   // prior over regressor and shape parameters
##   beta ~ normal(0, 1);
##   alpha ~ cauchy(0, 5);
##
##   // fit model
##   y ~ weibull(alpha, sigma);
## }
##
## generated quantities {
##   // compute predictive distribution for survival time
##   real ypred[N] = weibull_rng(alpha, sigma);
##
##   // log-likelihood
##   vector[N] log_lik;
##   for (n in 1:N) {
##     log_lik[n] = weibull_lpdf(y[n] | alpha, sigma[n]);
##   }
## }
```

```
# learn the model parameters
weibull_model <- rstan::sampling(wm, iter = 10000, data = weibull_data)
```

2.2 Weibull with censored data

The density function for Weibull distributed survival times is given as

$$p(t_i|\alpha, \lambda_i) = \alpha t_i^{\alpha-1} \exp\left(\lambda_i - \exp \lambda_i t_i^{\alpha}\right),$$

and can be rewritten as

$$p(t_i|\alpha, \gamma_i) = \exp\left(-\left(\frac{t_i}{\gamma_i}\right)^{\alpha}\right)\frac{\alpha}{\gamma_i}\left(\frac{t_i}{\gamma_i}\right)^{\alpha-1},$$

where α is the shape parameter, and γ the scale. We move on to define a new variable λ is created, defined in relation to γ as

$$\lambda = -\alpha \log \gamma.$$

The survival function, showing the probability that the death will be after a certain time t, is then

$$S(t_i|\alpha, \lambda_i) = \exp(-\exp(\lambda_i)t_i^{\alpha}).$$

The likelihood of α and λ follows the equation below, with v_i an indicator showing 0 if the data are censored and 1 if not,

$$L(\alpha, \lambda|t) = \prod_{i=1}^{n} p(t_i|\alpha, \lambda_i)^{v_i} S(t_i|\alpha, \lambda_i)^{1-v_i} = \prod_{i=1}^{n} (\alpha t_i^{\alpha-1} exp(\lambda_i))^{v_i} (exp(-exp(\lambda_i)t_i^{\alpha})).$$

If $\lambda = X\beta$, than log-likelihood function can be expressed as,

$$l(\alpha, \beta|t, x) = \sum_{i=1}^{n} v_i(\log(\alpha) + (\alpha - 1)log(t_i) + X_i\beta) - \exp(X_i\beta)t_i^{\alpha}.$$

If the data are censored, then the log-likelihood is simply the logarithm of the survival function. In Stan this can be expressed with `weibull_lccdf()` function, corresponding to the log of the Weibull complementary cumulative distribution function of y given shape α and scale σ, and it is exactly the logarithm of survival function. For clarity, the complementary cumulative distribution function is

$$\bar{F}_X(x) = P(X > x) = 1 - F_X(x),$$

where $F_X(x)$ is the cumulative distribution function of a random variable.

2.2.1 Priors

The same priors are used for the censored model as for the pooled model. Even we are not expert oncologists, we know that typically person's life with advanced lung cancer is a little bit lower than a year, some patients live even for three years. The chosen prior is suitable, as it allows to produce survival time values, which are not too strict and at the same time really unlike to be larger than 5 years, for example.

Chosen priors do not contribute strongly to the posterior, so the data can express themselves. Algebraically then, the model is given as

$$\begin{aligned}
y_i &\sim \begin{cases} \text{Weibull}(\sigma, \alpha)\,, & \text{if } y_i \text{ is observed,} \\ \text{log-likelihood Weibull}(\sigma, \alpha)\,, & \text{if } y_i \text{ is censored,} \end{cases} \\
\alpha &\sim \text{Half-Cauchy}(5), \\
\sigma &= \exp\left(-\frac{\boldsymbol{X}\beta}{\alpha}\right), \\
\beta_j &\sim N(0, 1)\,.
\end{aligned}$$

Note that this is graphically equivalent to the pooled model.

2.2.2 Implemented in Stan

We once more execute some data preprocessing.

```
# read lung cancer data from "survival' library
data("cancer", package = "survival")

# omittimg NAs
data = cancer %>%
  filter(inst %in% c(1, 12, 13, 3, 11, 22, 16)) %>%
  na.omit()

# Censoring status is transformed to the column with 0-censored, 1-observed.
# The continuous variables are centered.
X=data
X$status[X$status==1] = 0
X$status[X$status==2] = 1
#X$male = ifelse(X$sex==1,1,0)
#X$female = ifelse(X$sex==2,1,0)
X$age=X$age-mean(X$age)
```

```
X$meal.cal=X$meal.cal-mean(X$meal.cal)
X$wt.loss=X$wt.loss-mean(X$wt.loss)

Xcens=X[X$status==0,]
Xcens = Xcens[-c(1,2,3)]
ycens=X$time[X$status==0]

Xobs=X[X$status==1,]
Xobs = as.matrix(Xobs[-c(1,2,3)])
yobs=X$time[X$status==1]
```

And collate the data in such a way as to present them to the Stan model.

```
data_model = list(
  yobs = yobs,
  Xobs = Xobs,
  N = nrow(Xobs),
  M = ncol(Xobs),
  ycen = ycens,
  Xcen = Xcens,
  Ncen = nrow(Xcens)
)
```

We are now able to run the model and display the Stan code.

```
# compile and run censored model
cwm = rstan::stan_model(file = "../stan/weibull_censored.stan")
```

```
## Running /anaconda3/envs/seurat4/lib/R/bin/R CMD SHLIB foo.c
## x86_64-apple-darwin13.4.0-clang -I"/anaconda3/envs/seurat4/lib/R/include" -DNDEBUG   -I"/anaconda3/envs
## In file included from <built-in>:1:
## In file included from /anaconda3/envs/seurat4/lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/
## In file included from /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Dense:1:
## In file included from /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Core:88:
## /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unkn
## namespace Eigen {
## ^
## /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error: exp
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /anaconda3/envs/seurat4/lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/
## In file included from /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Dense:1:
## /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file n
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [/anaconda3/envs/seurat4/lib/R/etc/Makeconf:174: foo.o] Error 1
```

```
# print out Stan code
print(cwm)
```

```
## S4 class stanmodel 'weibull_censored' coded as follows:
## data {
##   int<lower=0> N;    // number of object
```

```
##   int<lower=0> M;    // number of features
##   int<lower=0> Ncen;    // number of object
##   vector<lower=0>[N] yobs; // target-survival time
##   matrix[N, M] Xobs;    // covariates
##   vector<lower=0>[Ncen] ycen; // target-survival time
##   matrix[Ncen, M] Xcen;    // covariates
## }
##
## parameters {
##   vector[M] beta;       // regressors
##   real<lower=0> alpha;  // shape parameter
## }
##
## transformed parameters {
##   // Log inverse link function
##   vector<lower=0>[N] sigma = exp(-Xobs*beta / alpha);
## }
##
## model {
##   // priors
##   beta ~ normal(0, 10);
##   alpha ~ cauchy(0, 5);
##
##   // fitting model
##   yobs ~ weibull(alpha, sigma);
##
##   // Increment log-density with Survival Function
##   target += weibull_lccdf(ycen | alpha, exp(-Xcen*beta / alpha));
## }
##
## generated quantities {
##   // compute predictive distribution for survival time
##   real ypred[N] = weibull_rng(alpha, sigma);
##
##   // log-likelihood
##   vector[N+Ncen] log_lik;
##   for (i in 1:N) {
##     log_lik[i] = weibull_lpdf(yobs[i] | alpha, sigma[i]);
##   }
##   // Survival function
##   for (j in 1:Ncen){
##     log_lik[N+j] = weibull_lccdf(ycen[j] | alpha, exp(-Xcen[j,]*beta / alpha));
##   }
## }
```

```
# learn the model with parameters 4 chains, 10000 iterations for each, 5000 iterations for warm-up
weibull_cens = rstan::sampling(cwm, data = data_model, iter = 10000)
```

2.3 Hierarchical Weibull without censored data

Here we implement a model with some global shape parameter α to be learned as in the pooled model, but independent regressor parameters β_j for each institution j. Once more, we ignore the censored data. By virtue of this, we need not worry about the more complex distribution functions shown above, but we do sacrifice the number

of data points we can learn from for each institution. We now consider our model in terms of the same N data points and M regressors, but we will estimate the covariates' weights according to the institution, of which we have J in total. Thus our model is defined mathematically as

$$\begin{aligned}
y_{ij} &\sim \text{Weibull}(\sigma_j, \alpha), \quad i = 1, \ldots, N,\ j = 1, \ldots, J \\
\alpha &\sim \text{Half-Cauchy}(5), \\
\sigma &\propto \boldsymbol{X}\beta, \text{ via previously seen link function} \\
\beta_{mj} &\sim N(\mu_\beta, \sigma_\beta), \quad m = 1, \ldots, M, \\
\mu_\beta &\sim N(0, 1), \\
\sigma_\beta &\sim \text{Gamma}(1, 1).
\end{aligned}$$

This is seen visually in the directed acyclic graph in Figure 3 below.

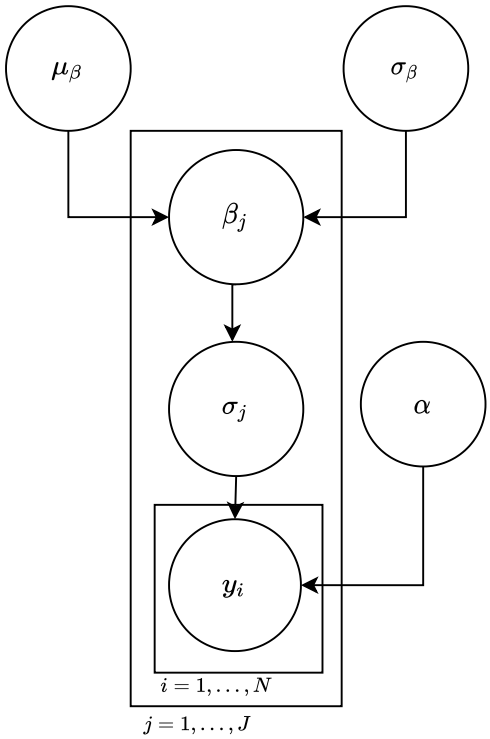


Figure 4: Hierarchical model DAG

2.3.1 Defining the link function

The link function used is the same as that shown in the first model, and is similarly motivated.

2.3.2 Priors

The same global priors are used for the hierarchical model as the pooled model.

We motivate a Gamma hyperprior over the variance of β with Gelman [2006], since we have 7 different groups, a Half-Cauchy will be too sensitive to different group data whereas a Gamma will be more stable for more groups and aid convergence as a result. We have a standard Gaussian hyperprior over the the mean of β, which we can motivate since the value of μ_β can be either positive or negative so this allows the groups to express themselves either way, and this is a standard hyperprior over a mean we assume to have zero mean.

2.3.3 Implemented in Stan

We once more engage in data preparation and run the Stan model, displaying the code, before moving on to the next section where we will discuss the diagnostics of each model.

```
# build dataset from only those non-censored data points
uncensored_data <- cancer %>%
  filter(status == 2) %>%
  filter(inst %in% c(1, 12, 13, 3, 11, 22, 16)) %>%
  drop_na()
# identify covariate labels and build design matrix
cov_labels <- uncensored_data %>%
  dplyr::select(-status, -time, -inst) %>%
  colnames()
# build design matrix
X <- as.matrix(uncensored_data[cov_labels])
# observed survival times
y <- uncensored_data$time
# institution labels
ll <- as.numeric(as.factor(uncensored_data$inst))
# build some hierarchical data for Stan
hier_data = list(
  y = y,
  X = X,
  ll = ll,
  N = length(y),
  M = ncol(X),
  J = length(unique(ll))
)
# compile and run seperate model
whm <- rstan::stan_model(file = "../stan/weibull_hier.stan")
```

```
## Running /anaconda3/envs/seurat4/lib/R/bin/R CMD SHLIB foo.c
## x86_64-apple-darwin13.4.0-clang -I"/anaconda3/envs/seurat4/lib/R/include" -DNDEBUG   -I"/anaconda3/envs
## In file included from <built-in>:1:
## In file included from /anaconda3/envs/seurat4/lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/
## In file included from /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Dense:1:
## In file included from /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Core:88:
## /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unkn
## namespace Eigen {
## ^
## /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error: exp
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /anaconda3/envs/seurat4/lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/
## In file included from /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Dense:1:
## /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file n
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [/anaconda3/envs/seurat4/lib/R/etc/Makeconf:174: foo.o] Error 1
```

```
# print out Stan code
print(whm)
```

```
## S4 class stanmodel 'weibull_hier' coded as follows:
## data {
##   int<lower=0> N;               // number of object
##   int<lower=0> M;               // number of features
##   int<lower =0> J;              // number of institutions
##   vector<lower=0>[N] y;         // target-survival time
##   matrix[N, M] X;               // covariates
##   int<lower=1, upper=J> ll[N];  // instution labels
## }
##
## transformed data {
##   matrix[N, M] Xc;  // centered version of X without an intercept
##   vector[M] means_X;  // column means of X before centering
##
##   // column-center the design matrix for fitting the model
##   for (m in 1:M) {
##     means_X[m] = mean(X[, m]);
##     Xc[, m] = X[, m] - means_X[m];
##   }
## }
##
## parameters {
##   // hyperpriors
##   real mu_beta;
##   real sigma_beta;
##
##   // regressors
##   matrix[M, J] beta;            // regressors weights for different institutions
##   real<lower=0> alpha;          // shape parameter
## }
##
## transformed parameters {
##   vector[N] eta;
##   vector<lower=0>[N] sigma;
##   // compute latent predictor term
##   for (n in 1:N) {
##     eta[n] = Xc[ll[n], ] * beta[, ll[n]];
##   }
##   // apply the log inverse link function
##   sigma = exp(-eta / alpha);
## }
##
## model {
##   // hyperpriors
##   mu_beta ~ normal(0, 1);
##   sigma_beta ~ gamma(1, 1);
##
##   // prior over regressor and shape parameters
##   for (j in 1:J) {
##     beta[, j] ~ normal(mu_beta, sigma_beta);
##   }
```

```
##   alpha ~ cauchy(0, 5);
##
##   // fitting model
##   for (n in 1:N) {
##     y[n] ~ weibull(alpha, sigma[n]);
##   }
## }
##
## generated quantities {
##   // define quantities
##   real ypred[N];
##   vector[N] log_lik;
##
##   // compute quantities
##   for (n in 1:N) {
##     // predictive distribution for survival time
##     ypred[n] = weibull_rng(alpha, sigma[n]);
##     // log-likelihood
##     log_lik[n] = weibull_lpdf(y[n] | alpha, sigma[n]);
##   }
## }
```

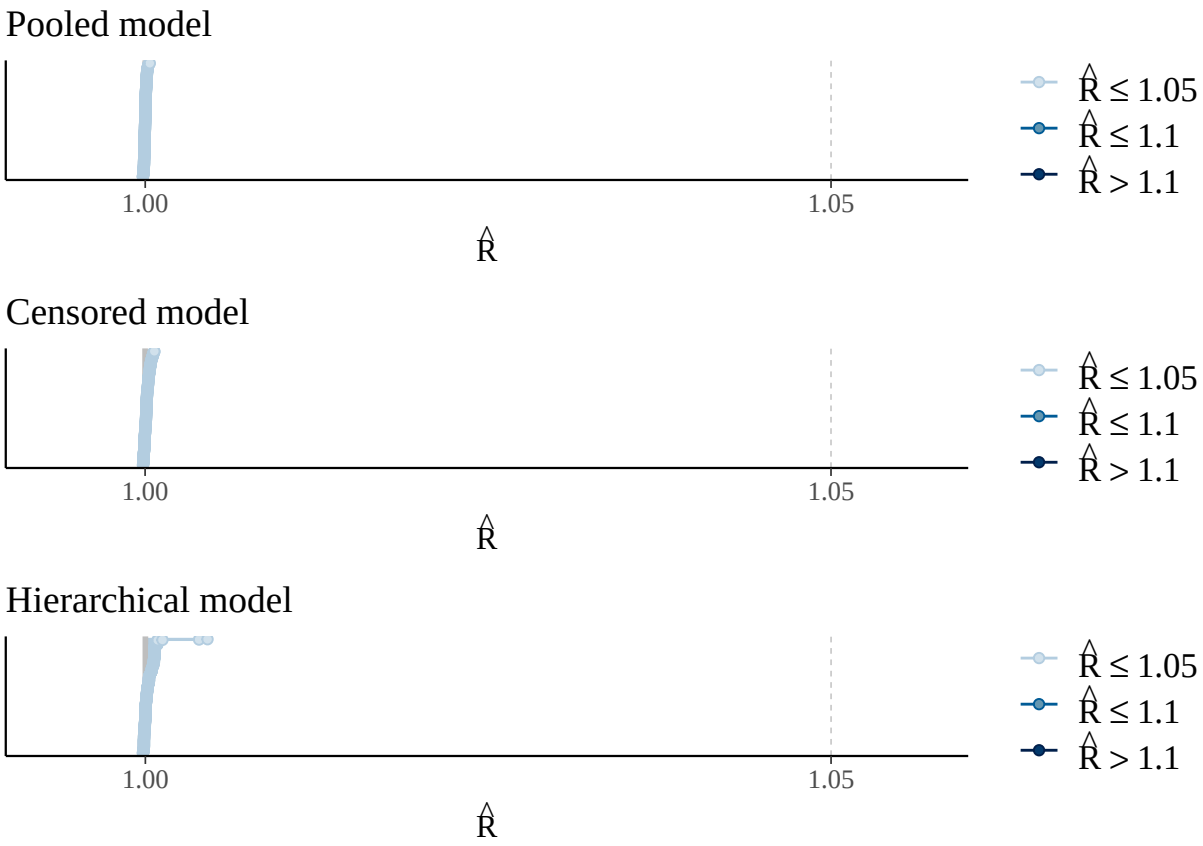
```
# learn the model parameters
weibull_hier <- rstan::sampling(whm, data = hier_data, iter = 10000)
```

3 Diagnostics and performance

3.1 $\hat{R}$ and effective sample size

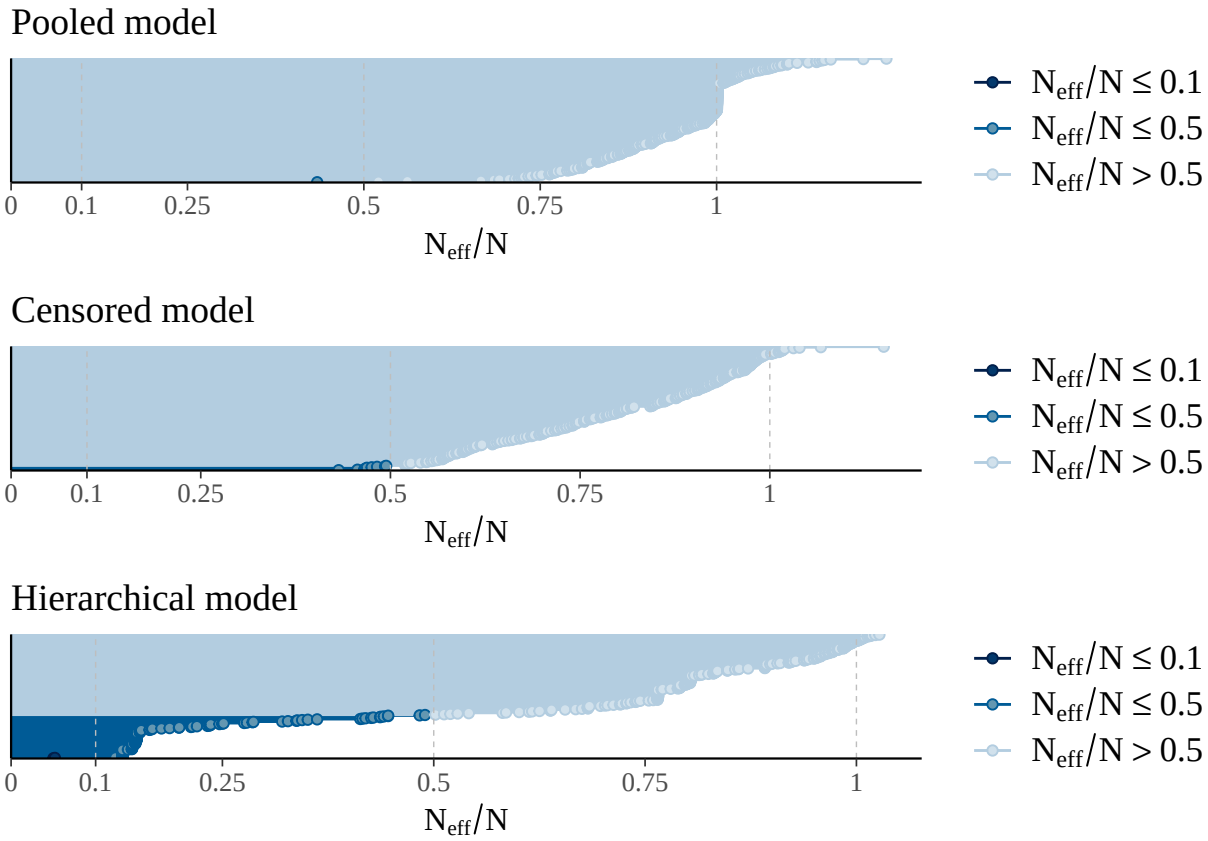
We begin by plotting the rank-normalised $\hat{R}$ values for each of the three models in accordance with Vehtari et al. [2021], which provides an improved comparison of the between-chain and within-chain estimates for each model parameter. If the chains have not mixed well, then we expect this value of $\hat{R}$ to be significantly larger than 1. Contrarily, if we find that the $\hat{R}$ for all model parameters are below 1.05, then we will conclude that the chains have mixed well and agree on the parameter estimates. Below, we plot the $\hat{R}$ values of each parameter for all three models.

```
bayesplot_grid(
  mcmc_rhat(rhat = rhat(weibull_model)),
  mcmc_rhat(rhat = rhat(weibull_cens)),
  mcmc_rhat(rhat = rhat(weibull_hier)),
  titles = c(
    "Pooled model",
    "Censored model",
    "Hierarchical model"
  )
)
```



We find that all our models have converged and that the chains agree on all parameters, which is good news for our parameter estimates. We now look at the effective sample size of our MCMC draws for each our three models. The effective sample size, n_{eff}, is an estimate of the number of independent draws from the posterior distribution the are statistically important towards estimating a given parameter, as is defined in Gelman et al. [2020]. We will plot the ratio of n_{eff} to N, the total number of samples, and hope to find this ratio to be as large as possible, since a larger n_{eff} is indicative of stability across a simulated sequence, and Gelman et al. [2020] argues that this indicates the simulations suffice for practical purposes.

```
bayesplot_grid(
  mcmc_neff(neff_ratio(weibull_model)),
  mcmc_neff(neff_ratio(weibull_cens)),
  mcmc_neff(neff_ratio(weibull_hier)),
  titles = c("Pooled model", "Censored model", "Hierarchical model")
  )
```



We find that both the pooled models have no parameters with a n_{eff}/N ratio of below 0.1, which we consider to be the critical threshold as is given in Gelman et al. [2020], and thus we can assume that the simulations are stable and should suffice for practical purposes. However, the hierarchical model not considering censored data has some parameters with $n_{\text{eff}}/N \leq 0.1$, which is slightly concerning regarding the stability model's parameter estimates, but we nonetheless move on.

3.2 Divergence parameters

As is explained in the Stan documentation, for target distributions whose features are difficult to resolve, the MCMC may miss some samples relating to these features and thus return a biased estimate, which manifests itself as a divergence. There are no divergent transitions in any of our three model, which is identified by `divergent__` being equal to 0 in the below summaries.

Also, chains have a `treedepth__` of at most 10 which is also the default, and a much lower average. Since the maximum number is not exceeded, the sampler does not hit its limit on the number of leapfrog steps taken per iteration. This does not impact the validity of our estimates, but if we had exceeded the maximum `treedepth`, then our sampling may have been inefficient.

```
# pooled model
wm_sampler_params = get_sampler_params(weibull_model, inc_warmup = TRUE)
summary(do.call(rbind, wm_sampler_params), digits = 2)
```

```
##  accept_stat__    stepsize__      treedepth__    n_leapfrog__   divergent__
##  Min.   :0.00   Min.   : 0.000   Min.   : 0.0   Min.   :   1   Min.   :0.000
##  1st Qu.:0.85   1st Qu.: 0.089   1st Qu.: 4.0   1st Qu.:  15   1st Qu.:0.000
##  Median :0.96   Median : 0.143   Median : 5.0   Median :  31   Median :0.000
##  Mean   :0.87   Mean   : 0.121   Mean   : 4.6   Mean   :  43   Mean   :0.044
##  3rd Qu.:0.99   3rd Qu.: 0.163   3rd Qu.: 5.0   3rd Qu.:  31   3rd Qu.:0.000
```

```
##  Max.   :1.00   Max.   :14.386   Max.   :10.0   Max.   :1023   Max.   :1.000
##     energy__
##  Min.   :6.3e+02
##  1st Qu.:6.3e+02
##  Median :6.4e+02
##  Mean   :4.8e+50
##  3rd Qu.:6.4e+02
##  Max.   :2.4e+54
```

```
# censored model
wmc_sampler_params = get_sampler_params(weibull_cens, inc_warmup = TRUE)
summary(do.call(rbind, wmc_sampler_params), digits = 2)
```

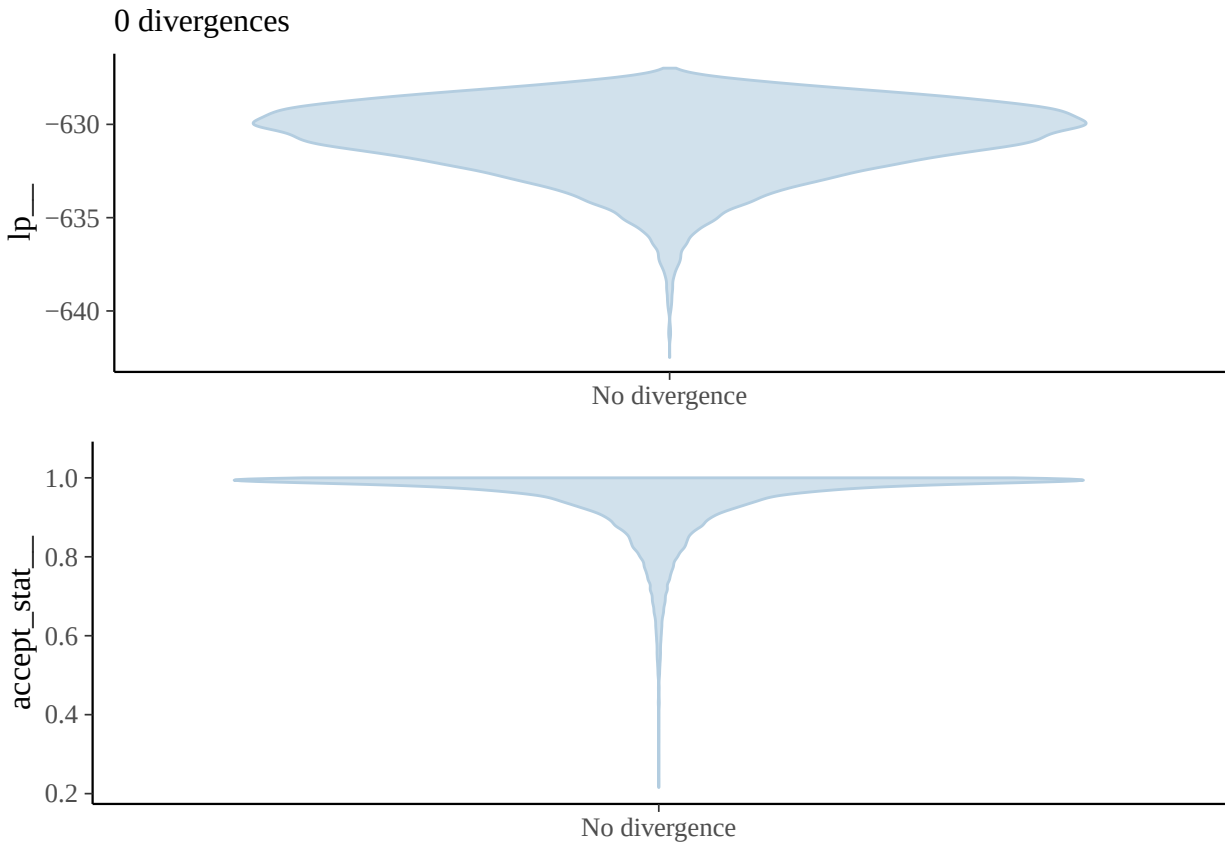
```
##  accept_stat__    stepsize__      treedepth__  n_leapfrog__   divergent__
##  Min.   :0.00   Min.   : 0.000   Min.   : 0   Min.   :   1   Min.   :0.00
##  1st Qu.:0.84   1st Qu.: 0.089   1st Qu.: 5   1st Qu.:  31   1st Qu.:0.00
##  Median :0.95   Median : 0.096   Median : 5   Median :  31   Median :0.00
##  Mean   :0.87   Mean   : 0.094   Mean   : 5   Mean   :  58   Mean   :0.02
##  3rd Qu.:0.99   3rd Qu.: 0.106   3rd Qu.: 5   3rd Qu.:  63   3rd Qu.:0.00
##  Max.   :1.00   Max.   :14.386   Max.   :10   Max.   :1023   Max.   :1.00
##     energy__
##  Min.   : 5.2e+02
##  1st Qu.: 5.2e+02
##  Median : 5.2e+02
##  Mean   :3.7e+145
##  3rd Qu.: 5.2e+02
##  Max.   :3.6e+148
```

```
# hierarchical model
hwm_sampler_params = get_sampler_params(weibull_hier, inc_warmup = TRUE)
summary(do.call(rbind, hwm_sampler_params), digits = 2)
```

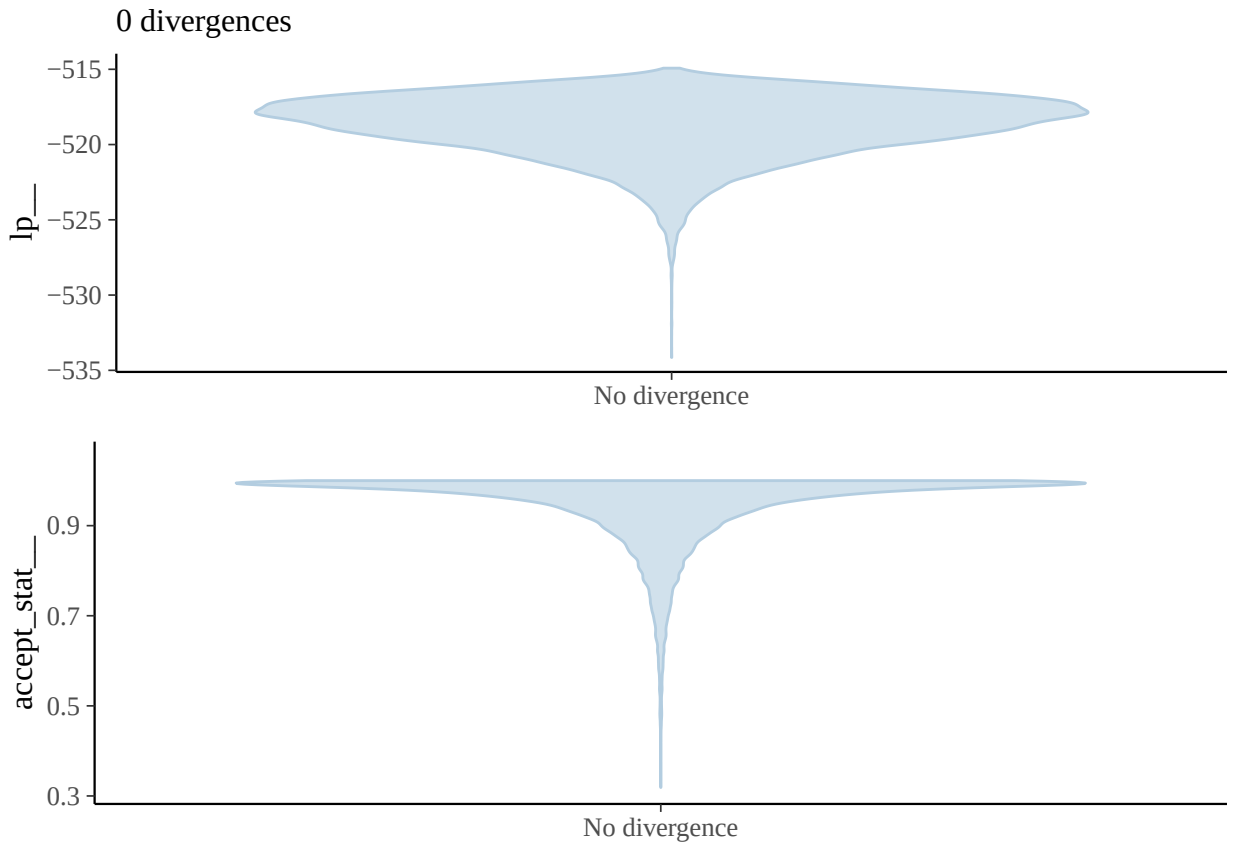
```
##  accept_stat__    stepsize__       treedepth__    n_leapfrog__   divergent__
##  Min.   :0.00   Min.   : 0.0000   Min.   : 0.0   Min.   :   1   Min.   :0.00
##  1st Qu.:0.87   1st Qu.: 0.0042   1st Qu.: 6.0   1st Qu.:  63   1st Qu.:0.00
##  Median :0.96   Median : 0.0134   Median : 6.0   Median :  63   Median :0.00
##  Mean   :0.87   Mean   : 0.0272   Mean   : 6.5   Mean   : 221   Mean   :0.04
##  3rd Qu.:0.99   3rd Qu.: 0.0469   3rd Qu.: 7.0   3rd Qu.: 127   3rd Qu.:0.00
##  Max.   :1.00   Max.   :14.3855   Max.   :10.0   Max.   :1023   Max.   :1.00
##     energy__
##  Min.   : 3.4e+02
##  1st Qu.: 3.8e+02
##  Median : 4.0e+02
##  Mean   :5.4e+190
##  3rd Qu.: 4.2e+02
##  Max.   :5.4e+193
```

We can plot the log-posterior and acceptance NUTS acceptance statistics given divergent samples for all three of our models below.

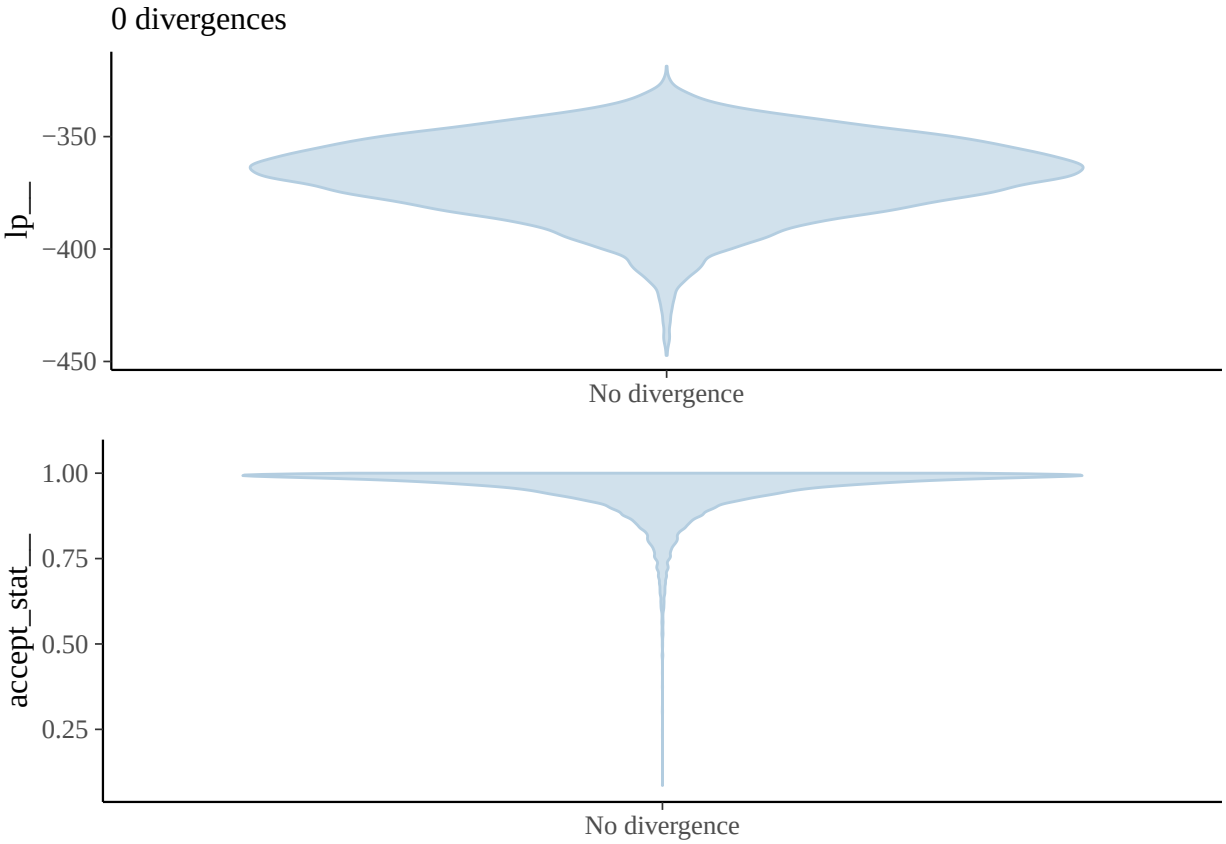
```
# plot divergence statistics for the pooled model
bayesplot::mcmc_nuts_divergence(nuts_params(weibull_model), log_posterior(weibull_model))
```




```
# plot divergence statistics for the censored model
bayesplot::mcmc_nuts_divergence(nuts_params(weibull_cens), log_posterior(weibull_cens))
```




```
# plot divergence statistics for the hierarhical model
bayesplot::mcmc_nuts_divergence(nuts_params(weibull_hier), log_posterior(weibull_hier))
```

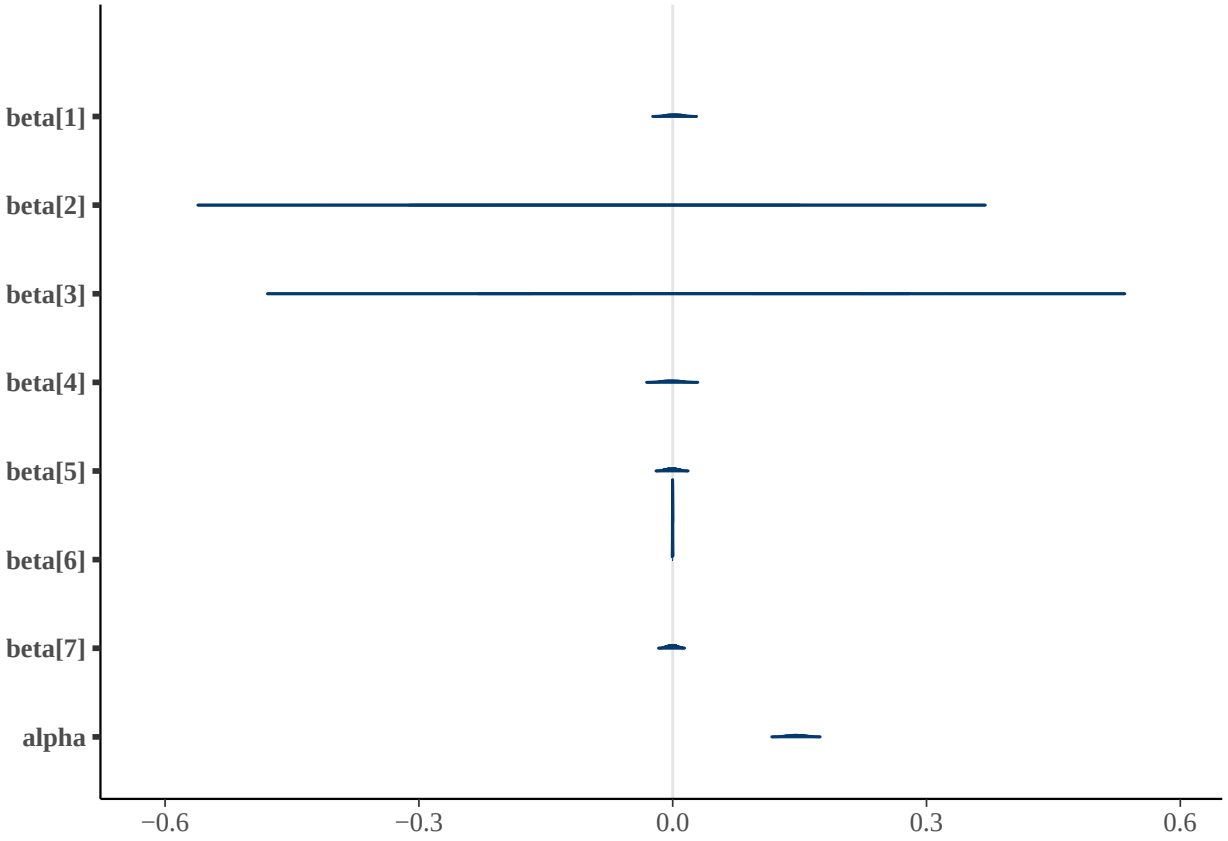


We find that there are no divergences following the warmup, which means our posterior estimates in our target distribution less likely to be biased, and the acceptance statistics are skewed quite high, suggesting our sampling is quite efficient. All of these are reassuring discoveries with regards to the utility of our models.

3.3 Intervals for posterior data

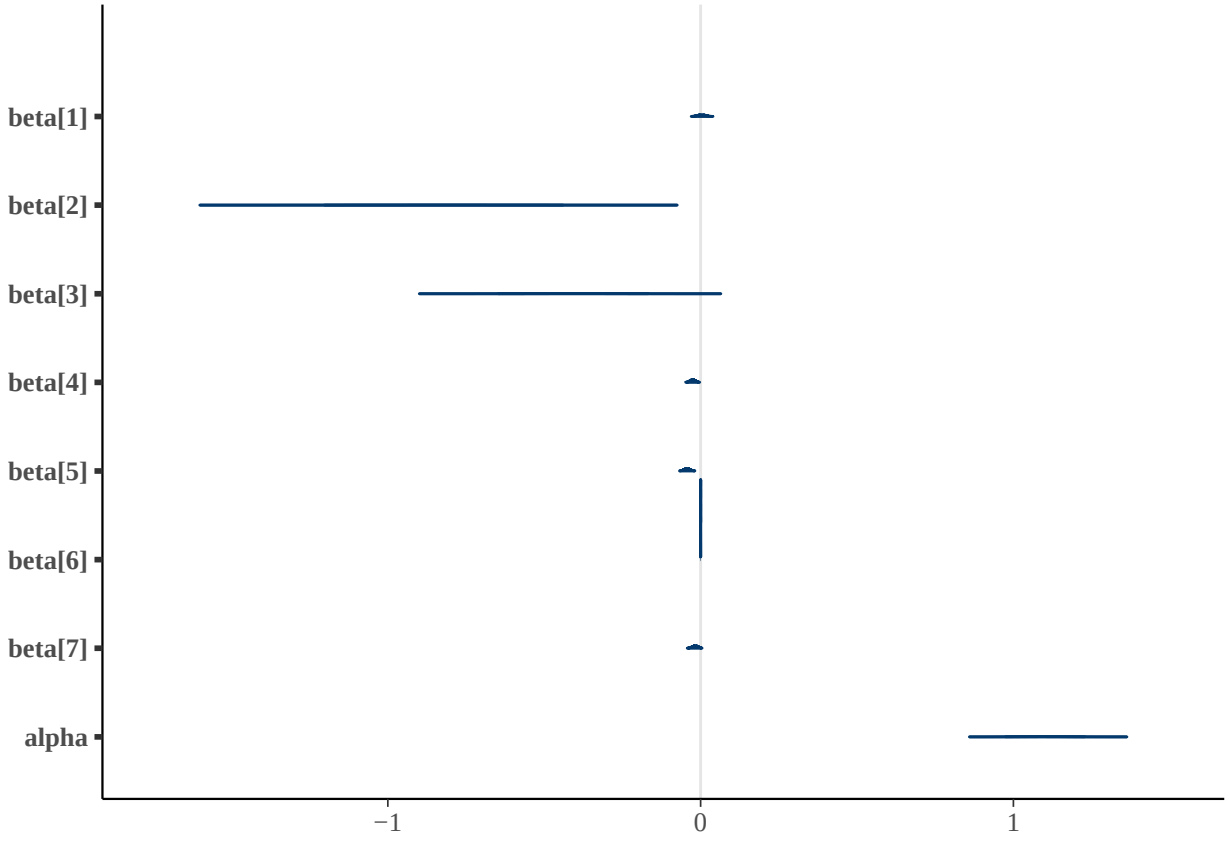
We will now plot the posterior distributions of the weights of our regressors β in our GLMs and the shape parameter α for our three models, starting with our pooled GLM not considering censored data.

```
weibull_posterior <- as.data.frame(weibull_model)
mcmc_areas(
  weibull_posterior,
  pars = colnames(weibull_posterior)[
    (colnames(weibull_posterior) %like% "beta")
    | (colnames(weibull_posterior) %like% "alpha")
  ],
  prob = 0.8,
  # 80% intervals
  prob_outer = 0.99,
  # 99%
  point_est = "mean"
)
```



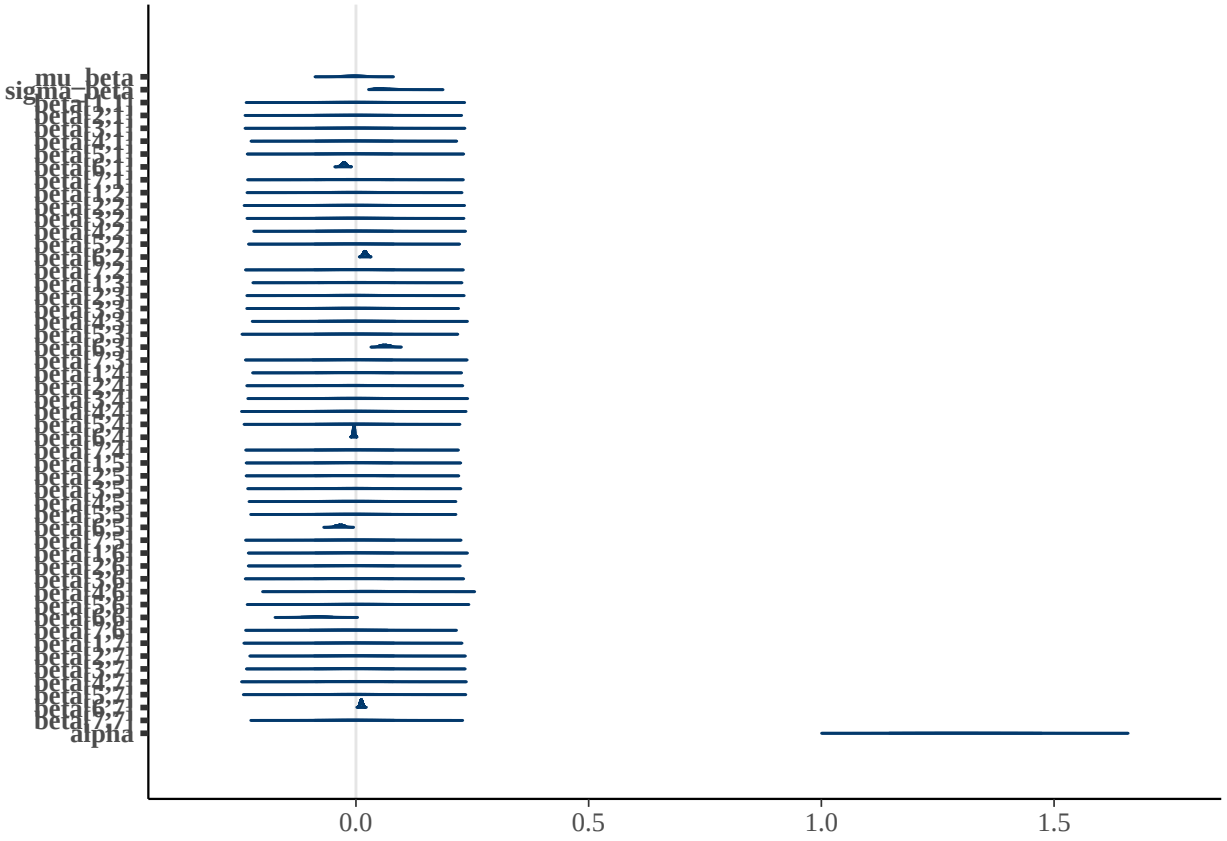
These posterior distributions of the β parameters in this model are very clearly centered around the origin, and in some cases, may not be hugely informative in the linear model. The α parameter seems to have well converged, and to a non-zero value which is a good sign. We move now to the model considering censored data.

```
cens_weibull_posterior <- as.data.frame(weibull_cens)
mcmc_areas(
  cens_weibull_posterior,
  pars = colnames(cens_weibull_posterior)[
    (colnames(cens_weibull_posterior) %like% "beta")
    | (colnames(cens_weibull_posterior) %like% "alpha")
  ],
  prob = 0.8,
  prob_outer = 0.99,
  point_est = "mean"
)
```



Here we find more informative parameter estimates in terms of effect on the latent predictor than the uncensored data. For hierarchical model the results are shown down below.

```
hier_posterior <- as.data.frame(weibull_hier)
mcmc_areas(
  hier_posterior,
  pars = colnames(hier_posterior)[
    (colnames(hier_posterior) %like% "beta")
    | (colnames(hier_posterior) %like% "alpha")
  ],
  prob = 0.8,
  prob_outer = 0.99,
  point_est = "mean"
  )
```

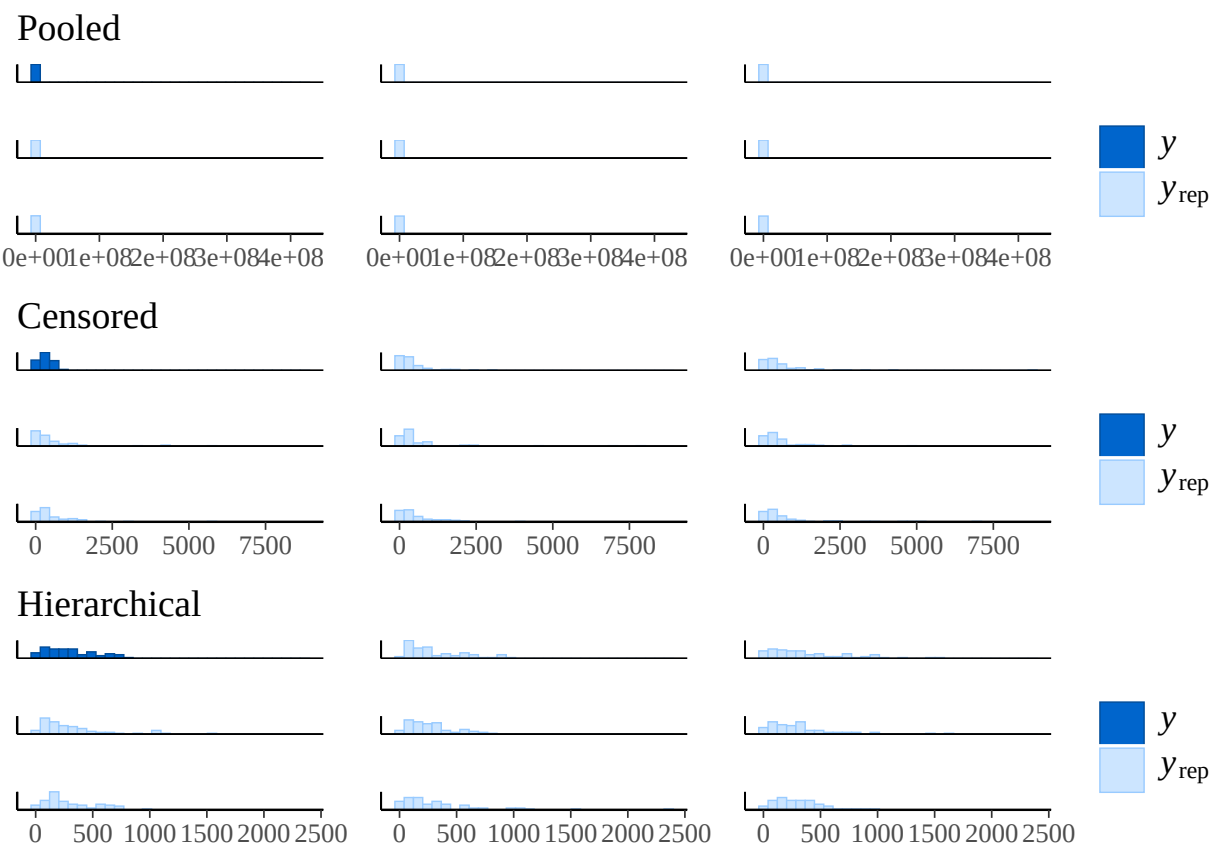


Once more we find a lot of regressor parameters including zero in their 80% credible interval, while some parameters are more significant. We also find that the signficant parmeters differ slightly between institutions, suggesting that the hazard does change as a function of which institution a patient has been admitted to. Causality, of course, can not be concluded however, and the statistical significance of these differences was not explicitly tested.

3.4 Posterior predictive checks

```
color_scheme_set("brightblue")
yrep_cens=extract(weibull_cens)$ypred
yrep_weib=extract(weibull_model)$ypred
yrep_hier=extract(weibull_hier)$ypred
bayesplot::bayesplot_grid(
  bayesplot::ppc_hist(y, yrep_weib[1:8, ]),
  bayesplot::ppc_hist(yobs, yrep_cens[1:8, ]),
  bayesplot::ppc_hist(y, yrep_hier[1:8, ]),
  titles = c("Pooled", "Censored", "Hierarchical")
)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



For the pooled model the posterior results are clearly similar to original survival survival distribution. Here, the censored data is not includded, so all the data is mainly concentrated near a particular peak. For the censored model the posterior draws are again similar to the original distribution. However, for some replicates the peak of the distribution is a little bit lower than for original data. For the hierarchical model there are more broad results, as here different institutional groups are taken into consideration. The predictive values in all cases follow the same distribution with of course some noise.

All in all, the posterior predictive distributions from our three models seem to match the original data, which suggests that the models are reasonable in their desgin and have produced good predictions. Importantly, it increases our confidence in their predictive performance which we presently move on to discuss.

3.5 Model comparison using leave-one-out cross-validation and WAIC

Having briefly examined the posterior distributions of our models, we now investigate the PSIS-LOO and WAIC values for model comparison, in the aim of determining which of our models is "best" for our purposes. This will be the model with the highest ELPD value. Note, that PSIS-LOO is more accurate than WAIC criteria in our case since PSIS provides useful diagnostics as well as effective sample size and Monte Carlo estimates, as is mentioned in Vehtari et al. [2016].

The PSIS-LOO and WAIC values are computed in accordance with the vignettes from Vehtari et al. [2016] below.

```
# perform approximate loo and psis-loo
wm_log_lik <- extract_log_lik(weibull_model, merge_chains = FALSE)
# estimate the PSIS effective sample size
wm_r_eff <- relative_eff(exp(wm_log_lik), cores = parallel::detectCores())
# compute loo
wm_loo <- loo(wm_log_lik, r_eff = wm_r_eff, cores = parallel::detectCores())
# compute waic
```

```
wm_waic <- waic(wm_log_lik, cores = parallel::detectCores())
# repeat for censored data model
cwm_log_lik <- extract_log_lik(weibull_cens, merge_chains = FALSE)
cwm_r_eff <- relative_eff(exp(cwm_log_lik), cores = parallel::detectCores())
cwm_loo <- loo(cwm_log_lik, r_eff = cwm_r_eff, cores = parallel::detectCores())
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

```
cwm_waic <- waic(cwm_log_lik, cores = parallel::detectCores())
```

```
## Warning:
## 4 (3.8%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
# repeat for hierarchical Weibull
hwm_log_lik <- extract_log_lik(weibull_hier, merge_chains = FALSE)
hwm_r_eff <- relative_eff(exp(hwm_log_lik), cores = parallel::detectCores())
hwm_loo <- loo(hwm_log_lik, r_eff = hwm_r_eff, cores = parallel::detectCores())
```

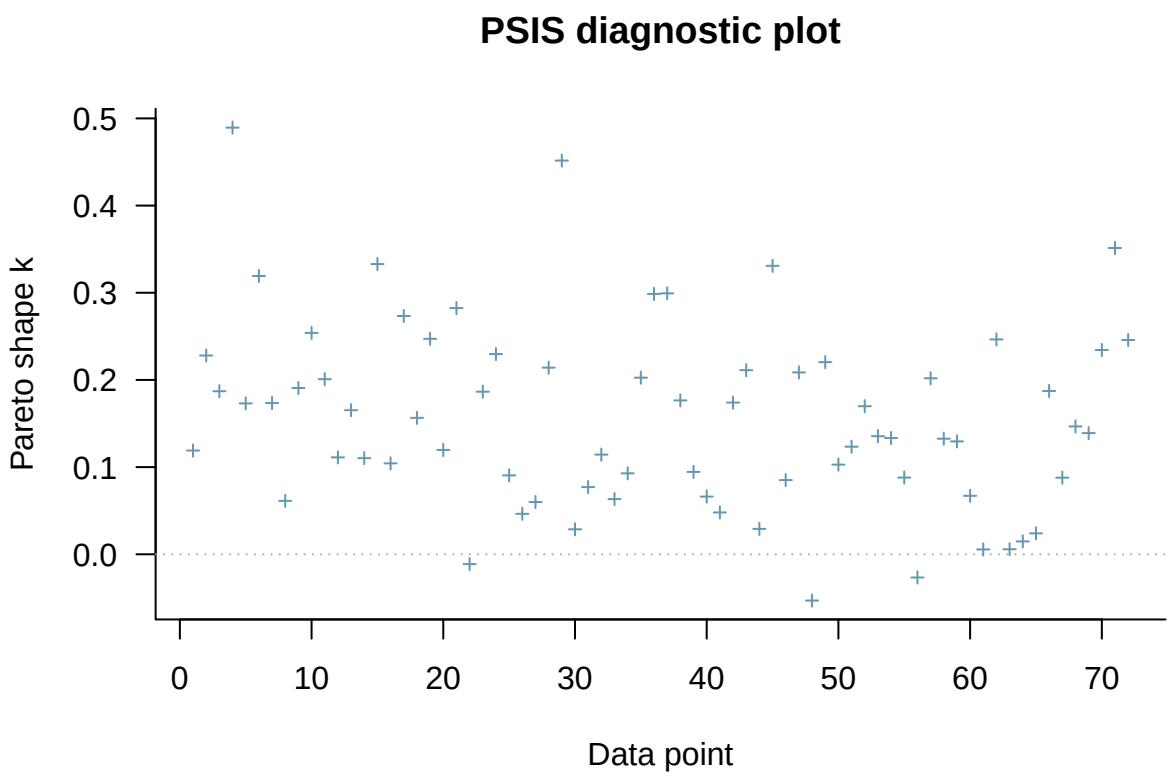
```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for details
```

```
hwm_waic <- waic(hwm_log_lik, cores = parallel::detectCores())
```

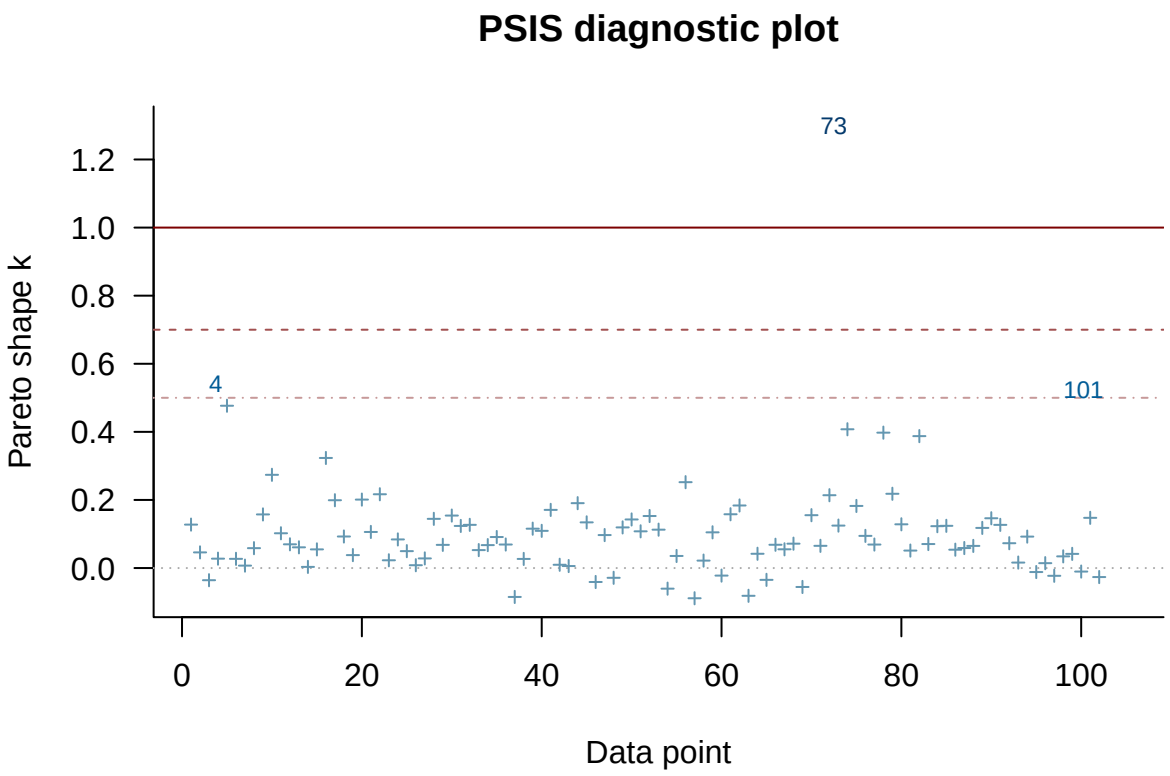
```
## Warning:
## 2 (2.8%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

We now plot the Pareto k values for all data points from our three models below.

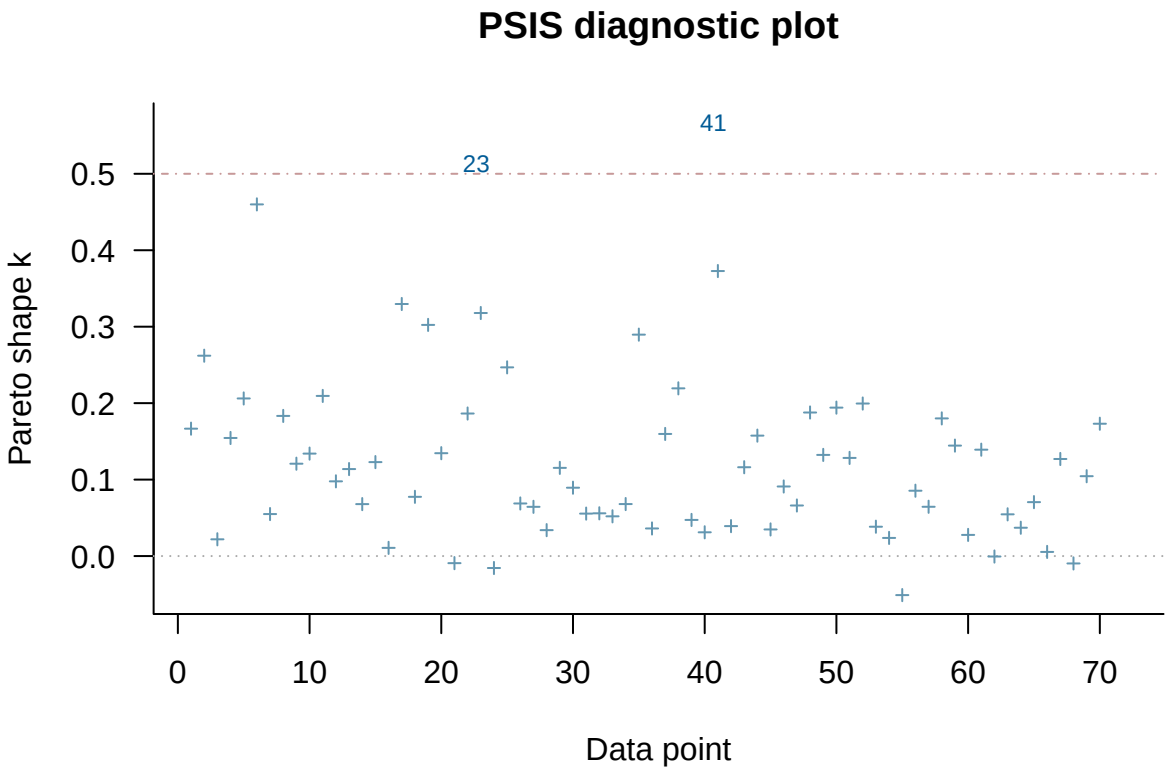
```
# plot pareto k diagnostics for the models
plot(wm_loo, label_points = TRUE)
```




```
plot(cwm_loo, label_points = TRUE)
```




```
plot(hwm_loo, label_points = TRUE)
```



We find once more than no data points have $\hat{k} > 0.7$ in either the pooled model without censored data or the hierarchical model, and thus we are able to conclude that the PSIS-LOO estimates are reliable. Some data points have $\hat{k} > 0.5$, which harms our confidence in the estimates slightly, but is still considered "ok". There is one data

point with $\hat{k} > 0.7$ in the pooled model considering censored data, suggesting that there is potential bias in our model, and it might be over-estimating the predictive accuracy of the model. However, one high value is not enough for us to completely discard the model.

Having established that our Pareto $\hat{k}$ values of sufficiently good to consider the PSIS-LOO estimates to be reliable, we move on to display them.

```
# compare loos
wm_loo
```

```
##
## Computed from 20000 by 72 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -631.9 10.0
## p_loo         5.4  0.5
## looic      1263.8 20.0
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
cwm_loo
```

```
##
## Computed from 20000 by 105 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -527.0 31.0
## p_loo        12.9  5.8
## looic      1053.9 62.0
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     102   97.1%   3215
##  (0.5, 0.7]   (ok)         2    1.9%   2559
##    (0.7, 1]   (bad)        0    0.0%   <NA>
##    (1, Inf)   (very bad)   1    1.0%   13
## See help('pareto-k-diagnostic') for details.
```

```
hwm_loo
```

```
##
## Computed from 20000 by 72 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -485.6  6.1
## p_loo         6.5  1.1
## looic       971.3 12.1
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
```

```
## (-Inf, 0.5]   (good)     70    97.2%   2727
##  (0.5, 0.7]   (ok)        2     2.8%   2606
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

Comparing the ELPD, the best model (with the highest ELPD) is the hierarchical model, followed by the pooled model with censored data, and finally the pooled model without censored data. It is worth noting, however, that the censored model was trained on more data since it included censored patients. This means that in theory it should have learned more information than the other two, and is not necessarily comparable. However, since we find a better ELPD for the hierarchical model even given the difference in training data, we are comfortable concluding that it is the best of the three.

Considering the `p_loo` values. For the pooled model, the effictive number of parameters is ~5.5 and given that there 7 in the model, this suggests that our parameters are mostly significant. For the censored pooled model ~13 effective parameters are shown and we actually have 15 parameters, which is again demostrative of significant parameters. For the hierarchical model, the `p_loo` shows 6.6 parameters which is much fewer than were trained, and so we understand that most of them are not significant. In this sense, the censored model on pooled data has found the most effective parameters.

We show also the WAIC values below, and achieve the same conclusion although understanding that our LOO values are more accurate.

```
# compare waics
wm_waic
```

```
##
## Computed from 20000 by 72 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic   -631.7 10.0
## p_waic         5.3  0.4
## waic        1263.5 20.0
```

```
cwm_waic
```

```
##
## Computed from 20000 by 105 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic   -525.6 30.8
## p_waic        11.5  4.6
## waic        1051.2 61.6
##
## 4 (3.8%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
hwm_waic
```

```
##
## Computed from 20000 by 72 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic   -485.4  6.0
## p_waic         6.3  1.0
## waic         970.8 12.0
##
```

```
## 2 (2.8%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

3.6 Predictive performance assessment

We will use ELPD with leave-one-out cross validation as is put forward in Vehtari et al. [2016] to measure predictive performance on the basis that it is better than, for example, MSE for continuous variable prediction, as it evaluates the whole predictive distribution and not just the mean. This is also the chosen metric is other similar literature.

For the pooled model not using censored data, ELPD $= -632$, for the pooled model using censored data, ELPD $= -527$, and for the hierarchical model, ELPD $= -486$. Knowing that all Pareto $\hat{k}$ values are reasonable, and that thus these values are reliable, we choose the hierarchical model as the best fitting model for our data.

3.7 Prior sensitivity analysis

The sensitivity of the posterior distribution of our sampled parameters to the proposed prior distribution was checked for all three models with different distribution parameters, and in some cases different distributions.

The dataset is not very small and posterior inferences are based on large numbers of MCMC iterations which do not tend to be particularly sensitive to the prior distribution. The parameters of half-Cauchy were changed a bit, as well as the Normal distribution for betas. The combinations tested were: Cauchy$(0, 8)$ and Normal$(0, 10)$, Cauchy$(0, 2)$ and Normal$(0, 10)$, Cauchy$(0, 5)$ and Normal$(0, 20)$, Cauchy$(0, 8)$ and Normal$(0, 20)$. The results in most cases remained the same.

In order to test the motivation set out in previous sections and in Gelman [2006] regarding the prior over the hyperparameter σ_β in the hierarchical model, its hyperprior was changed to a Half-Cauchy (the suggested priors for fewer hierarchical groups in), and in this case convergence was more difficult. As such, the rigidity and relative strength of the Gamma$(1, 1)$ helps convergence in this case with more groups and fewer data points for each group. It is also worth noting, however, that changing the prior over the global α parameter from Half-Cauchy to Gamma has a large impact on the final result, and ultimately did not result in convergence. We can draw from this information that the relative flexibility afforded by the weakly-informative Half-Cauchy prior allows the data to express itself more compared to the stronger and less flexible Gamma prior.

4 Conclusion

4.1 Issues and improvements

The target distribution in the data is quite skewed, making it is rather difficult to learn. In this work we chose a Weibull distribution to model it, although it would be interesting to investigate the performance of Exponential (which has properties similar to the Gamma distribution, but a survival function like a Weibull), or a Cox regression.

Another issue was the relatively low n_{eff} of the hierarchical model, and one critically low n_{eff}/N statistic. In certain situations, the group-level parameters do not constrain the hierarchical distribution closely enough. This can occur when we either have many groups or high variance between the groups. In order to make hierarchical model sampling more efficient and to improve effective sample size metrics, we could employ a so-called *non-centered parameterisation*, where we replace the parameterisation of

```
parameters {
  // GLM parameters
  matrix[M, J] beta;            // regressors weights for different institutions
  real<lower=0> alpha;          // shape parameter
  ...
}
model {
```

```
  // hyperpriors
  mu_beta ~ std_normal();
  sigma_beta ~ gamma(1, 1);
  // prior over regressor and shape parameters
  for (j in 1:J) {
    beta[j] ~ normal(mu_beta, sigma_beta);
  }
  ...
```

to

```
parameters {
  // hyperparameters
  real mu_beta;
  real<lower=0> sigma_beta;
  // GLM parameters
  matrix[M, J] beta_unif;       // non-centered parameterisation regressors
  real<lower=0> alpha;          // shape parameter
  ...
}
transformed parameters {
  vector[M] beta[J];            // regressors weights for different institutions
  // beta ~ normal(mu_beta, sigma_beta)
  for (j in 1:J) {
    beta[j] = mu_beta + sigma_beta * beta_unif[j];
  }
  ...
}
model {
  // hyperpriors
  mu_beta ~ std_normal();
  sigma_beta ~ gamma(1, 1);
  // prior over regressor and shape parameters (non-centered parameterisation)
  for (j in 1:J) {
    beta_unif[j] ~ std_normal();
  }
  ...
```

so that our `beta`, `mu_beta` and `sigma_beta` are less correlated with our posterior, and thus increasing the effective sample size, as is shown in Betancourt and Girolami [2013]. This reparameterisation, however, will likely not significantly improve n_{eff} of the hierarchical model, as we have neither too many groups nor too much data. A Stan implementation is available at McLatchie and Odnoblyudova [2021] for further interest.

Some warnings were returned relating to low Bayesian Fraction of Missing Information estimates. Betancourt and Girolami [2013] suggests that this is indicative of an improper adaptation phase in the Markov Chains, and thus results in an inefficient exploration of the posterior distribution. Much like the treedepth statistic, this is an efficiency concern as opposed to a validity concern. Increasing the number of warmup steps of MCMC iterations might help combat this.

Also, maybe it might be worth paying more attention to features and applying extra transformation (as well as mean-centering) or even extraction to them. We could also try repeating the analysis without mean-centering the binary variables in the data to see if this would improve the significance of the respective regressors. Moreover, the priors can be tested even more accurately, and more exotic distributions could be tried.

The sampling algorithms needs rather large number of iteration to converge, so perhaps changing the architecture of the model (link function, feature selection, etc.) could also be considered in future.

4.2 Things learnt from the data analysis

It was understood that the survival time of patients with advanced lung cancer can be modeled well with a Weibull GLM. Further, we concluded that understanding which institution an individual has been admitted to impacts their survival time, and incorporating this into our model aided predictive performance. We ultimately found this hierarchical model to be the best of the three presented in the report. This was confirmed by ELPD and WAIC parameters, and validated by the Pareto $\hat{k}$ diagnostic plot.

4.3 Self-reflection and learnings

The group learnt a lot about the applications of MCMC methods to survival analysis, domain specific notation, and vocabulary. The group were also afforded the opportunity to dive deep into more exotic distributions such as the Weibull and Cauchy distributions, and to apply these in GLMs. The process of prior ellicitation and model building, including motivating a link function in a non-standard GLM, was very informative. Last but not least, some functions and packages in `R` connected with bayesian statistics were discovered, and used effectively.

References

Mohammed H Abujarad and Athar Khan. Exponential model: A bayesian study with stan. *International Journal of Scientific Research*, 09 2018. doi: 10.24327/ijrsr.2018.0908.2470.

M. J. Betancourt and Mark Girolami. Hamiltonian monte carlo for hierarchical models, 2013.

Andrew Gelman. Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis*, 1, 09 2006. doi: 10.1214/06-BA117A.

Andrew Gelman, John Carlin, Hal Stern, David Dunson, Aki Vehtari, and Donald Rubin. *Bayesian Data Analysis*. 2020. URL http://www.stat.columbia.edu/~gelman/book/.

Mukesh Kumar and Prashant Sonker. Parametric survival analysis using r: Illustration with lung cancer data. *CANCER REPORTS*, 3, 08 2020. doi: 10.1002/cnr2.1210.

Charles Loprinzi, Jorge Laurie, H Wieand, J Krook, Paul Novotny, J Kugler, J Bartel, M Law, M Bateman, and N Klatt. Prospective evaluation of prognostic variables from patient-completed questionnaires. north central cancer treatment group. *Journal of Clinical Oncology*, 12:601–607, 03 1994. doi: 10.1200/JCO.1994.12.3.601.

Yann McLatchie and Arina Odnoblyudova. Bda project. https://github.com/yannmclatchie/bda-project, 2021.

Terry M Therneau. *A Package for Survival Analysis in R*, 2021. URL https://CRAN.R-project.org/package=survival. R package version 3.2-13.

Aki Vehtari, Andrew Gelman, and Jonah Gabry. Practical bayesian model evaluation using leave-one-out cross-validation and waic. *Statistics and Computing*, 27(5):1413–1432, Aug 2016. ISSN 1573-1375. doi: 10.1007/s11222-016-9696-4. URL http://dx.doi.org/10.1007/s11222-016-9696-4.

Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. Rank-normalization, folding, and localization: An improved r$\widehat{}$ for assessing convergence of mcmc (with discussion). *Bayesian Analysis*, 16(2), Jun 2021. ISSN 1936-0975. doi: 10.1214/20-ba1221. URL http://dx.doi.org/10.1214/20-BA1221.