

---

# Lung Cancer Survival Prediction with Bayesian Generalised Linear Models

---



Yann McLatchie, Arina Odnoblyudova

## Contents

1	Introduction	1
2	Description of models	4
3	Diagnostics and performance	16
4	Conclusion	22

## 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, trying 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"))
```

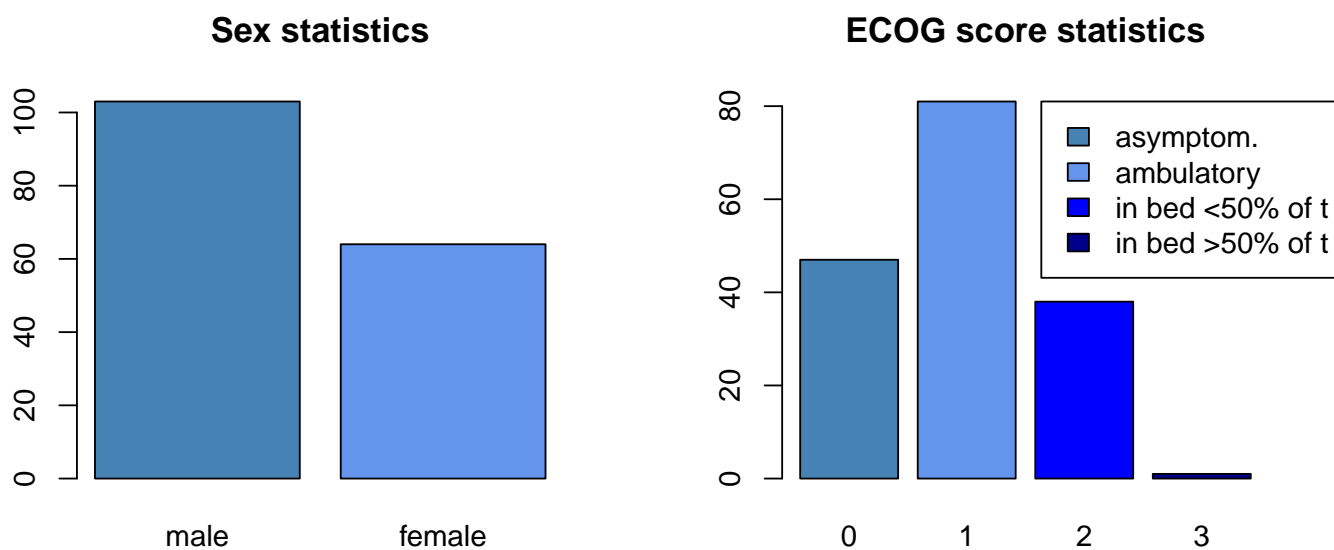


Figure 1: Categorical variables

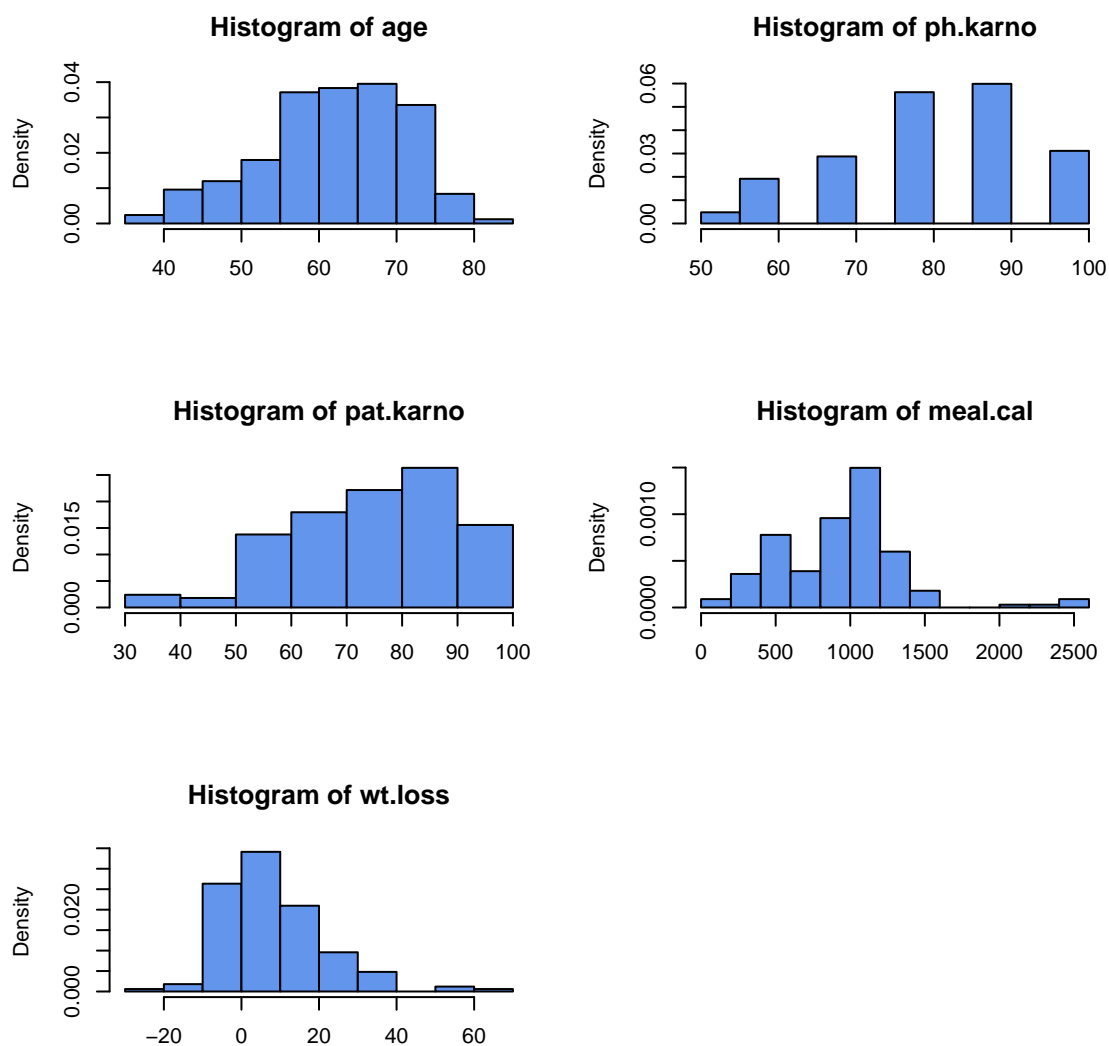


Figure 2: Continuous variables

```
##           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  1.00000000  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)

## -- Attaching packages ----- tidyverse 1.3.1 --
## v tibble  3.1.6      v purrr   0.3.4
## v tidyr   1.1.4      v stringr 1.4.0
## v readr   2.0.2      v forcats 0.5.1

## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(rstan)

## Loading required package: StanHeaders
## rstan (Version 2.21.2, GitRev: 2e1f913d3ca3)

## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)

##
## Attaching package: 'rstan'

## The following object is masked from 'package:tidyr':
##
##      extract
```

```
library(brms)
```

```
## Loading required package: Rcpp
## Loading 'brms' package (version 2.16.1). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').
##
## Attaching package: 'brms'
## The following object is masked _by_ '.GlobalEnv':
##
##      kidney
## The following object is masked from 'package:rstan':
##
##      loo
## The following object is masked from 'package:survival':
##
##      kidney
## The following object is masked from 'package:stats':
##
##      ar
```

```
library(bayesplot)
```

```
## This is bayesplot version 1.8.1
## - Online documentation and vignettes at mc-stan.org/bayesplot
## - bayesplot theme set to bayesplot::theme_default()
##   * Does _not_ affect other ggplot2 plots
##   * See ?bayesplot_theme_set for details on theme setting
```

```
library(loo)
```

```
## This is loo version 2.4.1
## - Online documentation and vignettes at mc-stan.org/loo
## - As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'cores' argument
##
## Attaching package: 'loo'
## The following object is masked from 'package:rstan':
##
##      loo
```

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
## The following object is masked from 'package:tidyr':
##
##      smiths
```

```

library(ggplot2)
library(splines2)
# set number of cores
options(mc.cores = parallel::detectCores())

# read lung cancer data from `survival` library
data("cancer", package = "survival")
# build dataset from only those non-censored data points
uncensored_data <- cancer %>%
  filter(status == 2) %>%
  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)
)

```

## 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  $\alpha$  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  $\eta = \exp(\mathbf{X}\beta)$  so that  $\eta$  is strictly positive. Thus we choose a log link function for our GLM such that

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

### 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

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

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.

```
# 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: c
```

```
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
)

```

## 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  $\alpha$  is the shape parameter, and  $\gamma$  the scale. We move on to define a new variable  $\lambda$  is created, defined in relation to  $\gamma$  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  $\alpha$  and  $\lambda$  follows the equation below, with  $v_i$  an indicator showing 0 if the data are censored and 1 if not,

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

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  $\alpha$  and scale  $\sigma$ , 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  $\beta$  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,  $\text{Gamma}(1, 1)$  is a variant, as the majority of the variance is not too close to zero, it has rather long tail.

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

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

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")

# omitting NAs
data = cancer %>% 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/envs/seurat4/lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/include"
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/fun/include:1:
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: unknown namespace Eigen {
FALSE namespace Eigen {
FALSE ^
FALSE /anaconda3/envs/seurat4/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error: unknown namespace Eigen {
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/fun/include:1:
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' file not found
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 ~ gamma(1,1);
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 = 5000)

```

## 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

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

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 BRMS

```

# build model formula of the form
hw_formula <- formula(paste("time ~ (",

```

```

                                paste0(cov_labels, collapse = " + "),
                                "| inst)))
hw_formula

FALSE time ~ (age + sex + ph.ecog + ph.karno + pat.karno + meal.cal +
FALSE      wt.loss | inst)

# define strong priors over regressors and shape
hw_prior <- c(
  set_prior("normal(0, 1)", class = "sd"),
  set_prior("cauchy(0, 5)", class = "shape")
)
# generate model stan code
make_stancode(hw_formula,
              uncensored_data,
              weibull(link = "log", link_shape = "log"),
              hw_prior)

FALSE // generated with brms 2.16.1
FALSE functions {
FALSE   /* compute correlated group-level effects
FALSE   * Args:
FALSE   *   z: matrix of unscaled group-level effects
FALSE   *   SD: vector of standard deviation parameters
FALSE   *   L: cholesky factor correlation matrix
FALSE   * Returns:
FALSE   *   matrix of scaled group-level effects
FALSE   */
FALSE   matrix scale_r_cor(matrix z, vector SD, matrix L) {
FALSE     // r is stored in another dimension order than z
FALSE     return transpose(diag_pre_multiply(SD, L) * z);
FALSE   }
FALSE }
FALSE data {
FALSE   int<lower=1> N; // total number of observations
FALSE   vector[N] Y; // response variable
FALSE   // data for group-level effects of ID 1
FALSE   int<lower=1> N_1; // number of grouping levels
FALSE   int<lower=1> M_1; // number of coefficients per level
FALSE   int<lower=1> J_1[N]; // grouping indicator per observation
FALSE   // group-level predictor values
FALSE   vector[N] Z_1_1;
FALSE   vector[N] Z_1_2;
FALSE   vector[N] Z_1_3;
FALSE   vector[N] Z_1_4;
FALSE   vector[N] Z_1_5;
FALSE   vector[N] Z_1_6;
FALSE   vector[N] Z_1_7;
FALSE   vector[N] Z_1_8;
FALSE   int<lower=1> NC_1; // number of group-level correlations
FALSE   int prior_only; // should the likelihood be ignored?
FALSE }
FALSE transformed data {
FALSE }

```

```

FALSE parameters {
FALSE   real Intercept; // temporary intercept for centered predictors
FALSE   real<lower=0> shape; // shape parameter
FALSE   vector<lower=0>[M_1] sd_1; // group-level standard deviations
FALSE   matrix[M_1, N_1] z_1; // standardized group-level effects
FALSE   cholesky_factor_corr[M_1] L_1; // cholesky factor of correlation matrix
FALSE }
FALSE transformed parameters {
FALSE   matrix[N_1, M_1] r_1; // actual group-level effects
FALSE   // using vectors speeds up indexing in loops
FALSE   vector[N_1] r_1_1;
FALSE   vector[N_1] r_1_2;
FALSE   vector[N_1] r_1_3;
FALSE   vector[N_1] r_1_4;
FALSE   vector[N_1] r_1_5;
FALSE   vector[N_1] r_1_6;
FALSE   vector[N_1] r_1_7;
FALSE   vector[N_1] r_1_8;
FALSE   // compute actual group-level effects
FALSE   r_1 = scale_r_cor(z_1, sd_1, L_1);
FALSE   r_1_1 = r_1[, 1];
FALSE   r_1_2 = r_1[, 2];
FALSE   r_1_3 = r_1[, 3];
FALSE   r_1_4 = r_1[, 4];
FALSE   r_1_5 = r_1[, 5];
FALSE   r_1_6 = r_1[, 6];
FALSE   r_1_7 = r_1[, 7];
FALSE   r_1_8 = r_1[, 8];
FALSE }
FALSE model {
FALSE   // likelihood including constants
FALSE   if (!prior_only) {
FALSE     // initialize linear predictor term
FALSE     vector[N] mu = Intercept + rep_vector(0.0, N);
FALSE     for (n in 1:N) {
FALSE       // add more terms to the linear predictor
FALSE       mu[n] += r_1_1[J_1[n]] * Z_1_1[n] + r_1_2[J_1[n]] * Z_1_2[n] + r_1_3[J_1[n]] * Z_1_3[n] + r_1_4[J_1[n]] * Z_1_4[n] + r_1_5[J_1[n]] * Z_1_5[n] + r_1_6[J_1[n]] * Z_1_6[n] + r_1_7[J_1[n]] * Z_1_7[n] + r_1_8[J_1[n]] * Z_1_8[n];
FALSE     }
FALSE     for (n in 1:N) {
FALSE       // apply the inverse link function
FALSE       mu[n] = exp(mu[n]) / tgamma(1 + 1 / shape);
FALSE     }
FALSE     target += weibull_lpdf(Y | shape, mu);
FALSE   }
FALSE   // priors including constants
FALSE   target += student_t_lpdf(Intercept | 3, 5.5, 2.5);
FALSE   target += cauchy_lpdf(shape | 0, 5)
FALSE   - 1 * cauchy_lccdf(0 | 0, 5);
FALSE   target += normal_lpdf(sd_1 | 0, 1)
FALSE   - 8 * normal_lccdf(0 | 0, 1);
FALSE   target += std_normal_lpdf(to_vector(z_1));
FALSE   target += lkj_corr_cholesky_lpdf(L_1 | 1);
FALSE }
FALSE generated quantities {

```

```

FALSE // actual population-level intercept
FALSE real b_Intercept = Intercept;
FALSE // compute group-level correlations
FALSE corr_matrix[M_1] Cor_1 = multiply_lower_tri_self_transpose(L_1);
FALSE vector<lower=-1,upper=1>[NC_1] cor_1;
FALSE // extract upper diagonal of correlation matrix
FALSE for (k in 1:M_1) {
FALSE   for (j in 1:(k - 1)) {
FALSE     cor_1[choose(k - 1, 2) + j] = Cor_1[j, k];
FALSE   }
FALSE }
FALSE }
FALSE }

```

```

# fit hierarchical GLM model with BRMS, expliciting the Weibull family
weibull_hier <- brm(
  formula = hw_formula,
  data = uncensored_data,
  prior = hw_prior,
  family = weibull(link = "log", link_shape = "log"),
  iter = 10000,
  cores = parallel::detectCores(),
  control = list(adapt_delta = 0.9) # combat divergent transitions
)

```

```

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 ~
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
FALSE [[1]]
FALSE Stan model 'f12802dd40c47d565ad833124adddd79' does not contain samples.
FALSE
FALSE [[2]]
FALSE Stan model 'f12802dd40c47d565ad833124adddd79' does not contain samples.
FALSE
FALSE [[3]]
FALSE Stan model 'f12802dd40c47d565ad833124adddd79' does not contain samples.

```

### 3 Diagnostics and performance

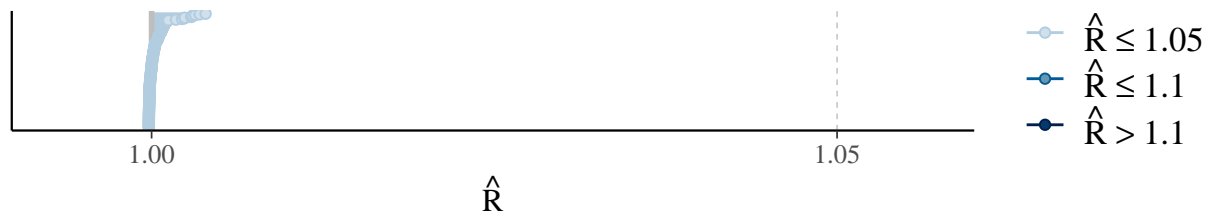
#### 3.1 $\hat{R}$ , effective sample size, and divergences

```
rhat_wm <- mcmc_rhat(rhat = rhat(weibull_model))
rhat_hwm <- mcmc_rhat(rhat = rhat(weibull_hier))
rhat_cwm <- mcmc_rhat(rhat = rhat(weibull_cens))
bayesplot_grid(
  rhat_wm,
  rhat_hwm,
  rhat_cwm,
  titles = c("Pooled model", "Hierarchical model", "Censored model")
)
```

Pooled model



Hierarchical model



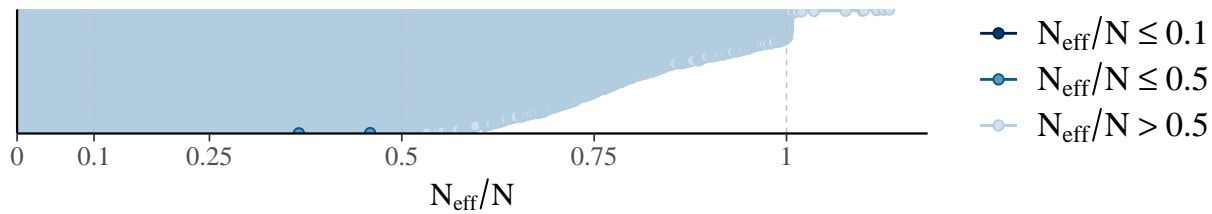
Censored model



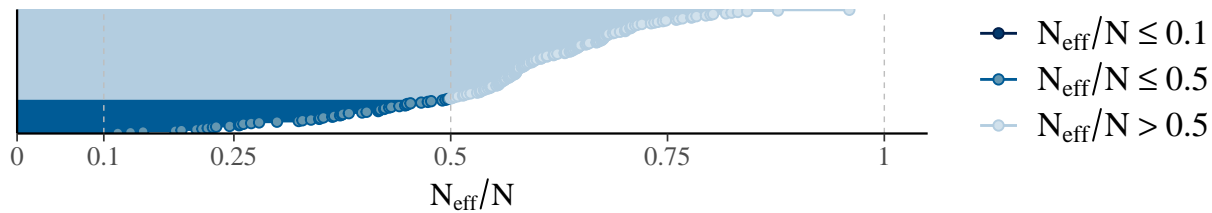
```
bayesplot_grid(
  mcmc_neff(neff_ratio(weibull_model)),
  mcmc_neff(neff_ratio(weibull_hier)),
  mcmc_neff(neff_ratio(weibull_cens)),
  titles = c("Pooled model", "Hierarchical model", "Censored model")
)
```



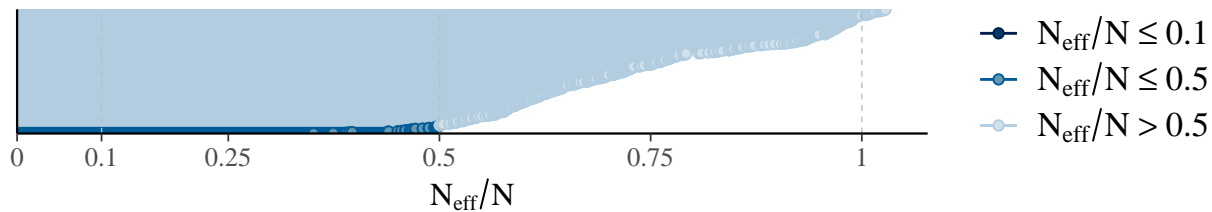
## Pooled model



## Hierarchical model



## Censored model



## 3.2 Posterior predictive checks

## 3.3 Model comparison using leave-one-out cross-validation and WAIC

```
# 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 (2.4%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
# repeat for hierarchical Weibull
```

```
hwm_loo <- loo(weibull_hier, cores = parallel::detectCores())
```

```
## Warning: Found 8 observations with a pareto_k > 0.7 in model 'weibull_hier'. It
## is recommended to set 'moment_match = TRUE' in order to perform moment matching
## for problematic observations.
```

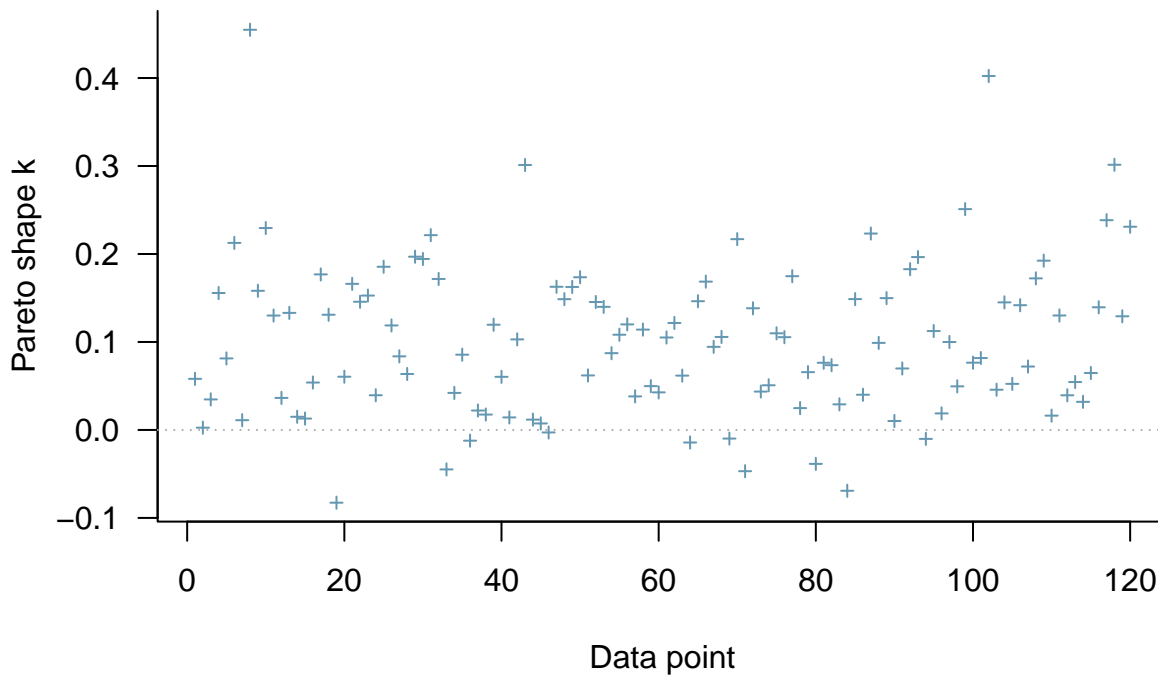
```
hwm_waic <- waic(weibull_hier, cores = parallel::detectCores())
```

```
## Warning:
## 15 (12.5%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

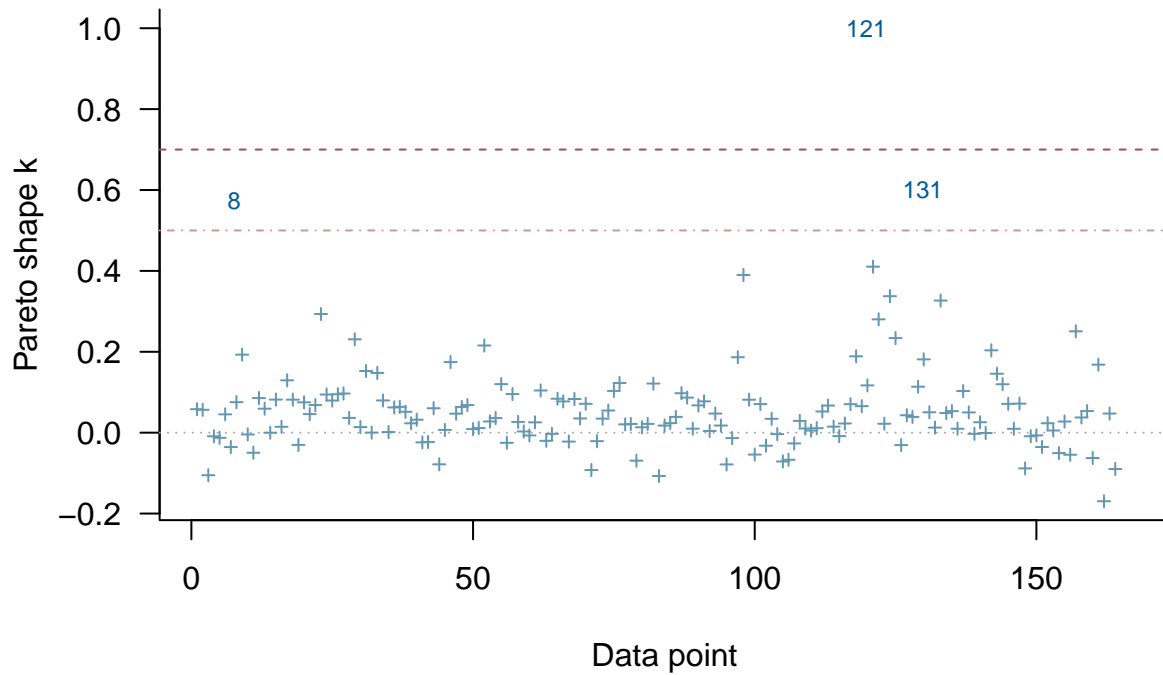
```
# plot pareto k diagnostics for the models
```

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

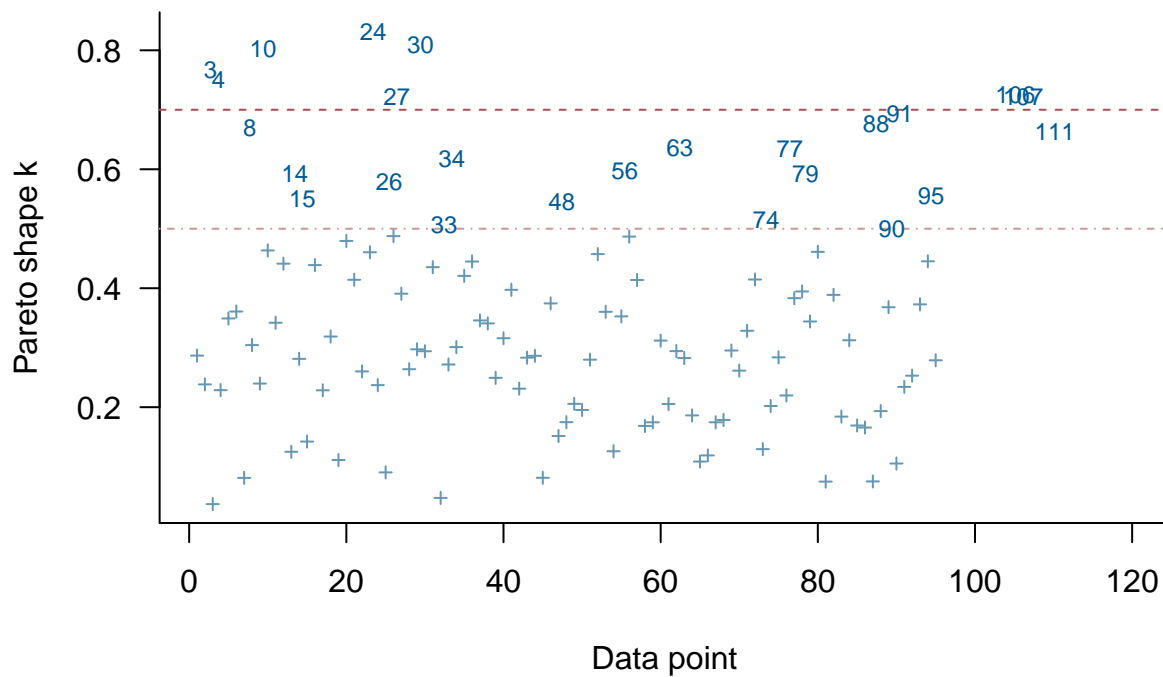
### PSIS diagnostic plot



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

**PSIS diagnostic plot**

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

**PSIS diagnostic plot**

```
# comparison of uncensored models with loo
loo_compare(x = list(wm_loo, hwm_loo))
```

```
## Warning: Not all models have the same y variable. ('yhash' attributes do not
## match)
```

```
##           elpd_diff se_diff
## weibull_hier    0.0      0.0
## model1         -253.6    10.1
```

```
# output loos
```

```
wm_loo
```

```
##
## Computed from 20000 by 120 log-likelihood matrix
##
```

```
##           Estimate   SE
## elpd_loo  -1046.6 12.4
## p_loo       5.2  0.4
## looic      2093.1 24.7
```

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

```
## 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 10000 by 167 log-likelihood matrix
##
```

```
##           Estimate   SE
## elpd_loo  -863.7 37.0
## p_loo      12.4  4.8
## looic     1727.4 74.1
```

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

```
## Monte Carlo SE of elpd_loo is NA.
```

```
##
```

```
## Pareto k diagnostic values:
```

		Count	Pct.	Min. n_eff
## (-Inf, 0.5]	(good)	164	98.2%	2406
## (0.5, 0.7]	(ok)	2	1.2%	1233
## (0.7, 1]	(bad)	1	0.6%	31
## (1, Inf)	(very bad)	0	0.0%	<NA>

```
## See help('pareto-k-diagnostic') for details.
```

```
hwm_loo
```

```
##
## Computed from 5000 by 120 log-likelihood matrix
##
```

```
##           Estimate   SE
## elpd_loo  -792.9  7.6
## p_loo      22.3  2.2
## looic     1585.9 15.2
```

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

```
## Monte Carlo SE of elpd_loo is NA.
```

```
##
```

```
## Pareto k diagnostic values:
```

		Count	Pct.	Min. n_eff
--	--	-------	------	------------

```
## (-Inf, 0.5] (good) 95 79.2% 424
## (0.5, 0.7] (ok) 17 14.2% 302
## (0.7, 1] (bad) 8 6.7% 142
## (1, Inf) (very bad) 0 0.0% <NA>
## See help('pareto-k-diagnostic') for details.

# comparison of uncensored models with waic
loo_compare(x = list(wm_waic, hwm_waic))

## Warning: Not all models have the same y variable. ('yhash' attributes do not
## match)

##          elpd_diff se_diff
## weibull_hier 0.0 0.0
## model1 -255.5 10.0

# output waics
wm_waic

##
## Computed from 20000 by 120 log-likelihood matrix
##
##          Estimate SE
## elpd_waic -1046.5 12.4
## p_waic 5.2 0.3
## waic 2093.0 24.7

cwm_waic

##
## Computed from 10000 by 167 log-likelihood matrix
##
##          Estimate SE
## elpd_waic -863.0 36.9
## p_waic 11.8 4.3
## waic 1726.1 73.9
##
## 4 (2.4%) p_waic estimates greater than 0.4. We recommend trying loo instead.

hwm_waic

##
## Computed from 5000 by 120 log-likelihood matrix
##
##          Estimate SE
## elpd_waic -791.0 7.3
## p_waic 20.4 1.9
## waic 1582.0 14.7
##
## 15 (12.5%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

## 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. Maybe in the future it'd be better 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 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.
- 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.