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 exploration of survival of patients with lung cancer is crucial for controlling the disease development, obtaining right treatment methods, 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 analyzing time to event data. Exploring survival models can be challenging, since different models from non-parametric to parametric can be used, various distributions, like exponential, Weibull, log-normal, are applicable for each concrete case. The most common model is Cox hazard model, however, it is too simple and proposes constant effect of predictor variables on survival duration throughout time.

This study examines the way Bayesian approach proceeds to fit Weibull model for lifetime data of patients with advanced lung cancer analysis. Weibull approach is more flexible and hazard rate is not 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.

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, Therneau [2021].

The problem, tring to be solved, is connected with the prediction of survival time of patients with lung cancer.

Dataset contains 9 features and 228 observations, which are assumed to be independent and identically distributed. The target variable is the survival time in days. The covariates are presented by both categorical and numerical values.

Special attention has to be paid for "censoring status" feature. It indicates if the patient had an event (=1) or not (=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. We can get rid of it.

There are three variables, needed to be explained: ph.ecog - ECOG performance score (0-4). 0-good condition, 4-the worst condition, much time in bad. ph.karno - Karnofsky performance score (bad=0-good=100). Provided by physician. pat.karno - Karnofsky performance score. Provided by patient.

1.1.2 Exploratory data analysis

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

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

There were 61 observations with missed values, which have been removed from dataset. Now it consists of 167 rows.

```
data = cancer %>% na.omit()
```

Institutions are considered as variables, useful for hierarchical model.

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

```
##
##  1  2  3  4  5  6  7 10 11 12 13 15 16 21 22 26 32
## 28  4 12  4  7 12  7  4 13 16 13  6 10  8 13  4  6
```

Moving to categorical variables (fig.1), the number of men prevails over women. 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"))
```

The distribution for continuous variables is shown in fig. 2. The features do not follow 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 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, eventually, patient and doctor measure the same quantity.

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

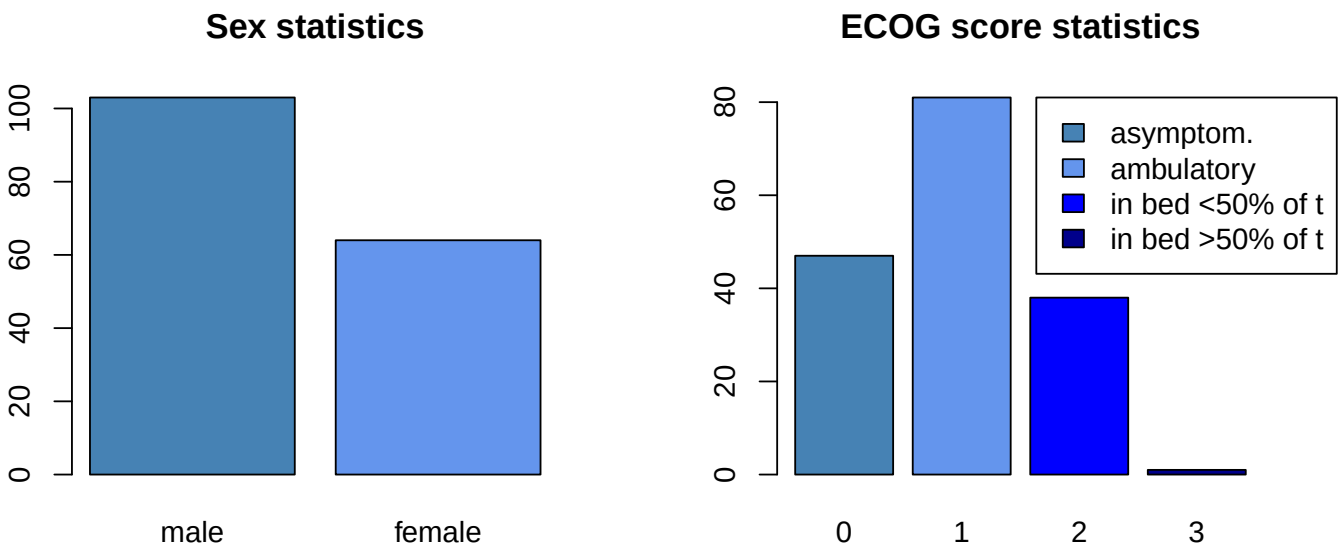


Figure 1: Categorical variables

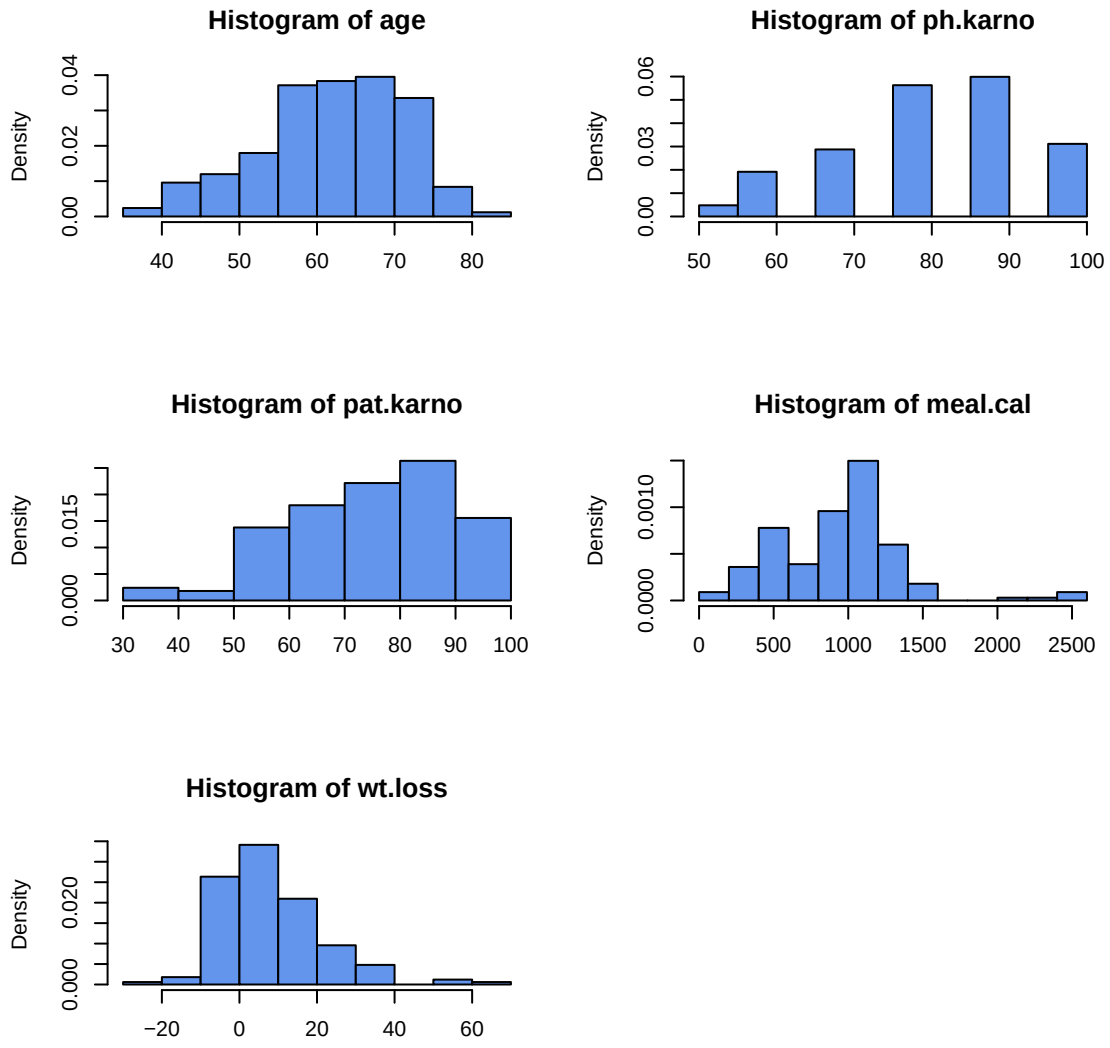


Figure 2: Continuous variables

```
##                  age    ph.karno  pat.karno    meal.cal     wt.loss
## age         1.00000000 -0.32261297 -0.2398974 -0.23958240  0.04286056
## ph.karno   -0.32261297  1.00000000  0.5350275  0.05385409 -0.12524032
## pat.karno  -0.23989736  0.53502749  1.0000000  0.17465190 -0.18213953
## meal.cal   -0.23958240  0.05385409  0.1746519  1.00000000 -0.11134425
## wt.loss     0.04286056 -0.12524032 -0.1821395 -0.11134425  1.00000000
```

1.2 Relative 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 **?** the comparison of semi-parametric and non-parametric models for survival analysis was presented, but different dataset was used, also there was no deep description of Weibull model implementation, i.e. more attention was payed to Cox regression. Study 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 that work two Weibull Survival models are built: hierarchical and non-hierarchical. The main approaches and theory of building survival models were used from the studies above, but the stan implementation, priors choice was made by the authors.

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.

```
# install libraries
library(survival)
library(tidyverse)
library(rstan)
library(brms)
library(bayesplot)
library(loo)
library(reshape2)
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")
```

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

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

$$\begin{aligned} y_i &\sim \text{Weibull}(\sigma, \alpha)\,, \quad i = 1, \ldots, N, \\ \alpha &\sim \text{Half-Cauchy}(5), \\ \sigma &= \exp\left(-\frac{\boldsymbol{X}\beta}{\alpha}\right), \\ \beta_k &\sim N(0, 10)\,, \quad k = 1, \ldots, M. \end{aligned}$$

The choice of a Half-Cauchy prior is motivated in 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 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 Figure **??**.

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")
```

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

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

```
FALSE S4 class stanmodel 'weibull_survival' coded as follows:
FALSE data {
FALSE   int<lower=0> N; // number of data realisations
FALSE   int<lower=0> M; // feature dimensionality
FALSE   vector<lower=0>[N] y; // survival time
FALSE   matrix[N, M] X; // design matrix
FALSE }
FALSE
FALSE transformed data {
FALSE   matrix[N, M] Xc;  // centered version of X without an intercept
FALSE   vector[M] means_X;  // column means of X before centering
FALSE
```

```
FALSE   // column-center the design matrix for fitting the model
FALSE   for (m in 1:M) {
FALSE     means_X[m] = mean(X[, m]);
FALSE     Xc[, m] = X[, m] - means_X[m];
FALSE   }
FALSE }
FALSE
FALSE parameters {
FALSE   // GLM parameters
FALSE   vector[M] beta; // regressors
FALSE   real<lower=0> alpha;  // shape parameter
FALSE }
FALSE
FALSE transformed parameters {
FALSE   // compute latent predictor term
FALSE   vector[N] eta = Xc * beta;
FALSE   // apply the log inverse link function
FALSE   vector<lower=0>[N] sigma = exp(-eta / alpha);
FALSE }
FALSE
FALSE model {
FALSE   // prior over regressor and shape parameters
FALSE   beta ~ normal(0, 1);
FALSE   alpha ~ cauchy(0, 5);
FALSE
FALSE   // fit model
FALSE   y ~ weibull(alpha, sigma);
FALSE }
FALSE
FALSE generated quantities {
FALSE   // compute predictive distribution for survival time
FALSE   real ypred[N] = weibull_rng(alpha, sigma);
FALSE
FALSE   // log-likelihood
FALSE   vector[N] log_lik;
FALSE   for (n in 1:N) {
FALSE     log_lik[n] = weibull_lpdf(y[n] | alpha, sigma[n]);
FALSE   }
FALSE }
```

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

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, 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

Weakly-informative priors for parameters are used. There are 7 covariates in the model, for each regressor β can take positive and negative values, however, these values are not likely to be very large. For them, normal distribution $N(0, 10)$ can be used. Also, the prior has to be made for the shape parameter, $Gamma(1, 1)$ is a variant, as the majority of the variance is not too close to zero, it has rather long tail, but which is not too heavy. Even we are not experts-oncologists, we know that typically person's 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.

$$\beta \sim N(0, 10)$$

$$\alpha \sim \text{Gamma}(1, 1)$$

Chosen priors do not contribute strongly to the posterior, so the data can "speak for itself".

2.2.2 Implemented in Stan

Data preparation

```
# 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

```
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")
```

```
FALSE Running /anaconda3/envs/seurat4/lib/R/bin/R CMD SHLIB foo.c
FALSE x86_64-apple-darwin13.4.0-clang -I"/anaconda3/envs/seurat4/lib/R/include" -DNDEBUG   -I"/anaconda3/e
FALSE In file included from <built-in>:1:
FALSE In file included from /anaconda3/envs/seurat4/lib/R/library/StanHeaders/include/stan/math/prim/mat/f
FALSE In file included from /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Dense:1:
FALSE In file included from /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Core:88:
FALSE /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: u
FALSE namespace Eigen {
```

```
FALSE ^
FALSE /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error:
FALSE namespace Eigen {
FALSE                ^
FALSE                ;
FALSE In file included from <built-in>:1:
FALSE In file included from /anaconda3/envs/seurat4/lib/R/library/StanHeaders/include/stan/math/prim/mat/f
FALSE In file included from /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Dense:1:
FALSE /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' fil
FALSE #include <complex>
FALSE          ^~~~~~~~~
FALSE 3 errors generated.
FALSE make: *** [/anaconda3/envs/seurat4/lib/R/etc/Makeconf:174: foo.o] Error 1
```

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

```
FALSE S4 class stanmodel 'weibull_censored' coded as follows:
FALSE data {
FALSE   int<lower=0> N;    // number of object
FALSE   int<lower=0> M;    // number of features
FALSE   int<lower=0> Ncen;    // number of object
FALSE   vector<lower=0>[N] yobs; // target-survival time
FALSE   matrix[N, M] Xobs;    // covariates
FALSE   vector<lower=0>[Ncen] ycen; // target-survival time
FALSE   matrix[Ncen, M] Xcen;    // covariates
FALSE }
FALSE
FALSE parameters {
FALSE   vector[M] beta;       // regressors
FALSE   real<lower=0> alpha;  // shape parameter
FALSE }
FALSE
FALSE transformed parameters {
FALSE   // Log inverse link function
FALSE   vector<lower=0>[N] sigma = exp(-Xobs*beta / alpha);
FALSE }
FALSE
FALSE model {
FALSE   // priors
FALSE   beta ~ normal(0, 10);
FALSE   alpha ~ cauchy(0, 5);
FALSE
FALSE   // fitting model
FALSE   yobs ~ weibull(alpha, sigma);
FALSE
FALSE   // Increment log-density with Survival Function
FALSE   target += weibull_lccdf(ycen | alpha, exp(-Xcen*beta / alpha));
FALSE }
FALSE
FALSE generated quantities {
FALSE   // compute predictive distribution for survival time
FALSE   real ypred[N] = weibull_rng(alpha, sigma);
FALSE
FALSE   // log-likelihood
```

```
FALSE   vector[N+Ncen] log_lik;
FALSE   for (i in 1:N) {
FALSE     log_lik[i] = weibull_lpdf(yobs[i] | alpha, sigma[i]);
FALSE   }
FALSE   // Survival function
FALSE   for (j in 1:Ncen){
FALSE     log_lik[N+j] = weibull_lccdf(ycen[j] | alpha, exp(-Xcen[j,]*beta / alpha));
FALSE   }
FALSE }
```

```
# learn the model with parameters 4 chains, 5000 iterations for each, 2500 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

$$\begin{aligned} y_{ij} &\sim \text{Weibull}(\sigma_j, \alpha)\,, \quad i = 1, \ldots, N,\, j = 1, \ldots, J \\ \alpha &\sim \text{Half-Cauchy}(5), \\ \sigma &\propto \boldsymbol{X}\beta, \\ \beta_{kj} &\sim N(0, 1)\,, \quad k = 1, \ldots, M,\, j = 1, \ldots, J. \end{aligned}$$

This is seen visually below in Figure **??**.

2.3.1 Defining the link function

2.3.2 Priors

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

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])
# print(dim(X))
# [1] 120   7
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")
```

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

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

```
FALSE S4 class stanmodel 'weibull_hier' coded as follows:
FALSE data {
FALSE   int<lower=0> N;               // number of object
FALSE   int<lower=0> M;               // number of features
FALSE   int<lower =0> J;              // number of institutions
FALSE   vector<lower=0>[N] y;         // target-survival time
FALSE   matrix[N, M] X;               // covariates
FALSE   int<lower=1, upper=J> ll[N];  // instution labels
FALSE }
FALSE
FALSE transformed data {
FALSE   matrix[N, M] Xc;  // centered version of X without an intercept
FALSE   vector[M] means_X;  // column means of X before centering
FALSE
FALSE   // column-center the design matrix for fitting the model
FALSE   for (m in 1:M) {
```

```
FALSE     means_X[m] = mean(X[, m]);
FALSE     Xc[, m] = X[, m] - means_X[m];
FALSE   }
FALSE }
FALSE
FALSE parameters {
FALSE   // hyperpriors
FALSE   real mu_beta;
FALSE   real sigma_beta;
FALSE
FALSE   // regressors
FALSE   matrix[M, J] beta;            // regressors weights for different institutions
FALSE   real<lower=0> alpha;          // shape parameter
FALSE }
FALSE
FALSE transformed parameters {
FALSE   vector[N] eta;
FALSE   vector<lower=0>[N] sigma;
FALSE   // compute latent predictor term
FALSE   for (n in 1:N) {
FALSE     eta[n] = Xc[ll[n], ] * beta[, ll[n]];
FALSE   }
FALSE   // apply the log inverse link function
FALSE   sigma = exp(-eta / alpha);
FALSE }
FALSE
FALSE model {
FALSE   // hyperpriors
FALSE   mu_beta ~ normal(0, 1);
FALSE   sigma_beta ~ gamma(1, 1);
FALSE
FALSE   // prior over regressor and shape parameters
FALSE   for (j in 1:J) {
FALSE     beta[, j] ~ normal(mu_beta, sigma_beta);
FALSE   }
FALSE   alpha ~ cauchy(0, 5);
FALSE
FALSE   // fitting model
FALSE   for (n in 1:N) {
FALSE     y[n] ~ weibull(alpha, sigma[n]);
FALSE   }
FALSE }
FALSE
FALSE generated quantities {
FALSE   // define quantities
FALSE   real ypred[N];
FALSE   vector[N] log_lik;
FALSE
FALSE   // compute quantities
FALSE   for (n in 1:N) {
FALSE     // predictive distribution for survival time
FALSE     ypred[n] = weibull_rng(alpha, sigma[n]);
FALSE     // log-likelihood
FALSE     log_lik[n] = weibull_lpdf(y[n] | alpha, sigma[n]);
```

```
FALSE   }
FALSE }
```

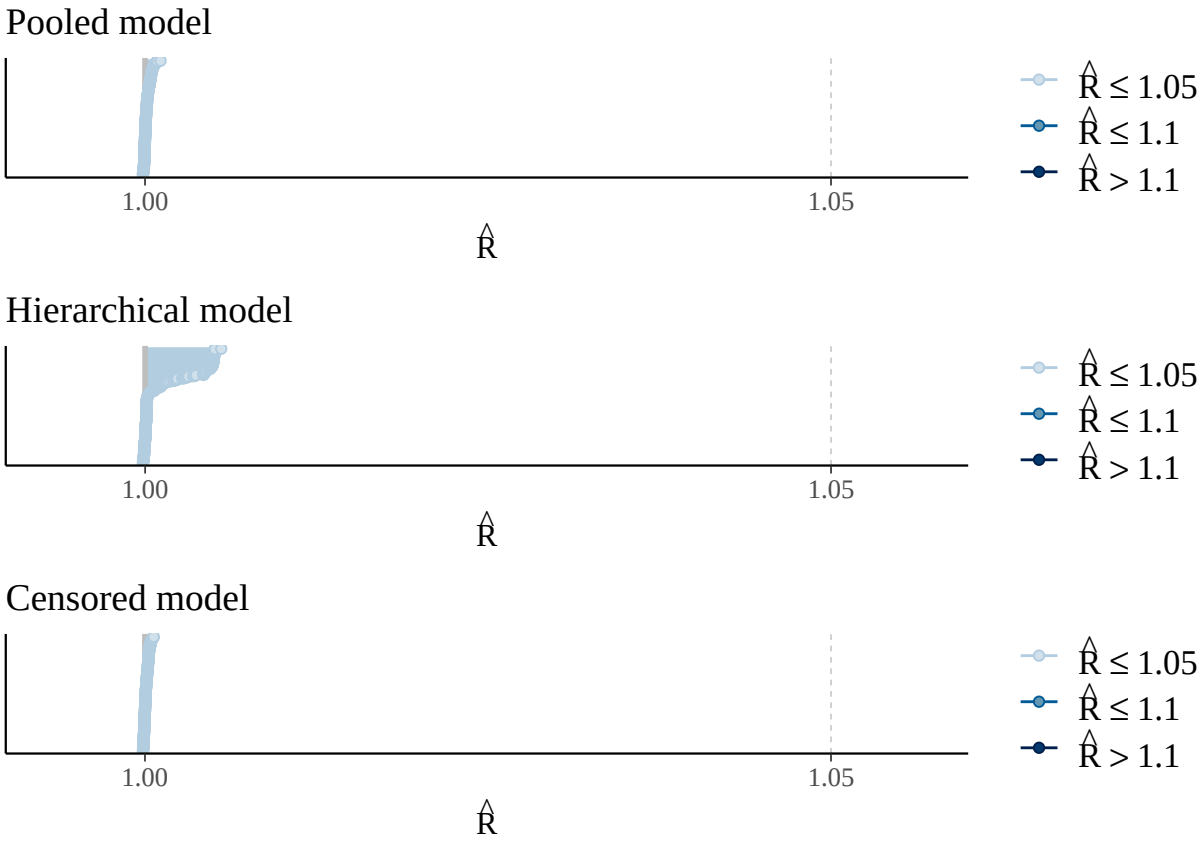
```
# learn the model parameters
weibull_hier <- rstan::sampling(whm, data = hier_data, iter = 10000)
```

3 Diagnostics and performance

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

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.

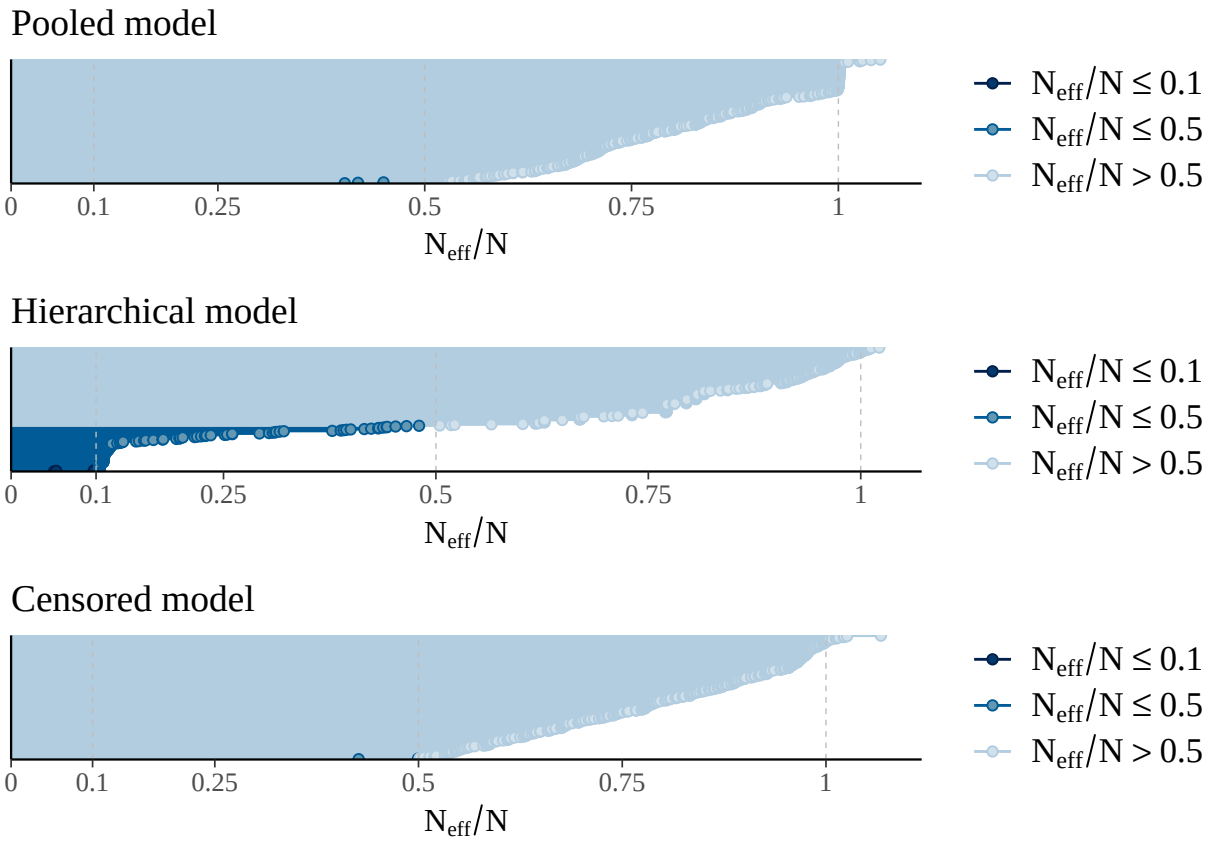
```
bayesplot_grid(
  mcmc_rhat(rhat = rhat(weibull_model)),
  mcmc_rhat(rhat = rhat(weibull_hier)),
  mcmc_rhat(rhat = rhat(weibull_cens)),
  titles = c("Pooled model", "Hierarchical model", "Censored model")
  )
```



We find that all our models have converged, 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_hier)),
  mcmc_neff(neff_ratio(weibull_cens)),
  titles = c("Pooled model", "Hierarchical model", "Censored model")
  )
```



We find that both the pooled models (not considered censored, and considered censored data respectively) have very few 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 have many such parameters with n_{eff} small, which leads us to be doubtful of the models future performance.

3.2 Divergence parameters

Censored model: 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.:0.84   1st Qu.: 0.034   1st Qu.: 4.0   1st Qu.:  15   1st Qu.:0.000
```

```
##  Median :0.95   Median : 0.114   Median : 4.0   Median :  31   Median :0.000
##  Mean   :0.86   Mean   : 0.105   Mean   : 4.5   Mean   :  79   Mean   :0.041
##  3rd Qu.:0.99   3rd Qu.: 0.159   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   :1.4e+201
##  3rd Qu.: 6.4e+02
##  Max.   :1.6e+205
```

```
# 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.   : 0.0   Min.   :   1   Min.   :0.000
##  1st Qu.:0.86   1st Qu.: 0.008   1st Qu.: 5.0   1st Qu.:  31   1st Qu.:0.000
##  Median :0.96   Median : 0.089   Median : 5.0   Median :  31   Median :0.000
##  Mean   :0.87   Mean   : 0.069   Mean   : 5.1   Mean   :  97   Mean   :0.021
##  3rd Qu.:0.99   3rd Qu.: 0.093   3rd Qu.: 6.0   3rd Qu.:  63   3rd Qu.:0.000
##  Max.   :1.00   Max.   :14.386   Max.   :10.0   Max.   :1023   Max.   :1.000
##     energy__
##  Min.   : 5.2e+02
##  1st Qu.: 5.2e+02
##  Median : 5.2e+02
##  Mean   :3.7e+157
##  3rd Qu.: 5.2e+02
##  Max.   :5.3e+160
```

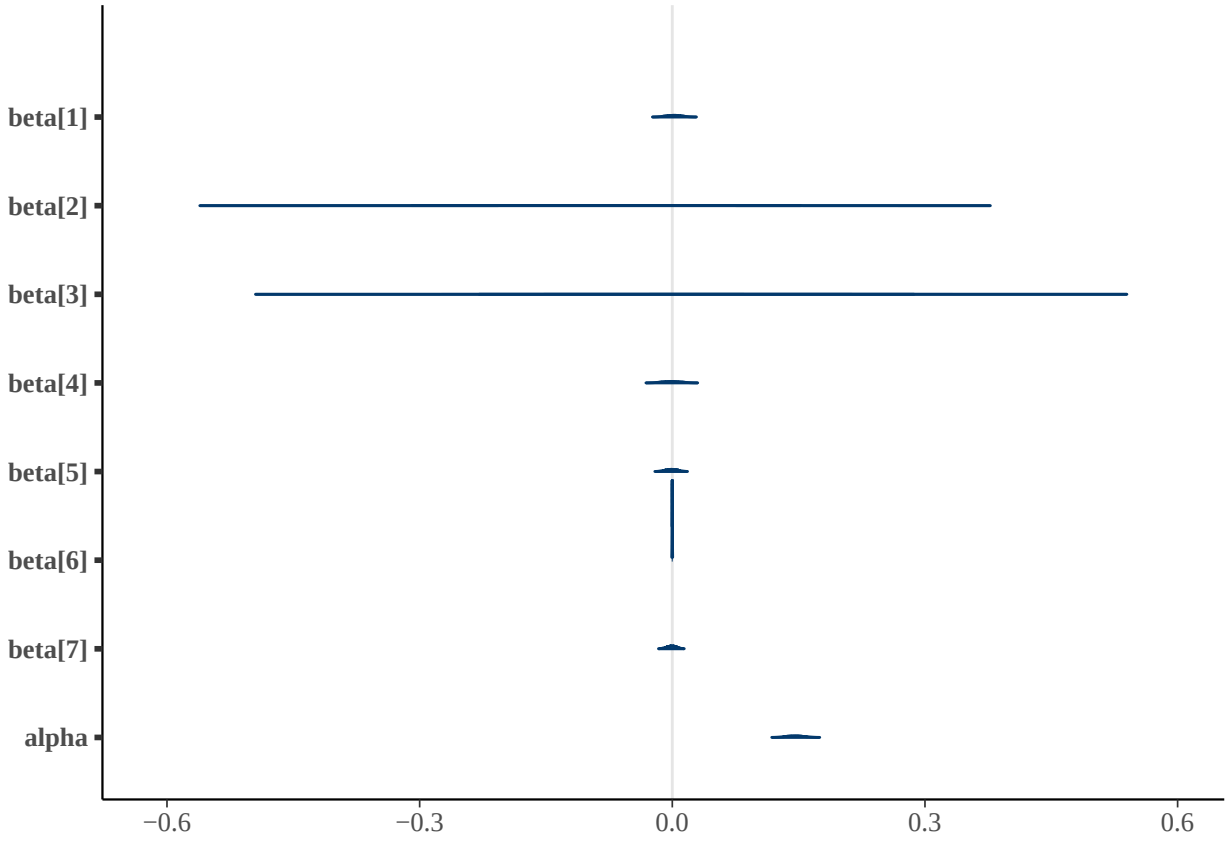
```
# hierarchical model
hwm_sampler_params = get_sampler_params(weibull_hier, inc_warmup = TRUE)
summary(do.call(rbind, hwm_sampler_params), digits = 2)
```

```
##  accept_stat__    stepsize__       treedepth__    n_leapfrog__   divergent__
##  Min.   :0.00   Min.   : 0.0000   Min.   : 0.0   Min.   :   1   Min.   :0.000
##  1st Qu.:0.86   1st Qu.: 0.0052   1st Qu.: 6.0   1st Qu.:  63   1st Qu.:0.000
##  Median :0.95   Median : 0.0296   Median : 6.0   Median :  63   Median :0.000
##  Mean   :0.87   Mean   : 0.0274   Mean   : 6.4   Mean   : 200   Mean   :0.043
##  3rd Qu.:0.99   3rd Qu.: 0.0419   3rd Qu.: 7.0   3rd Qu.: 127   3rd Qu.:0.000
##  Max.   :1.00   Max.   :14.3855   Max.   :10.0   Max.   :1023   Max.   :1.000
##     energy__
##  Min.   : 3.4e+02
##  1st Qu.: 3.8e+02
##  Median : 4.0e+02
##  Mean   :7.8e+196
##  3rd Qu.: 4.2e+02
##  Max.   :8.9e+200
```

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.

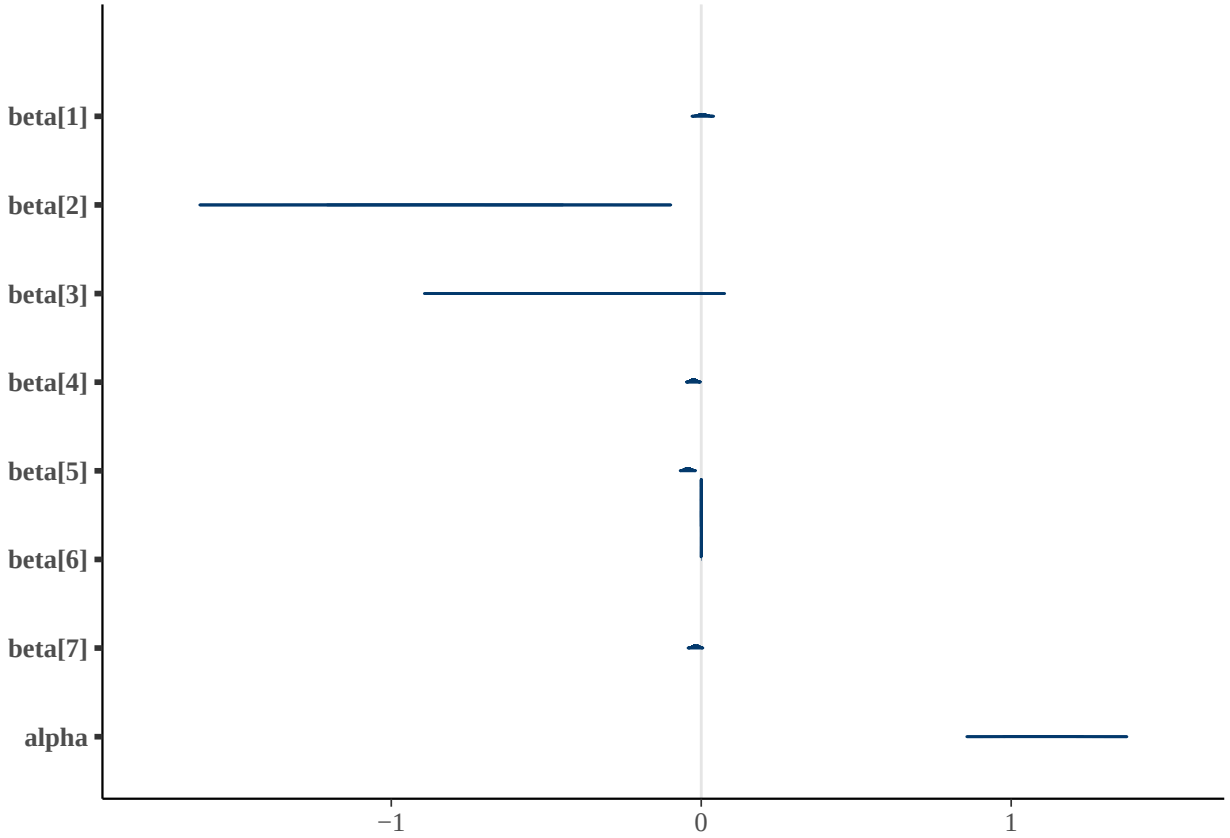
```
regression_pars <- c(
  "beta[1]",
  "beta[2]",
  "beta[3]",
  "beta[4]",
  "beta[5]",
  "beta[6]",
  "beta[7]",
  "alpha"
)
weibull_posterior <- as.data.frame(weibull_model)
mcmc_areas(
  weibull_posterior,
  pars = regression_pars,
  prob = 0.8,
  # 80% intervals
  prob_outer = 0.99,
  # 99%
  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 pooled model considering censored data.

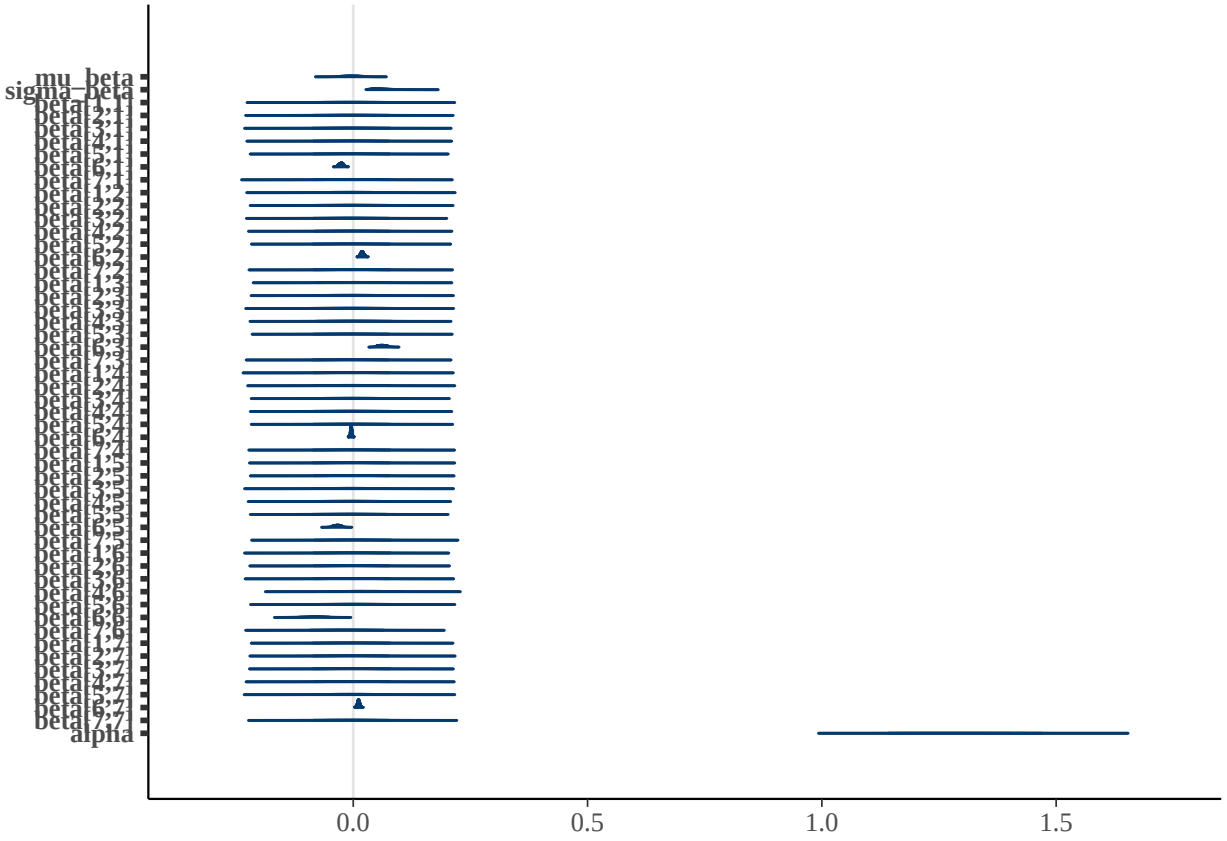
```
cens_weibull_posterior <- as.data.frame(weibull_cens)
mcmc_areas(
  cens_weibull_posterior,
```

```
  pars = regression_pars,
  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. Our hierarchical model, . . .

```
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"
  )
```

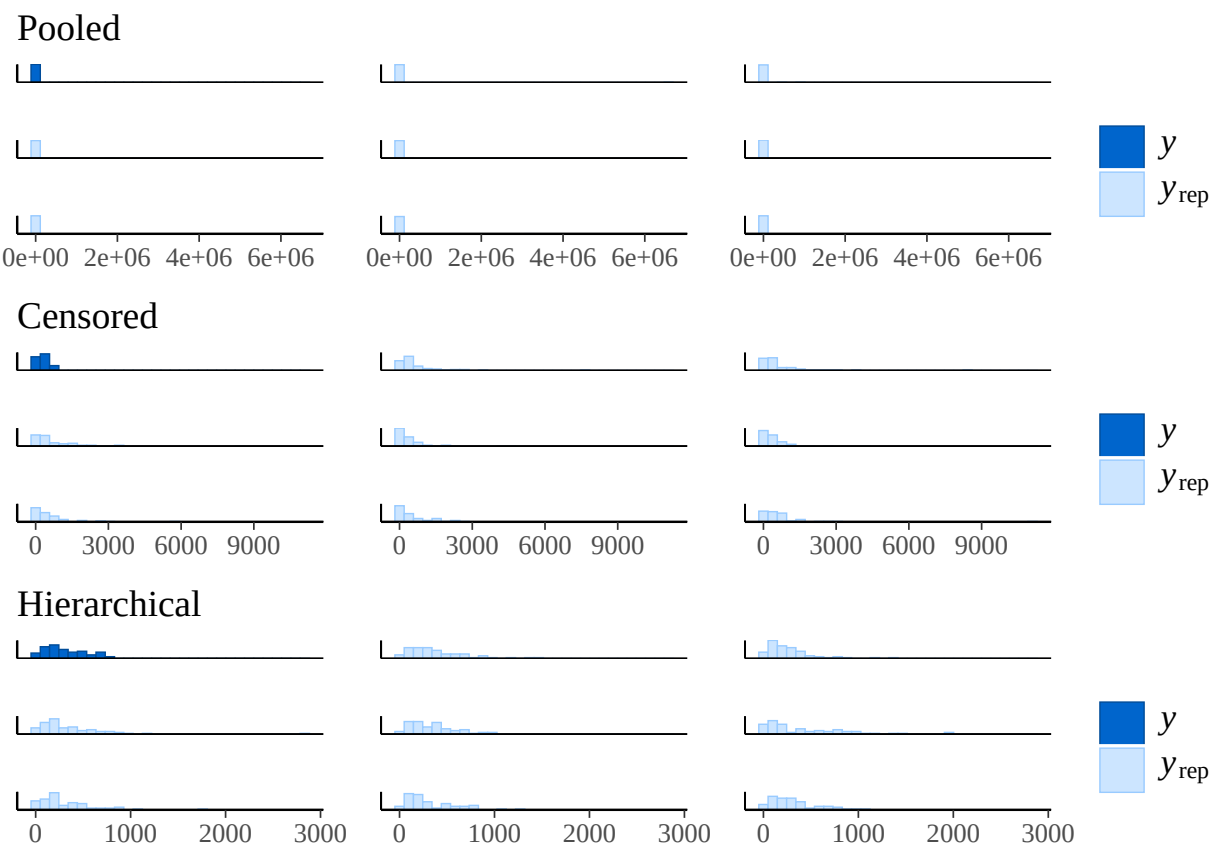


3.4 Posterior predictive checks

For censored model the posterior draws are similar to the original distribution, however, there are two main differences. In original data the peak of the distribution is larger, also, it is a little bit more narrow, for model draws sometimes wider distribution is received.

```
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`.
```



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 as it has better diagnostics.

```
# 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())
```

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

```
# 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.
```

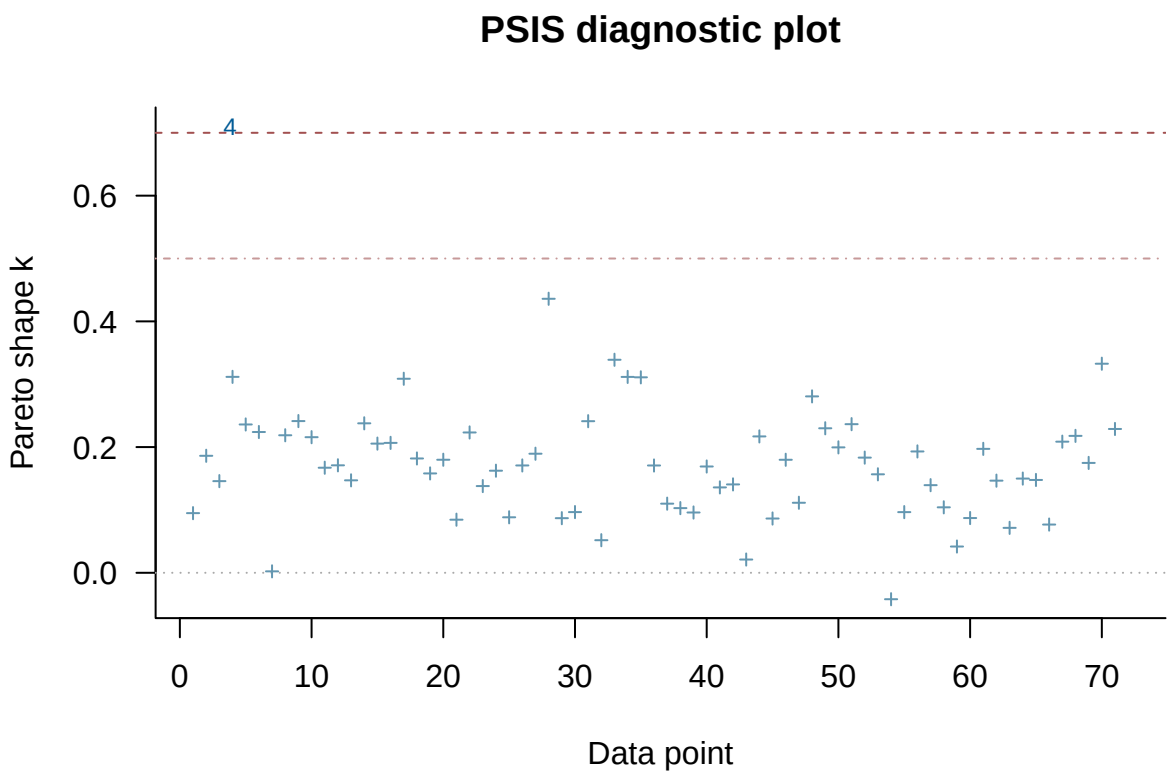
```
# repeat for hierarchical Weibull
hwm_log_lik <- extract_log_lik(weibull_hier, merge_chains = FALSE)
hwm_r_eff <- relative_eff(exp(hwm_log_lik), cores = parallel::detectCores())
hwm_loo <- loo(hwm_log_lik, r_eff = hwm_r_eff, cores = parallel::detectCores())
```

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

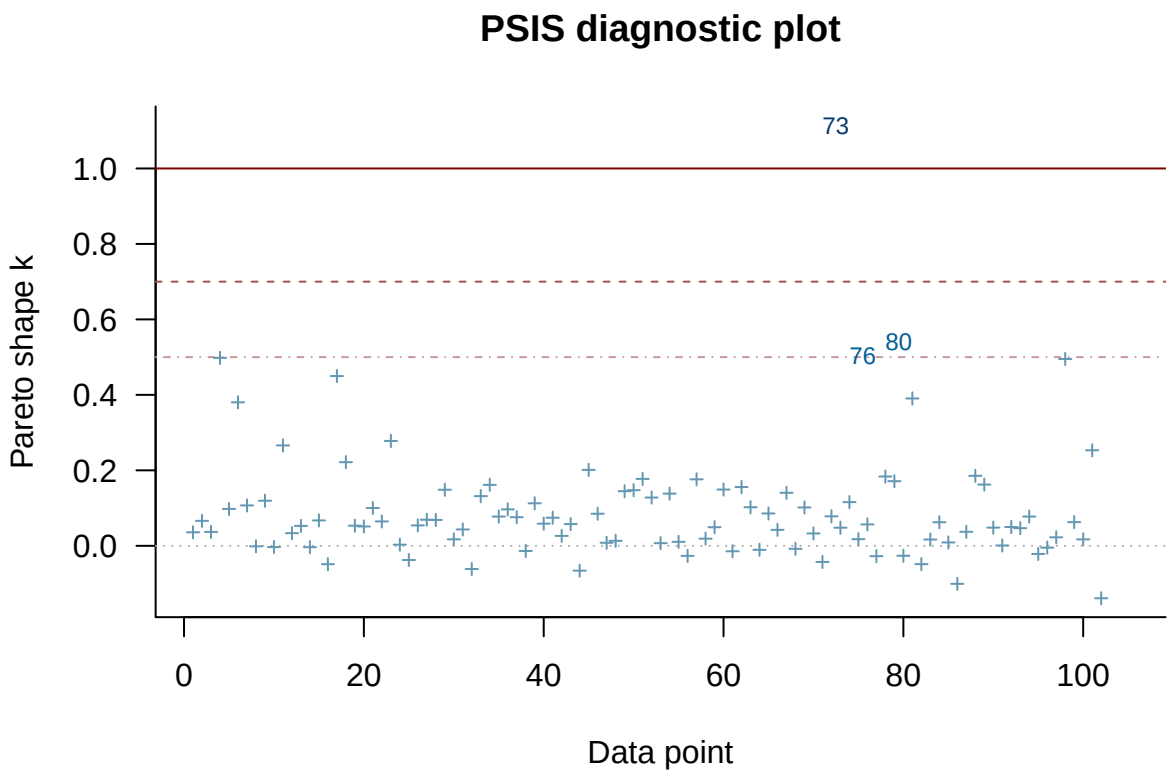
```
hwm_waic <- waic(hwm_log_lik, cores = parallel::detectCores())
```

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

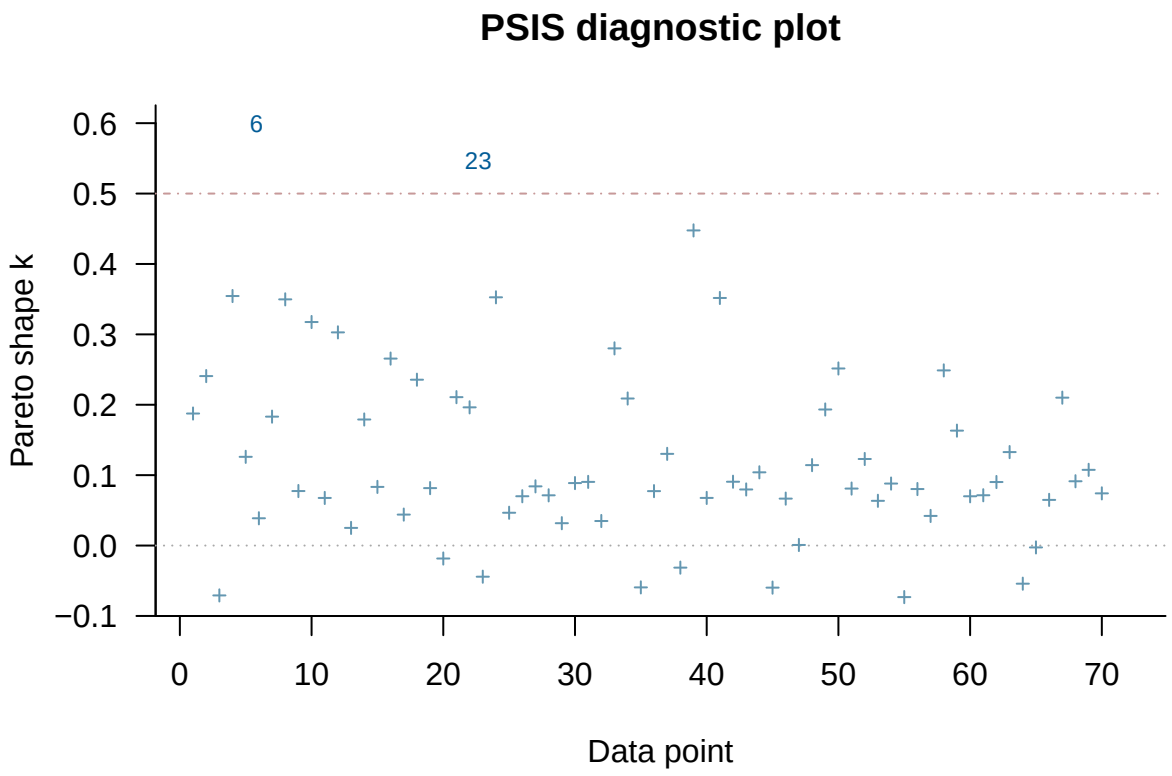
```
# plot pareto k diagnostics for the models
plot(wm_loo, label_points = TRUE)
```




```
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   -632.0 10.0
## p_loo         5.5  0.5
## looic      1264.0 20.0
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                            Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     71    98.6%   9675
##  (0.5, 0.7]   (ok)        0     0.0%   <NA>
##    (0.7, 1]   (bad)       1     1.4%   2071
##    (1, Inf)   (very bad)  0     0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

```
cwm_loo
```

```
##
## Computed from 20000 by 105 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -526.9 31.0
## p_loo        12.8  5.6
## looic      1053.7 61.9
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                            Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     102   97.1%   3060
##  (0.5, 0.7]   (ok)         2    1.9%   2953
##    (0.7, 1]   (bad)        0    0.0%   <NA>
##    (1, Inf)   (very bad)   1    1.0%   6
## See help('pareto-k-diagnostic') for details.
```

```
hwm_loo
```

```
##
## Computed from 20000 by 72 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -485.8  6.1
## p_loo         6.6  1.1
## looic       971.6 12.2
## ------
## 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%   2113
##  (0.5, 0.7]   (ok)        2     2.8%   1830
##    (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   -631.8 10.0
## p_waic         5.3  0.4
## waic        1263.6 20.0
```

```
cwm_waic
```

```
##
## Computed from 20000 by 105 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic   -525.5 30.8
## p_waic        11.5  4.5
## waic        1051.1 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   -485.5  6.0
## p_waic         6.4  1.0
## waic         971.1 12.1
##
## 2 (2.8%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

The best model is..... After her comes the....

3.6 Predictive performance assessment

Elpd_loo is better than, for example, MSE for continuous variable prediction, as it evaluates the whole predictive distribution and not just the mean.

3.7 Prior sensitivity analysis

The sensitivity of posterior inference about θ to the proposed prior distribution has been checked for models. The dataset in that work is not small and posterior inferences based on a large sample are not particularly sensitive to the prior distribution.

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).

Also, maybe it'd be worth to pay more attention to features and apply transformation or even extraction to them. Moreover, the priors can be tested even more accurately, variant like half-cauchy can be tried.

4.2 Things, learnt from the data analysis

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. Last but not least, some functions 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.

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/.

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, 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.