Lung Cancer Survival Prediction with Bayesian Generalised Linear Models

Yann McLatchie, Arina Odnoblyudova

Contents

1 Introduction

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

Survival analysis is the combination of different statistical methods for analysing time-to-event data. Exploring survival models can be challenging, since different models from non-parametric to parametric can be used, and various distributions like exponential, Weibull, and log-normal are all applicable for each concrete case. The most common model is the Cox hazard model, however, it is too simple for this case, and proposes a constant effect of predictor variables on survival duration throughout time which we know to be untrue.

This study employs a Bayesian Weibull model of survival time data of patients with advanced lung cancer. The Weibull approach is more flexible than Cox, and the hazard rate is not assumed to be constant through time. For simulation Bayesian inference with MCMC is used, providing us with satisfying approximation of uncertainty and ability to use priors as domain knowledge. The model is implemented and tested with the help of R and Stan package.

We will investigate the effecicacy of three models: a pooled model across insitutions, trained on only observed data, a pooled model using censored data, and a hierarchical model not using censored data. The respective models will eventually be compared with regards to their predictive power.

The code with model implementation is provided in McLatchie and Odnoblyudova [2021].

1.1 Data description

1.1.1 General description

The data used in the study shows survival 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 aim to predict the survival time of patients with lung cancer.

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

Special attention has to be paid for "censored status" feature. It indicates if the patient had some terminal event (in which case it is equal to 1) or not (equal to 0). If patient is censored, true survival time for him is not known. 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:

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

It is better to provide some descriptive statistics to familiarize with data.

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

There were 44 observations with missed values, which have been removed from dataset. Now it consists of 105 rows.

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

Institutions are considered as variables, useful for 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 datapoints for all models.

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

```
##
##  1  3 11 12 13 16 22
## 28 12 13 16 13 10 13
```

Moving to categorical variables (fig.1), the number of men prevails over women. The majority of patients are ambulatory with symptoms.

```r
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"))
```

The distribution for continuous variables is shown in fig. 2. The features do not follow normal distribution.

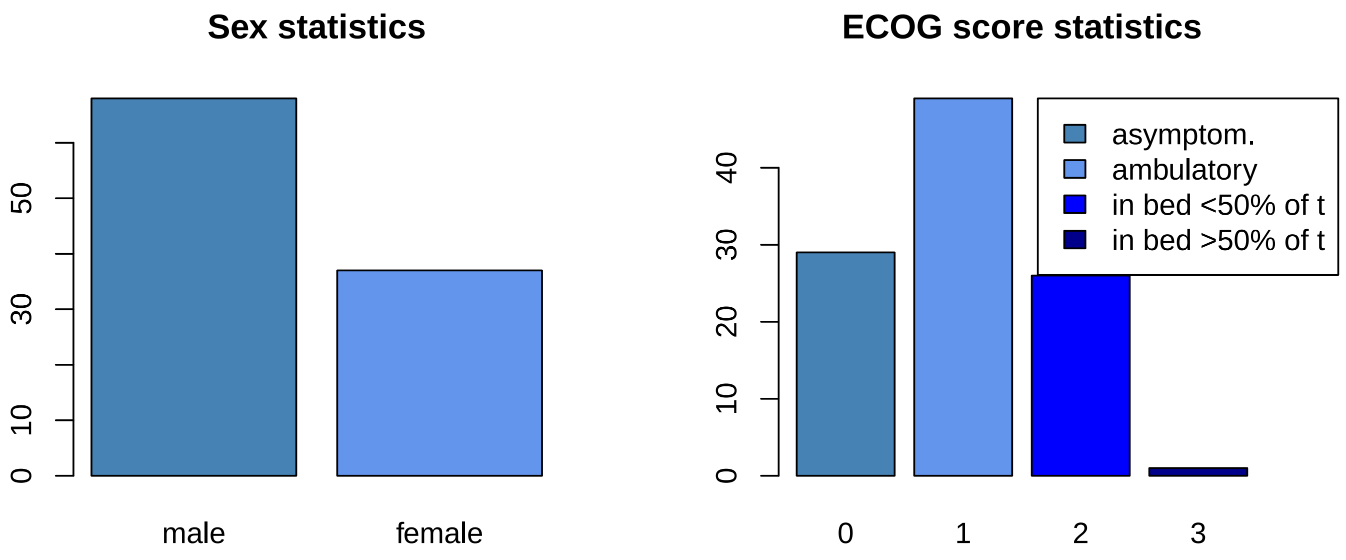


Figure 1: Categorical variables

```
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 if there is linear correlation between variables. The correlation is not that high, the only one is between ph.karno and pat.karno (0.525), but it is reasonable, as since patient and doctor may measure the same quantity.

```
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
```

1.2 Related studies

The original data was presented in the work Loprinzi et al. [1994] in 1994, it just provided descriptive information from a lung patient-completed questionnaire, which was aggregated into 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 deep description of Weibull model implementation, i.e. more attention was payed to Cox regression. Abujarad and Khan [2018] provided the description of exponential models, applied to lung cancer data analysis with Stan code. The Weibull distribution was just mentioned their as a possibility, but no formulas and conclusions were derived for Weibull model. That is why in our work two Weibull Survival models are built: a hierarchical and a pooled which will be described in the following section. The main approaches and theory of building survival models were used from the studies above, but the stan implementation, data preprocess, and choice of priors was made by the authors.

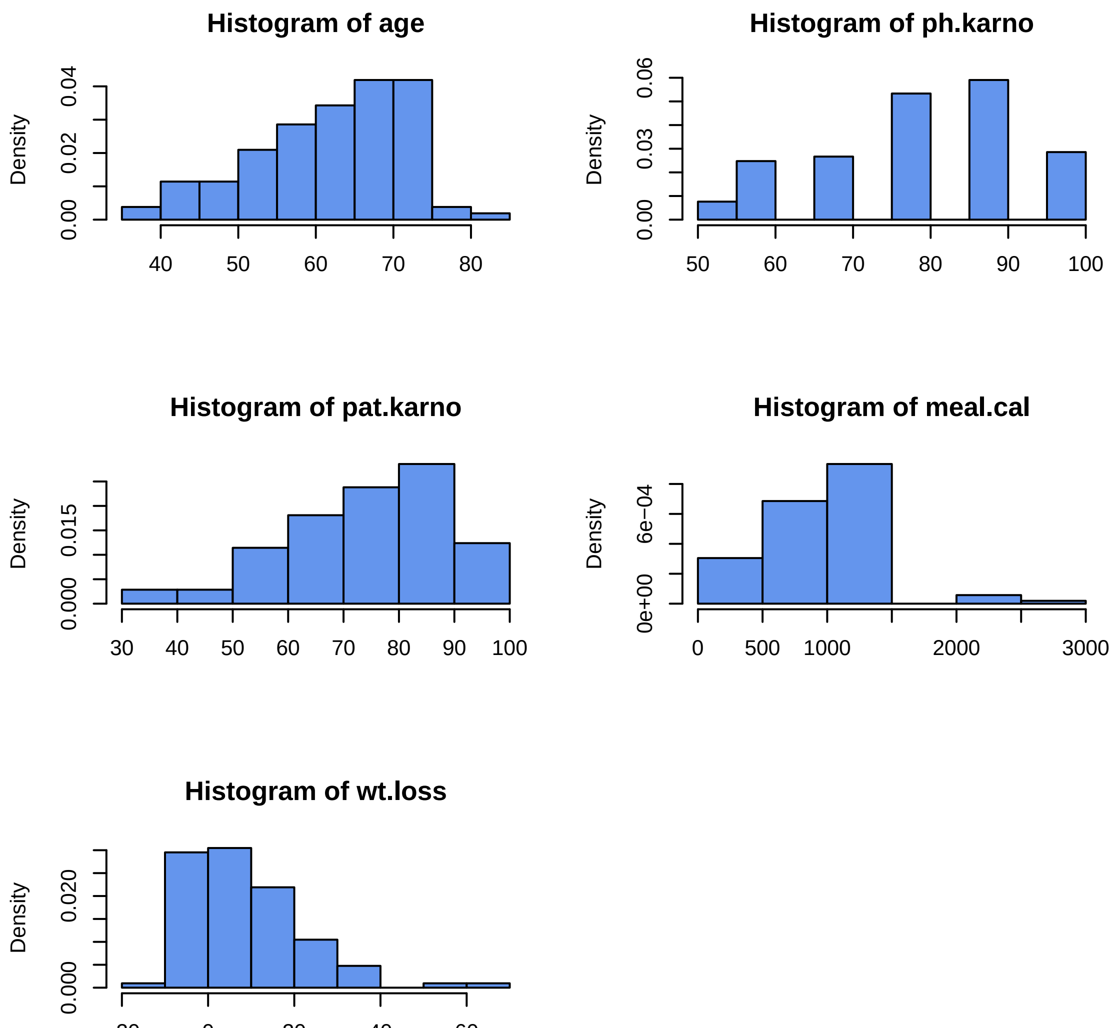


Figure 2: Continuous variables

2 Description of models

In the following section, we will motivate and define mathematically four Generalised Linear Models (GLMs) implemented in Stan and using BRMS in two cases.

```r
# 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

Let $y \sim \text{Weibull}(\alpha, \sigma)$, so that

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

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.

2.1.2 The Weibull distribution as a member of the Exponential family

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

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

For some arbitrary parameter functions $h(\cdot), T(\cdot), A(\cdot)$, and canonical parameter η. Now take α fixed and finite, then it can be shown that this distribution belongs to the exponential family since we can write it's probability density function

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

with

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

2.1.3 Defining the link function

Looking at our sufficient 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 strictly positive. Thus we choose a log link function for our GLM such that

$$\sigma = \exp(-\frac{\boldsymbol{X}\beta}{\alpha}).$$

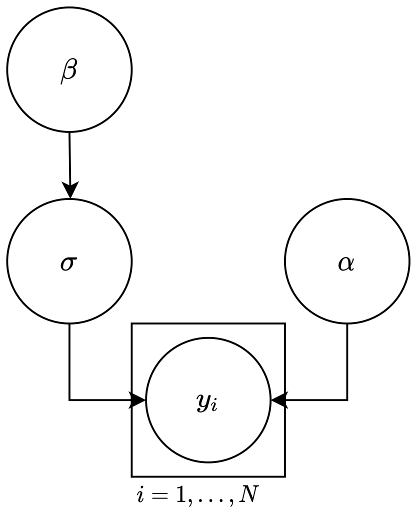
2.1.4 Priors

In our Stan model, we will enforce two priors over each of the regressors in the linear model, and the shape parameter of our resulting 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 inferences are sensitive to the choice of weakly-informative priors. 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. Our Gaussian prior over the regressors is easily motivated by the parameterisation of the data. 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 can be negative, making this an appropriate weakly-informative prior.

In this pooled model, we model these parameters the same across 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 below in the directed acyclic graph below.



2.1.5 Implemented in Stan

Below, we fit the model in Stan and output the Stan code for the reader.

```
# 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]);
##   }
## }
```

```r
# 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

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

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, log-likelihood consists only of 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, 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.

2.2.1 Priors

The same priors are used for the censored model as for the pooled model. Even we are not experts-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 "speak for itself". 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

Data preparation

```r
# 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]
```

Building data list for Stan model

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

Running model

```
# 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: unkno
## namespace Eigen {
## ^
## /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error: expe
## 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 no
## #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));
##   }
## }
```

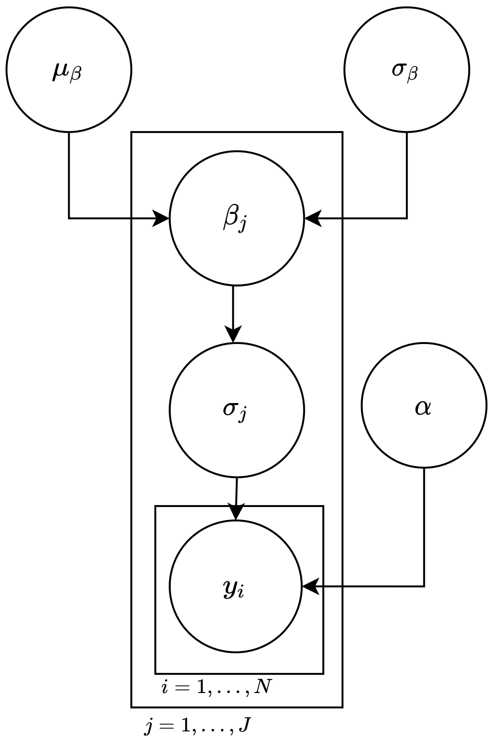
```r
# learn the model with parameters 4 chains, 10000 iterations for each, 5000 iterations for warm-up
weibull_cens = rstan::sampling(cwm, data = data_model, iter = 10000)
```

2.3 Hierarchical Weibull without censored data

Here we implement a model with some global shape parameter to be learned, but independent regressor parameters for each institution. Once more, we ignore the censored data. By virtue of this, we need not worry about complex distribution functions, but 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 our covariate's weight according to the institution, of which we have J in total. Thus our model is defined 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 below in the directed acyclic graph below.



2.3.1 Defining the link function

2.3.2 Priors

The same global priors are used for the hierarchical model as the pooled model. We motivate the Gamma hyperprior over the variance of β with Gelman [2006], since we have 7 groups a Half-Cauchy will be too flexible to different group whereas a Gamma will be more stable for more groups. We have a standard Gaussian hyperprior over the the mean of β, which we can motivate since the value of β can be either positive or negative so this allows the groups to express themselves either way.

2.3.3 Implemented in Stan

```
# 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]);
##   }
## }
```

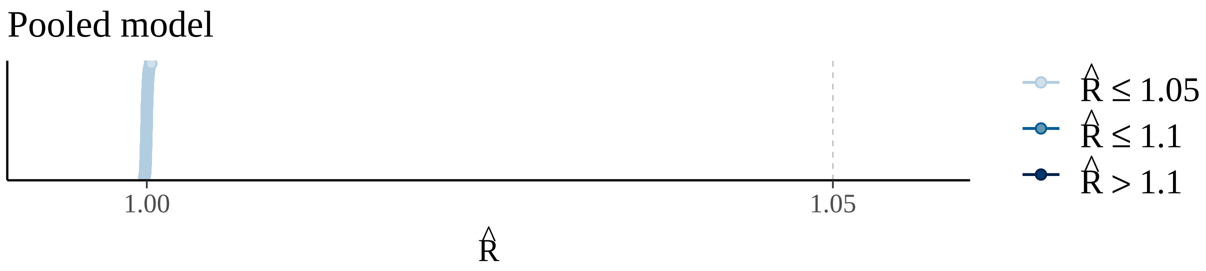
```r
# learn the model parameters
weibull_hier <- rstan::sampling(whm, data = hier_data, iter = 10000)
```

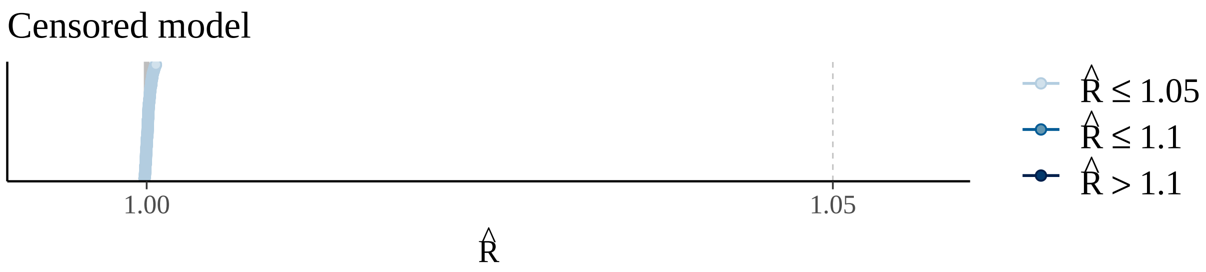
3 Diagnostics and performance

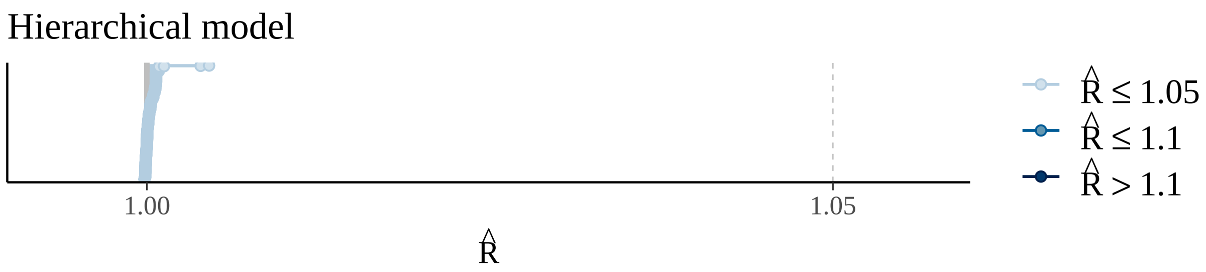
3.1 $\hat{R}$ and effective sample size

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 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.

```r
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"
  )
)
```

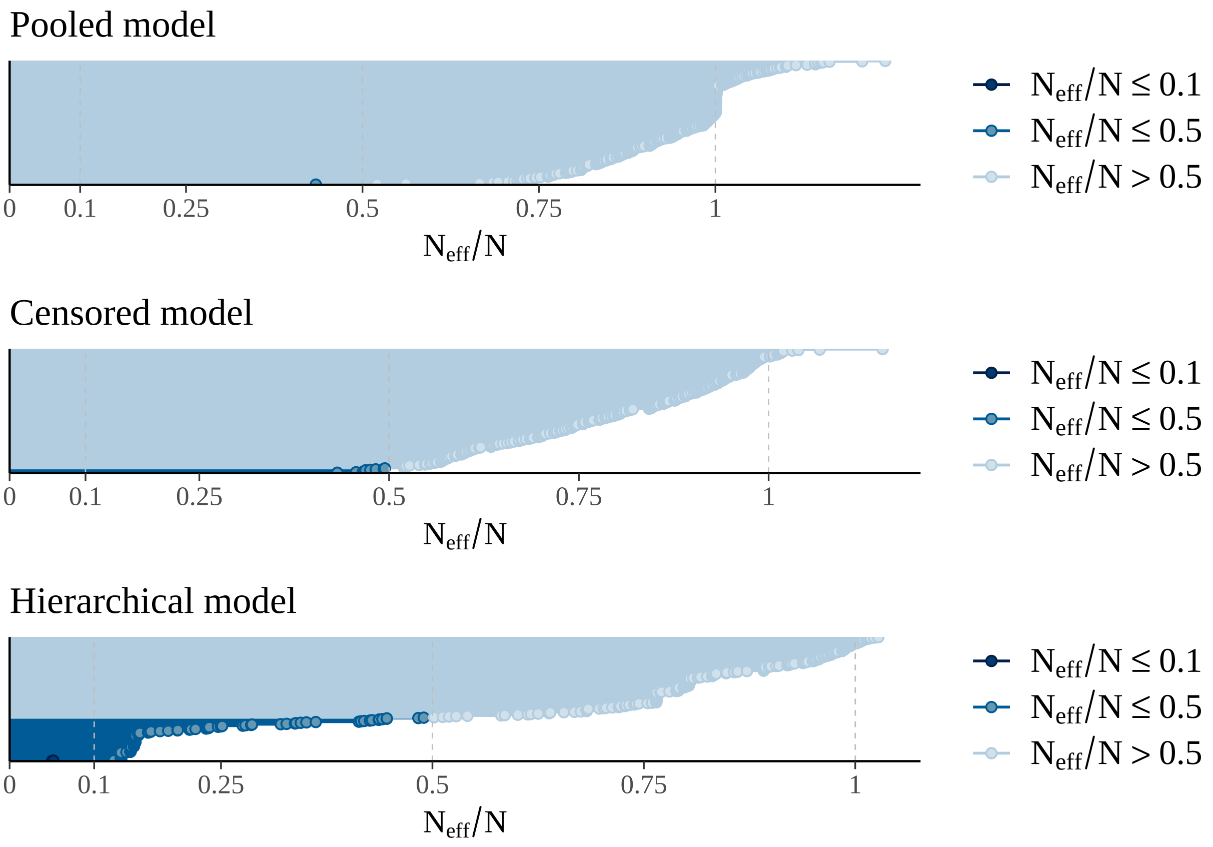






We find that all our models have converged and that the chains agree on the estimates, which is good news for our parameter estimates. We now look at the effective sample size of our MCMC draws for each our three models. The effective sample size, n_{eff}, is an estimate of the number of independent draws from the posterior distribution are statistically important towards estimating a given parameter. 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 simulated sequence and that the simulations suffice for practical purposes as is argued in Gelman et al. [2020].

```
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, 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 $n_{\text{eff}} \leq 0.1$, which is slightly concerning regarding the model's future performance.

3.2 Divergence parameters

There is small number of divergent transitions, which is identified by `divergent__` being 1. Also, chains have a `treedepth__` of at most 10 and the default is 10. The maximum number is not exceeded, so the sampler does not hit its limit on the number of leapfrog steps, taken per iteration.

```
# 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)
```

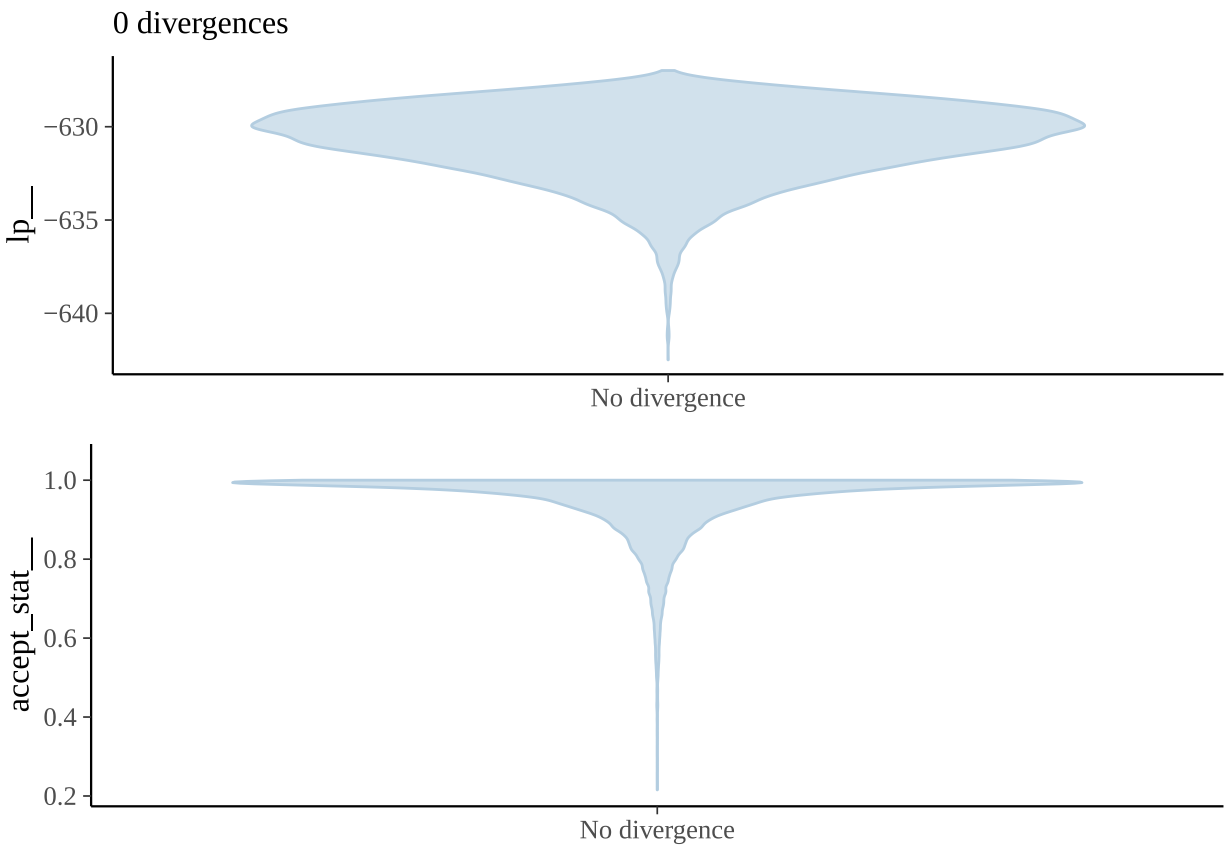
```
##  accept_stat__    stepsize__      treedepth__  n_leapfrog__   divergent__
##  Min.   :0.00   Min.   : 0.000   Min.   : 0   Min.   :   1   Min.   :0.00
##  1st Qu.:0.84   1st Qu.: 0.089   1st Qu.: 5   1st Qu.:  31   1st Qu.:0.00
##  Median :0.95   Median : 0.096   Median : 5   Median :  31   Median :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)
```

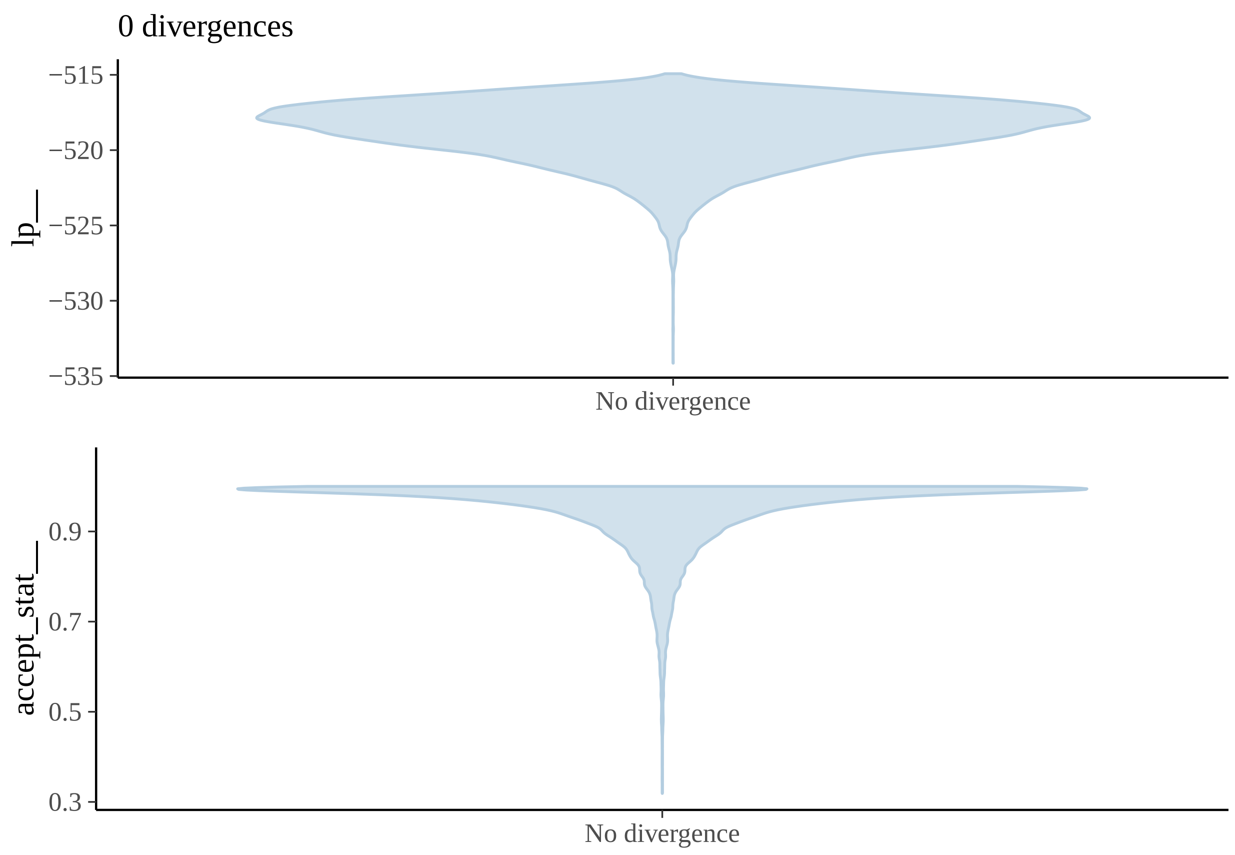
```
##  accept_stat__    stepsize__       treedepth__    n_leapfrog__   divergent__
##  Min.   :0.00   Min.   : 0.0000   Min.   : 0.0   Min.   :   1   Min.   :0.00
##  1st Qu.:0.87   1st Qu.: 0.0042   1st Qu.: 6.0   1st Qu.:  63   1st Qu.:0.00
##  Median :0.96   Median : 0.0134   Median : 6.0   Median :  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 as follows. We first start with the pooled model and the censored model below.

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

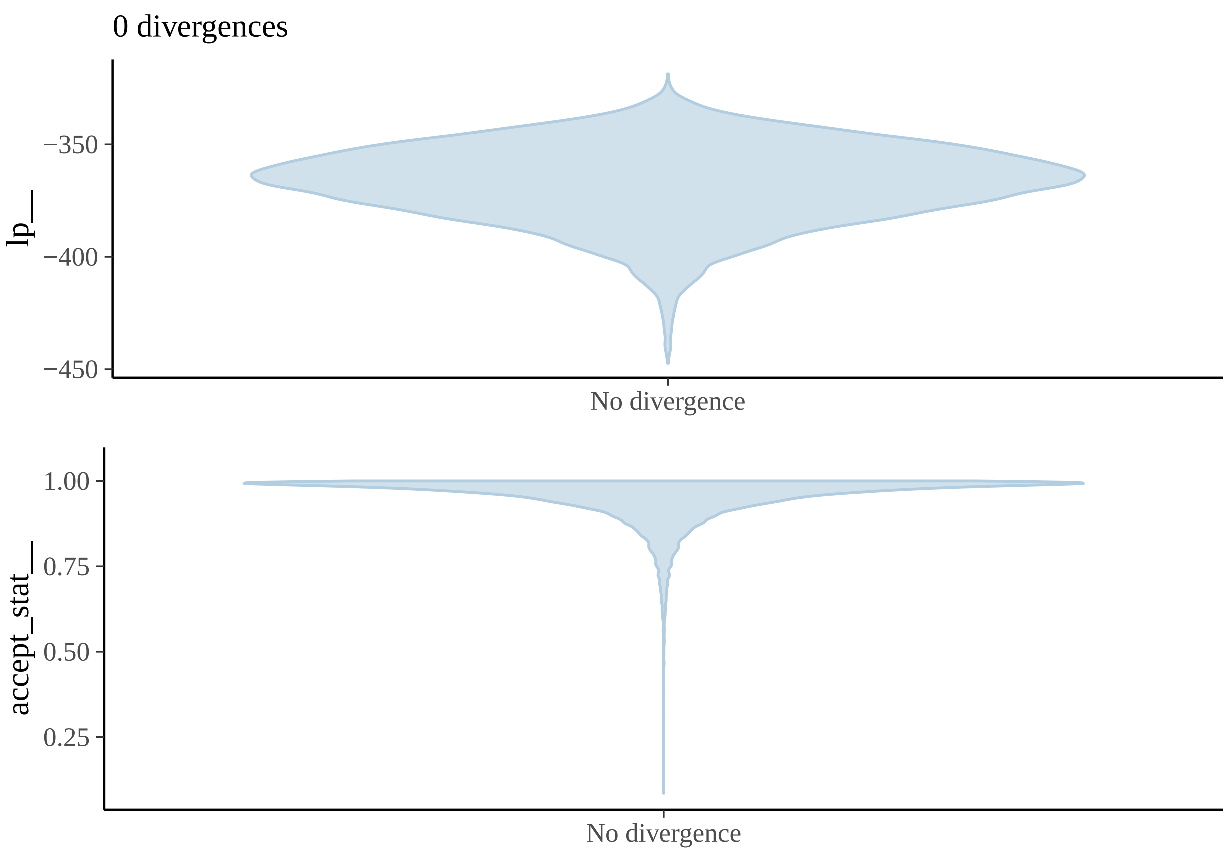


```
# plot divergence statistics for the censored model
bayesplot::mcmc_nuts_divergence(nuts_params(weibull_cens), log_posterior(weibull_cens))
```



We find that there are no divergences following the warmup, which is great! Now moving on the hierarchical model, we find that there were some divergences.

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

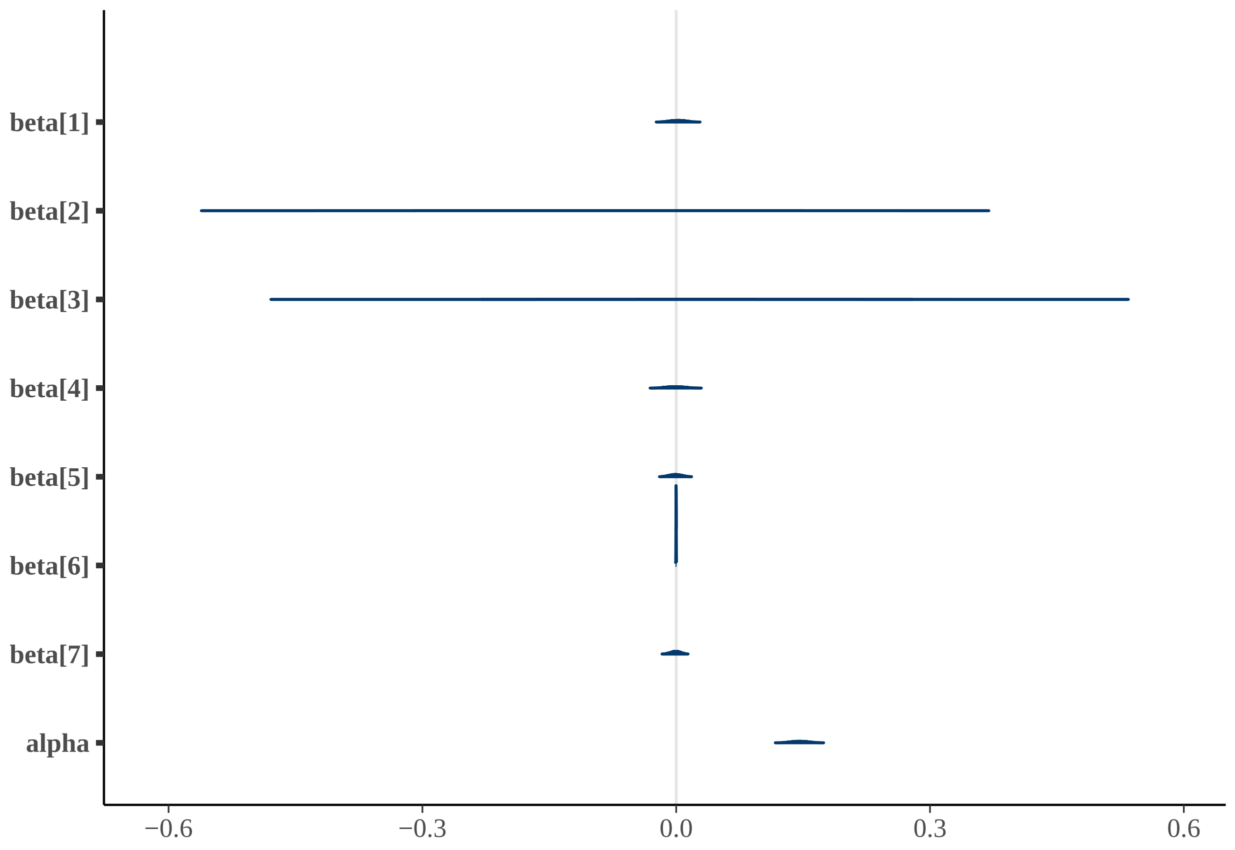


We find that divergent steps have a lower than average acceptance statistic to non-divergent steps, and also a lower log-posterior statistic. This means that even when we have a divergent step, it is not greatly affecting the posterior distribution, which is good!

3.3 Intervals for posterior data

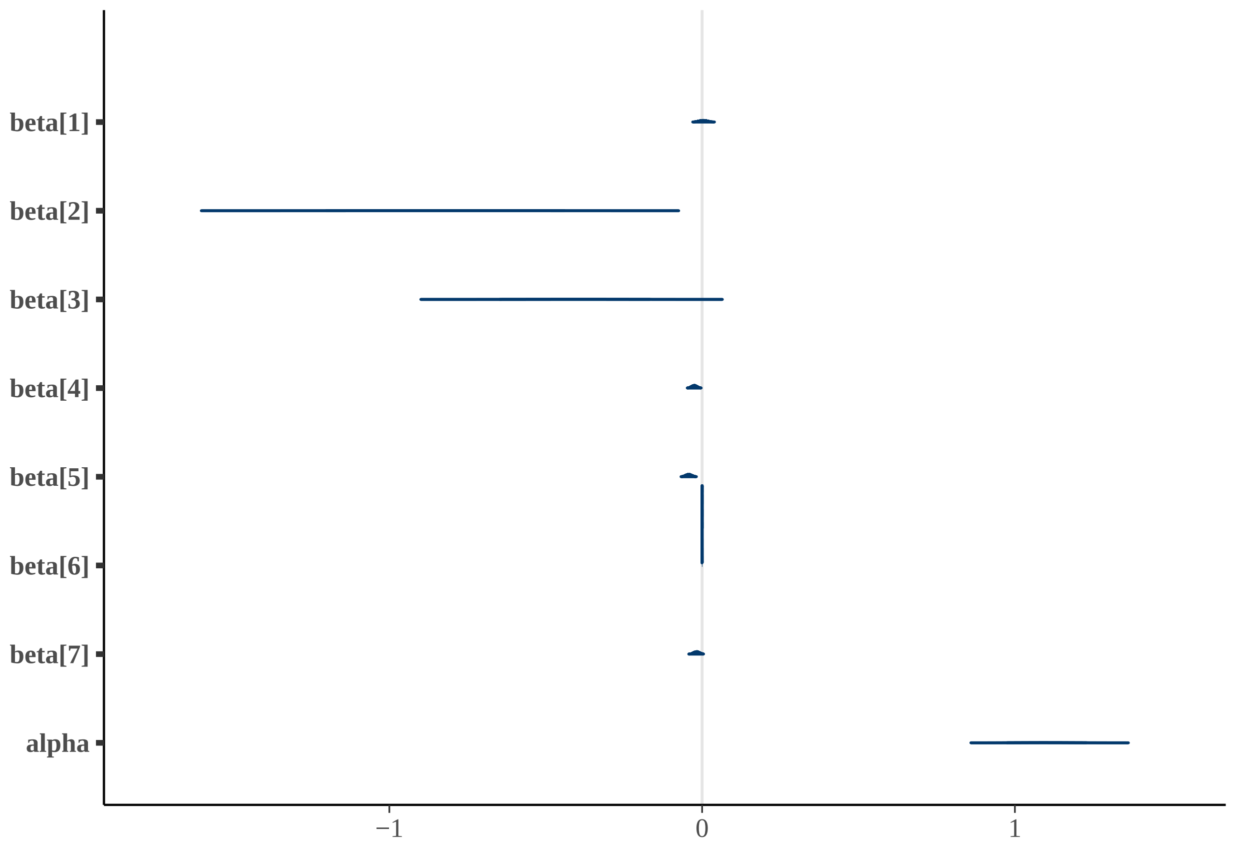
We will now plot the posterior distributions of the weights of our regressors in our GLMs and the shape parameter α for our three models, starting with our pooled GLM not considering censored data.

```
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 to our variate. 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.

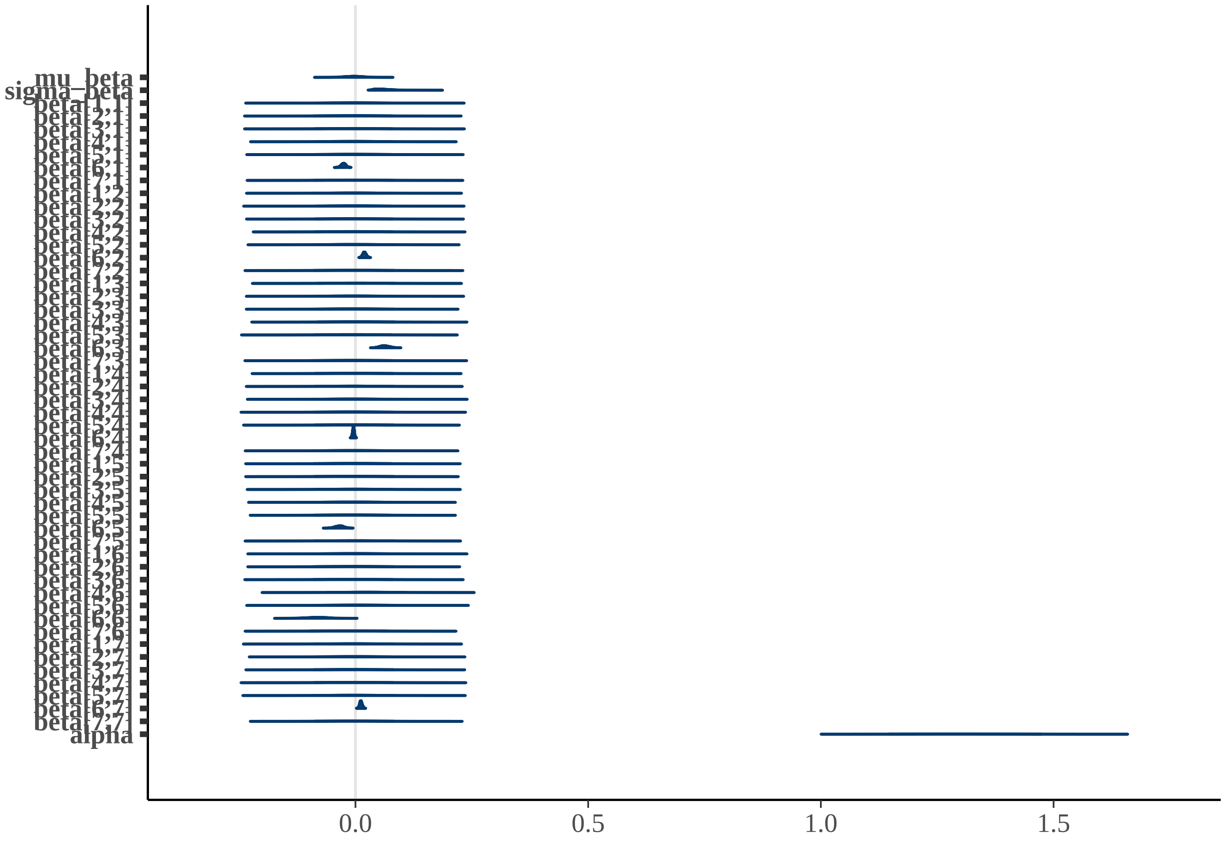
```r
cens_weibull_posterior <- as.data.frame(weibull_cens)
mcmc_areas(
  cens_weibull_posterior,
  pars = colnames(cens_weibull_posterior)[
    (colnames(cens_weibull_posterior) %like% "beta")
    | (colnames(cens_weibull_posterior) %like% "alpha")
  ],
  prob = 0.8,
  prob_outer = 0.99,
  point_est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

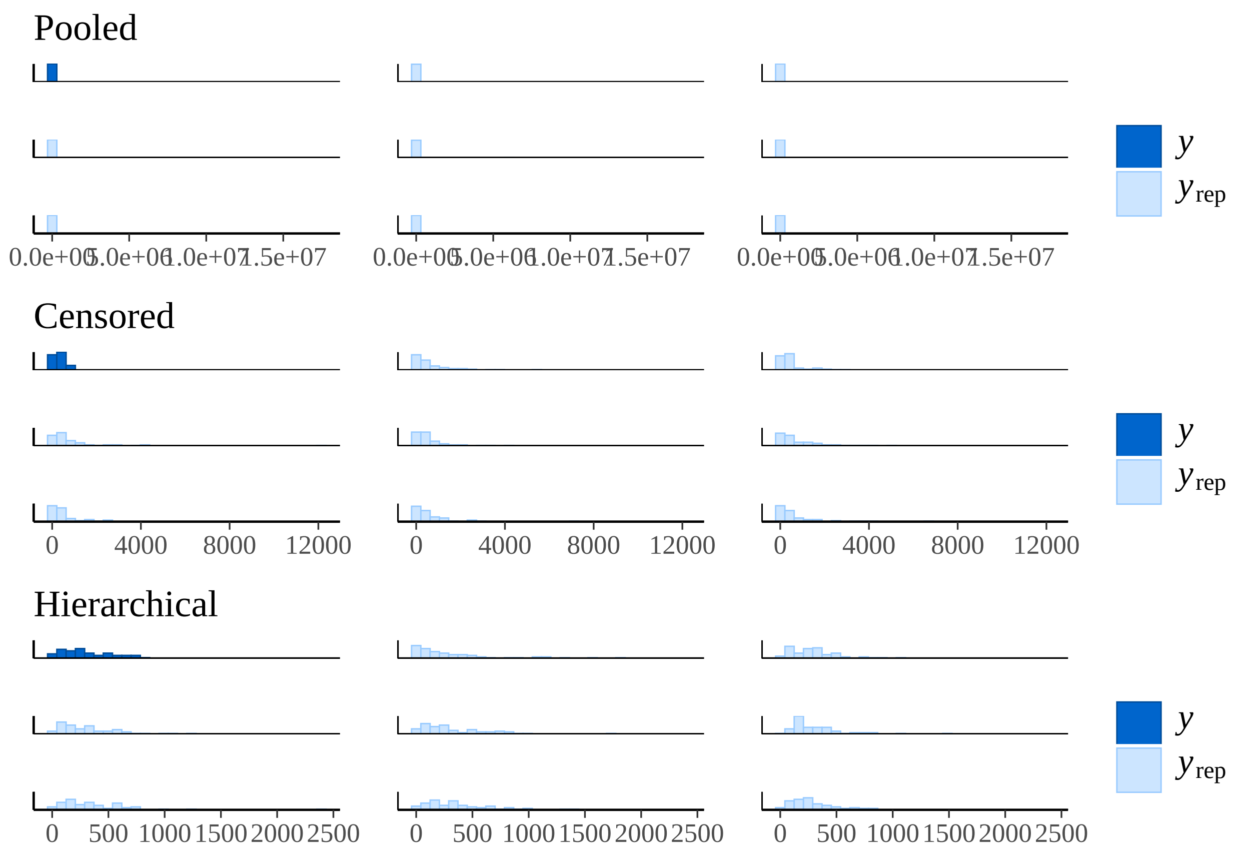
```r
hier_posterior <- as.data.frame(weibull_hier)
mcmc_areas(
  hier_posterior,
  pars = colnames(hier_posterior)[
    (colnames(hier_posterior) %like% "beta")
    | (colnames(hier_posterior) %like% "alpha")
  ],
  prob = 0.8,
  prob_outer = 0.99,
  point_est = "mean"
  )
```



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 pooled model the posterior results are similar to original survival survival distribution. Here, the censored data is not includled, so all the data is mainly concentrated near particular peak.

For censored model the posterior draws are similar to the original distribution. However, for some replicates the peak of the distribution is a little bit lower than for original data.

For hierarchical model there are more broaden results, as here different institutional groups are taken into consideration. The predictive values in most cases follow the same distribution with, of course, some noise.

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. 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].

```r
# 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.
```

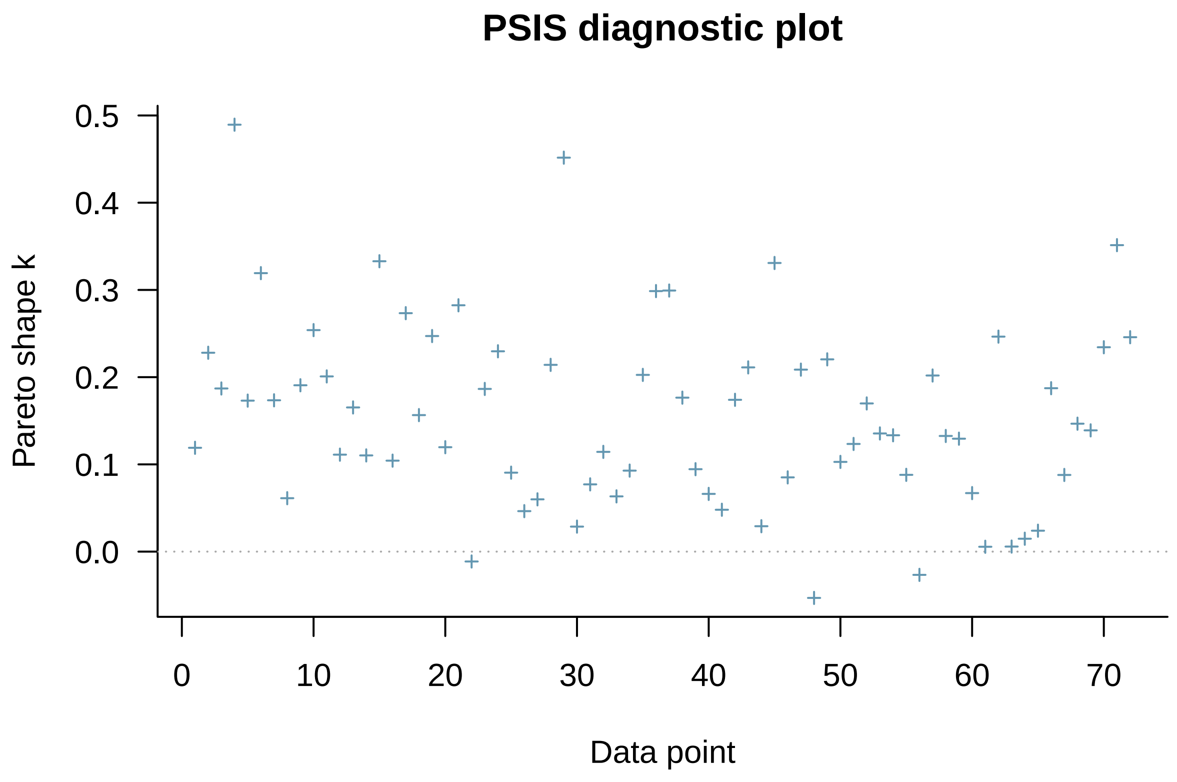
```
# repeat for hierarchical Weibull
hwm_log_lik <- extract_log_lik(weibull_hier, merge_chains = FALSE)
hwm_r_eff <- relative_eff(exp(hwm_log_lik), cores = parallel::detectCores())
hwm_loo <- loo(hwm_log_lik, r_eff = hwm_r_eff, cores = parallel::detectCores())
```

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

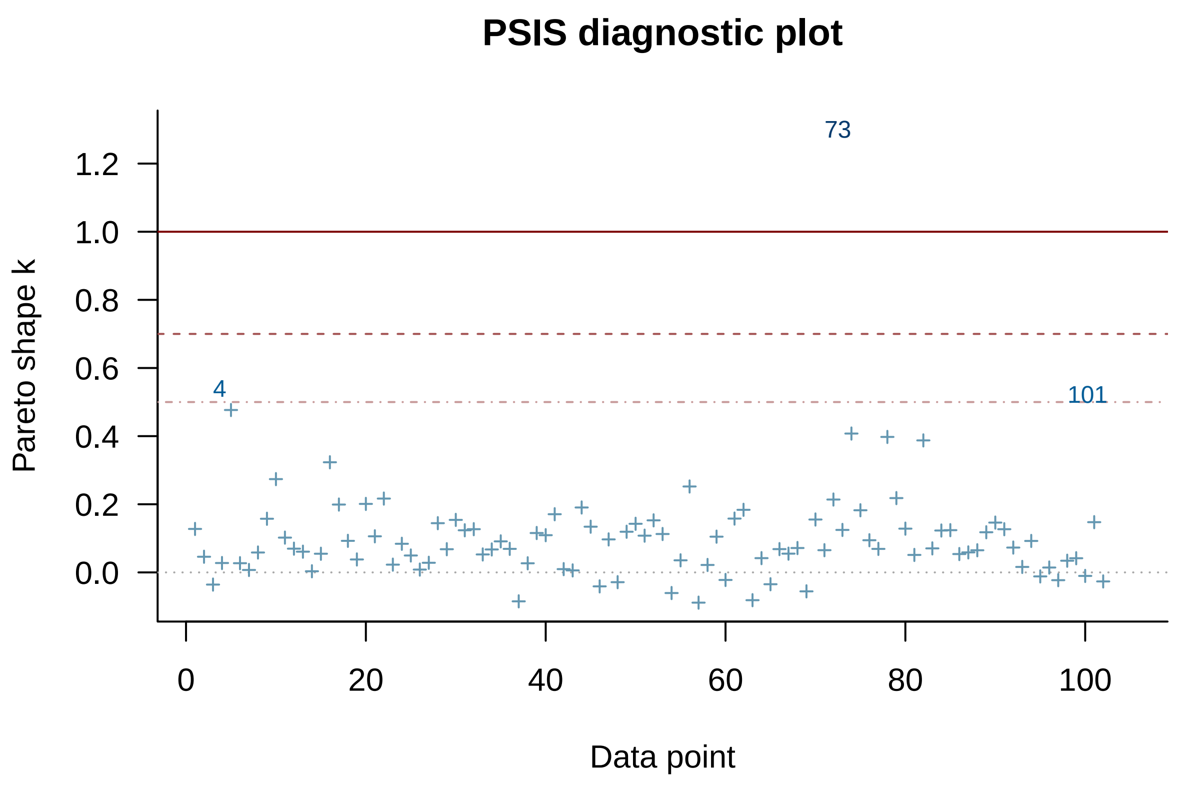
```
hwm_waic <- waic(hwm_log_lik, cores = parallel::detectCores())
```

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

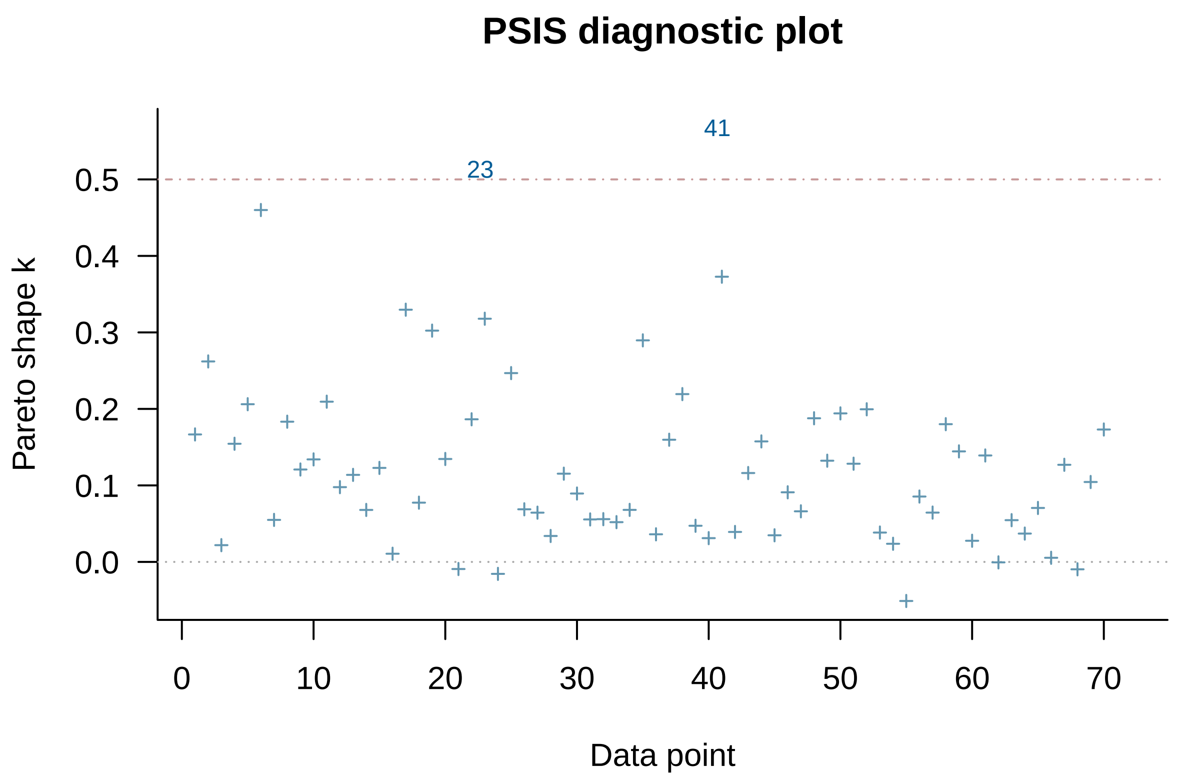
```
# plot pareto k diagnostics for the models
plot(wm_loo, label_points = TRUE)
```



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



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



```
# 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.
```

```
# 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.
```

Considering the Pareto k-values, for pooled model all values are good. For the censored model one value is larger than 1, all others are good Hierarchical model shows better performance in that case, as for it all k-values are lower than 0.7 like the pooled model.

Comparing the ELPD, the best (largest) value is shown by hierarchical model, than comes the model with censored data, after that the pooled model. 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 would thus not be 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 pooled the optimal number of parameters is ~5.5 and given that there 7 in the model, the result is pretty close. For censored ~13 params are shown and we actually have 15 parameters, which is again pretty close. For the hierarchical model, the `p_loo` shows 6.6 parameters.

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.

For the pooled model, ELPD $= -632$, for the pooled model censored 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 models.

The dataset is not very small and posterior inferences based on large numbers of MCMC iterations tend to be not 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], the variance hyperparameter for the hierarchical model was changed to a Half-Cauchy (the suggested priors for fewer hierarchical groups), 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. 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. 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 problem may be that the distribution is rather skewed. Therefore, it is rather difficult to choose for it a competent distribution. In that work we chose Weibull, maybe in the future it'd interesting to try one more distribution - gamma or such extensions of exponential, like: exponentiated exponential (has properties, similar to Gamma distribution, but survival function like a Weibull).

Another issue was the relatively low n_{eff} of the hierarchical model. 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
  // implies: 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 many data. A Stan implementation is available at McLatchie and Odnoblyudova [2021] for further interest.

Also, maybe it'd be worth to pay more attention to features and apply extra transformation (except mean-centering) or even extraction to them. Moreover, the priors can be tested even more accurately, variants like gamma can be tried.

The sampling algorithms needs rather large number of iteration to converge, some reparametrization can be made to decrease that number.

4.2 Things learnt from the data analysis

First of all, it has to be said that the dataset is rather complex and data analysis process only confirms it. It was understood that the survival time of patients with advanced lung cancer can be approximated by Weibull distribution. And the thoughts that the data belonging to various institutions may have different shape and scale parameters have been confirmed. That is why, the model with hierarchical structure, which provides a little different distributions for each institute parameters is better, it was confirmed by ELPD, WAIC parameters and PSIS diagnostic plot.

4.3 Self-reflection and learnings

The group learnt a lot about the applications of MCMC methods to Survival analysis. Previously, some of us did not even know about the concept of Survival analysis, about such terms, like survival, hazard function, or with what distributions survival time can be simulated. Also, previously, there wasn't chance to dive deep into the Weibull regression models, now degree of understanding of this model increased. 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` were discovered (connected with bayesian statistics), about which have never heard before.

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^ 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.