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. There are some institutional groups, where the number of patiens observed is rather small. They are excluded from dataset.

```
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.

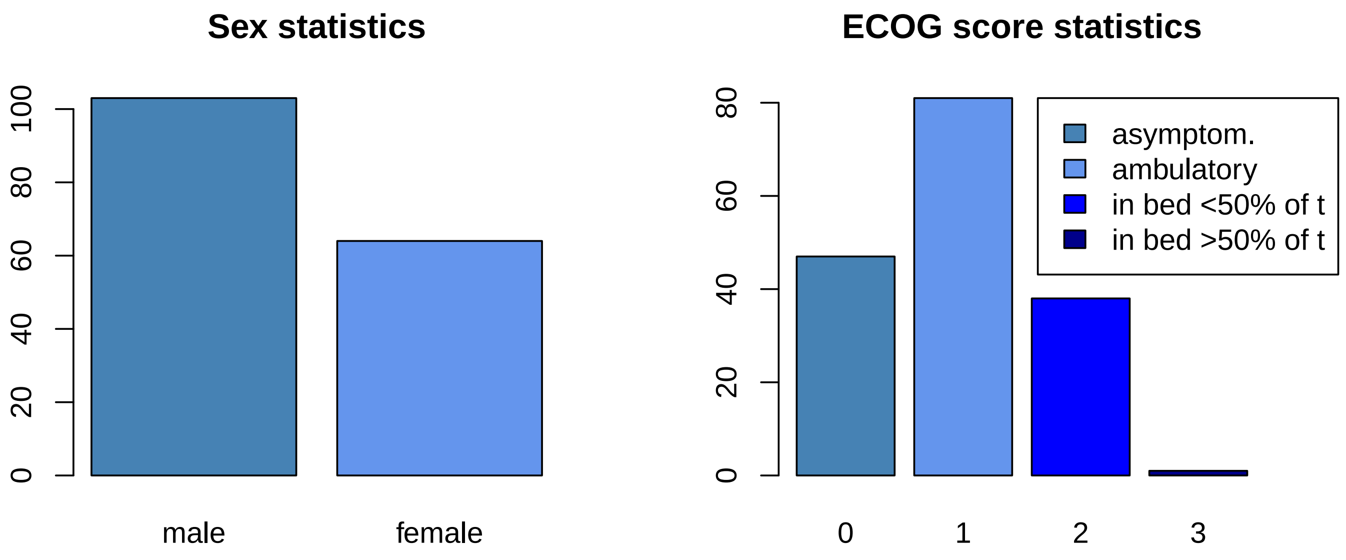


Figure 1: Categorical variables

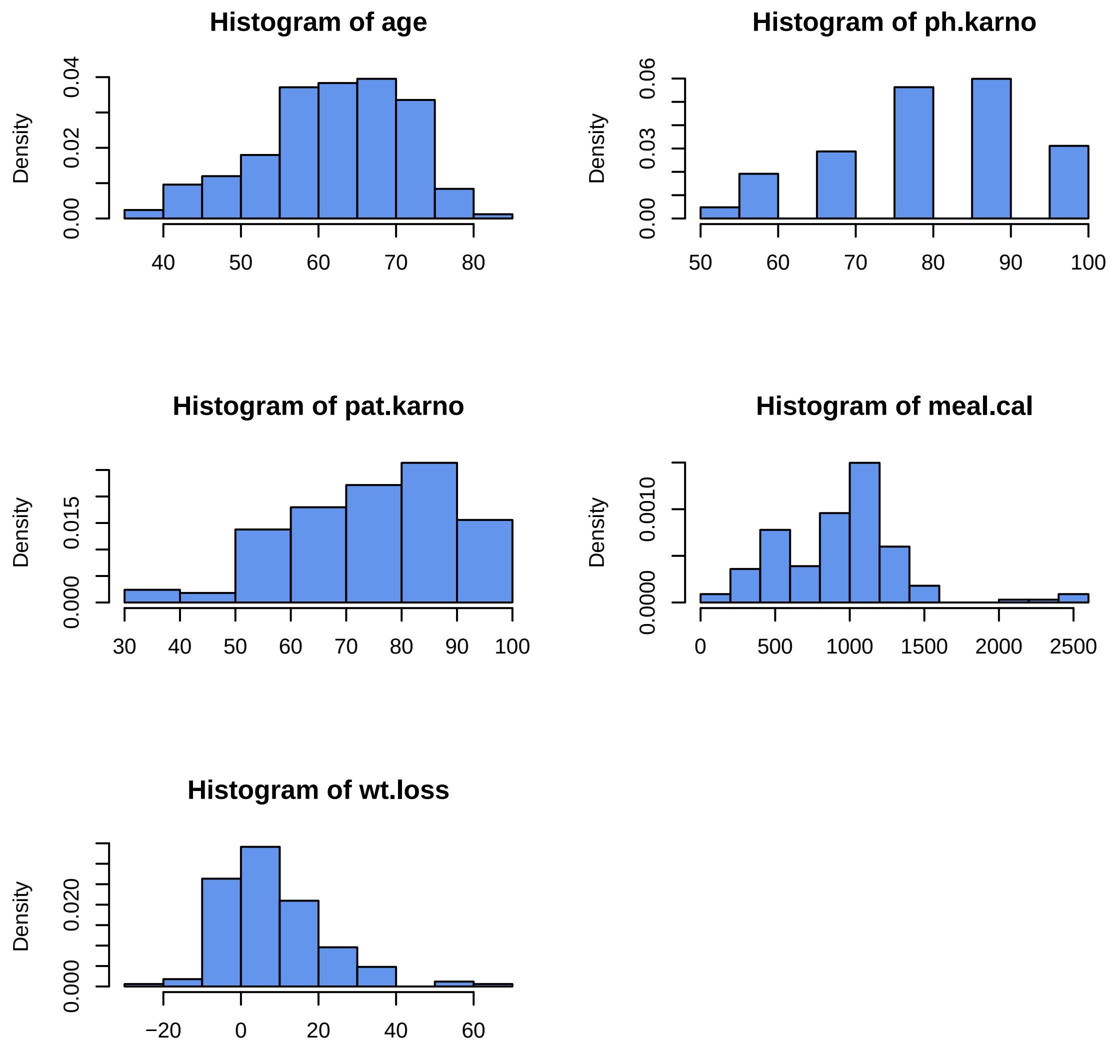


Figure 2: Continuous variables

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

```
##                    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(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

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

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")
```

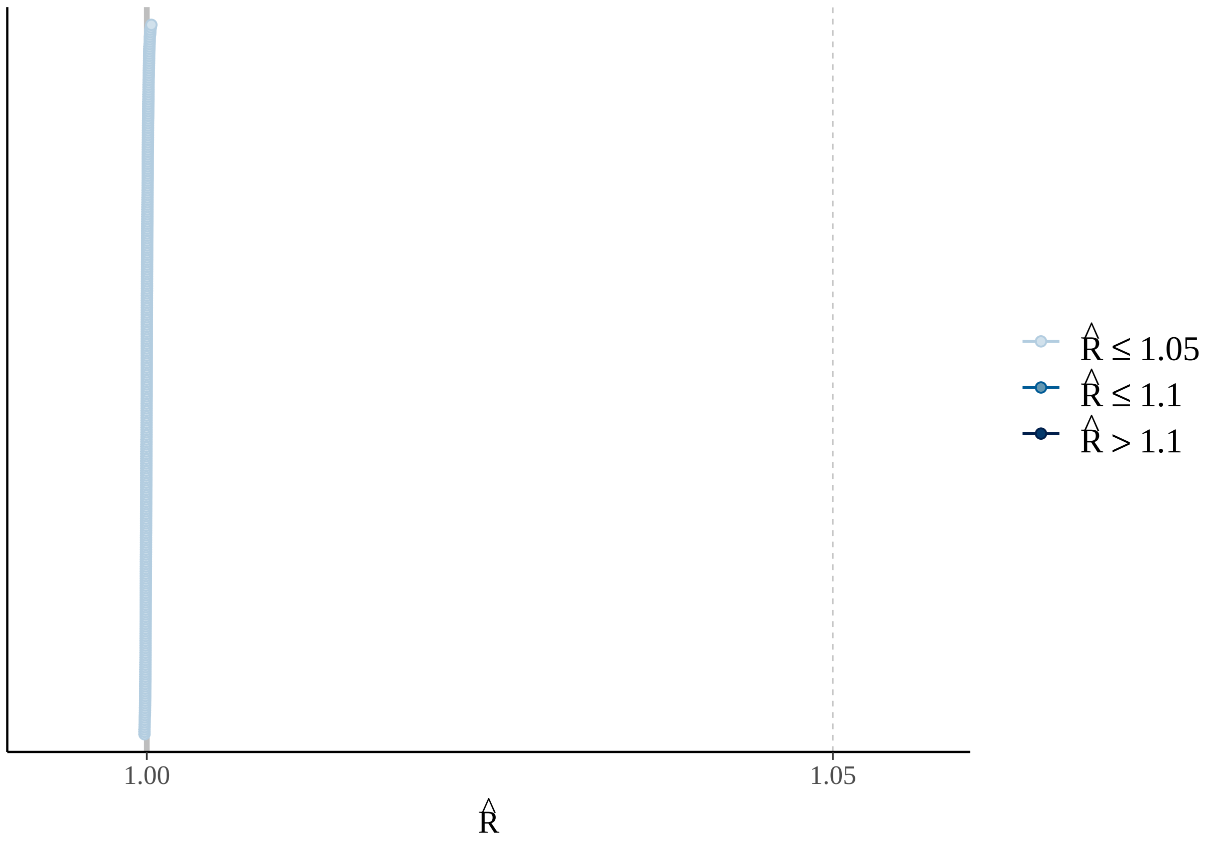
```
## Running /anaconda3/envs/seurat4/lib/R/bin/R CMD SHLIB foo.c
## x86_64-apple-darwin13.4.0-clang -I"/anaconda3/envs/seurat4/lib/R/include" -DNDEBUG   -I"/anaconda3/envs
## In file included from <built-in>:1:
## In file included from /anaconda3/envs/seurat4/lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/
## 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)
mcmc_rhat(rhat = rhat(weibull_model))
```



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

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".

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

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

Running model

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

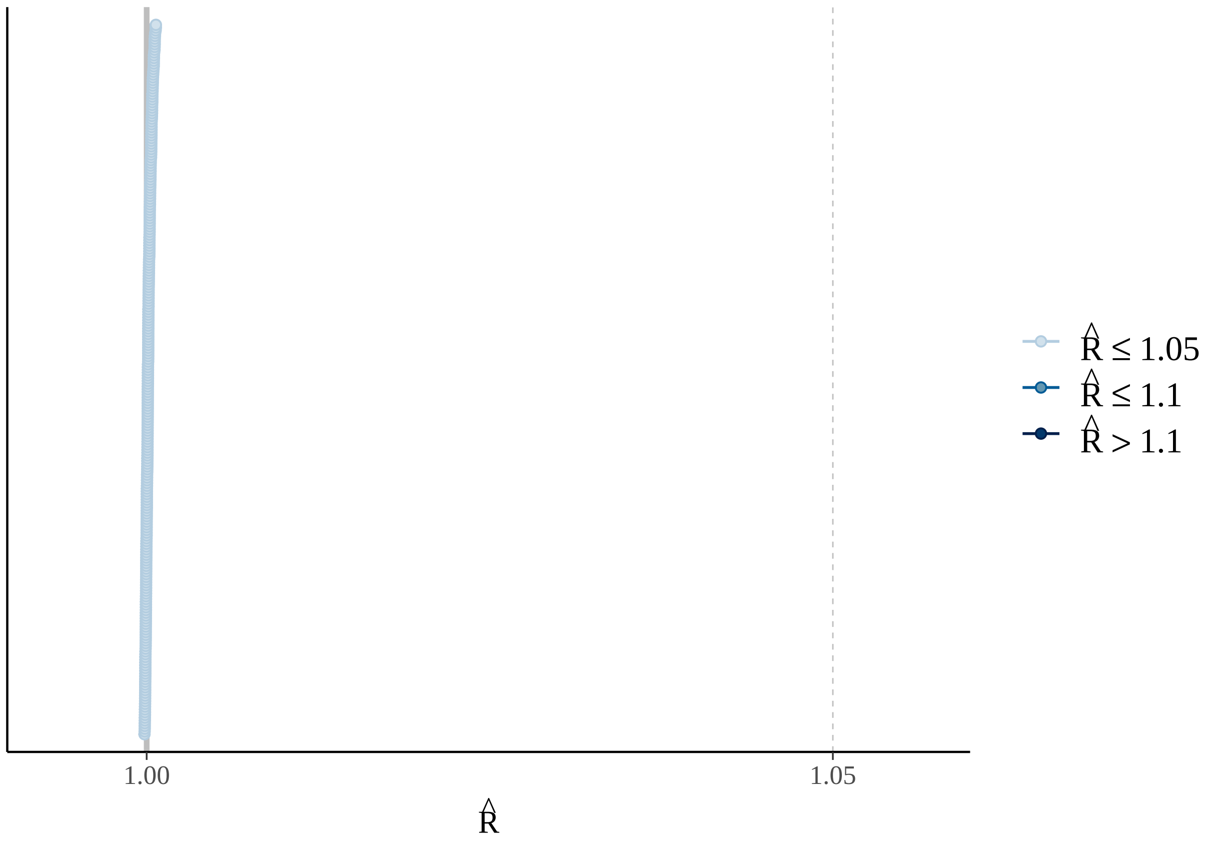
```
## Running /anaconda3/envs/seurat4/lib/R/bin/R CMD SHLIB foo.c
## x86_64-apple-darwin13.4.0-clang -I"/anaconda3/envs/seurat4/lib/R/include" -DNDEBUG   -I"/anaconda3/envs
## In file included from <built-in>:1:
## In file included from /anaconda3/envs/seurat4/lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/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

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

```
## S4 class stanmodel 'weibull_censored' coded as follows:
## data {
##   int<lower=0> N;    // number of object
##   int<lower=0> M;    // number of features
##   int<lower=0> Ncen;    // number of object
##   vector<lower=0>[N] yobs; // target-survival time
##   matrix[N, M] Xobs;    // covariates
##   vector<lower=0>[Ncen] ycen; // target-survival time
##   matrix[Ncen, M] Xcen;    // covariates
## }
##
## parameters {
##   vector[M] beta;       // regressors
##   real<lower=0> alpha;  // shape parameter
## }
```

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

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



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

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

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 Stan

```r
# 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
```

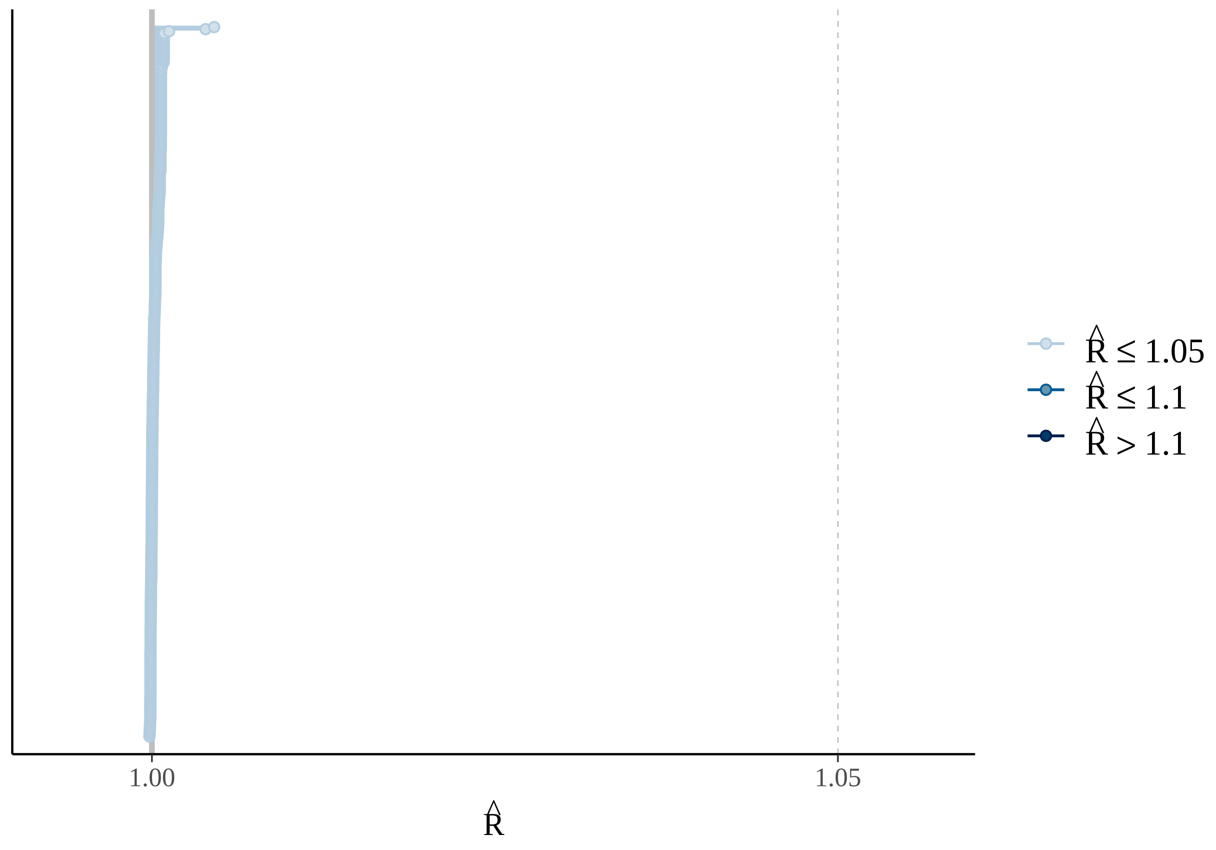
```r
# print out Stan code
print(whm)
```

```
## S4 class stanmodel 'weibull_hier' coded as follows:
```

```
## data {
##   int<lower=0> N;               // number of object
##   int<lower=0> M;               // number of features
##   int<lower =0> J;              // number of institutions
##   vector<lower=0>[N] y;         // target-survival time
##   matrix[N, M] X;               // covariates
##   int<lower=1, upper=J> ll[N];  // instution labels
## }
##
## transformed data {
##   matrix[N, M] Xc;  // centered version of X without an intercept
##   vector[M] means_X;  // column means of X before centering
##
##   // column-center the design matrix for fitting the model
##   for (m in 1:M) {
##     means_X[m] = mean(X[, m]);
##     Xc[, m] = X[, m] - means_X[m];
##   }
## }
##
## parameters {
##   // hyperpriors
##   real mu_beta;
##   real sigma_beta;
##
##   // regressors
##   matrix[M, J] beta;            // regressors weights for different institutions
##   real<lower=0> alpha;          // shape parameter
## }
##
## transformed parameters {
##   vector[N] eta;
##   vector<lower=0>[N] sigma;
##   // compute latent predictor term
##   for (n in 1:N) {
##     eta[n] = Xc[ll[n], ] * beta[, ll[n]];
##   }
##   // apply the log inverse link function
##   sigma = exp(-eta / alpha);
## }
##
## model {
##   // hyperpriors
##   mu_beta ~ normal(0, 1);
##   sigma_beta ~ gamma(1, 1);
##
##   // prior over regressor and shape parameters
##   for (j in 1:J) {
##     beta[, j] ~ normal(mu_beta, sigma_beta);
##   }
##   alpha ~ cauchy(0, 5);
##
##   // fitting model
##   for (n in 1:N) {
```

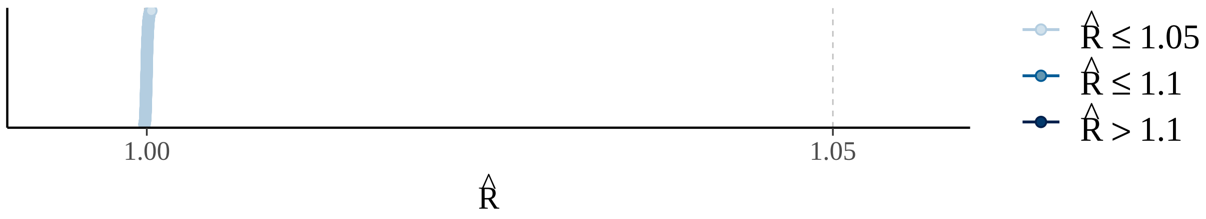
```
##     y[n] ~ weibull(alpha, sigma[n]);
##   }
## }
##
## generated quantities {
##   // define quantities
##   real ypred[N];
##   vector[N] log_lik;
##
##   // compute quantities
##   for (n in 1:N) {
##     // predictive distribution for survival time
##     ypred[n] = weibull_rng(alpha, sigma[n]);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

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

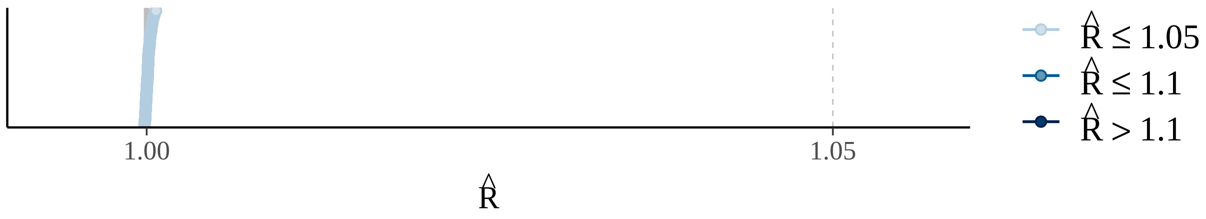


```
bayesplot_grid(
  mcmc_rhat(rhat = rhat(weibull_model)),
  mcmc_rhat(rhat = rhat(weibull_cens)),
  mcmc_rhat(rhat = rhat(weibull_hier)),
  titles = c(
    "Pooled model",
    "Censored model",
    "Hierarchical model"
  )
)
```

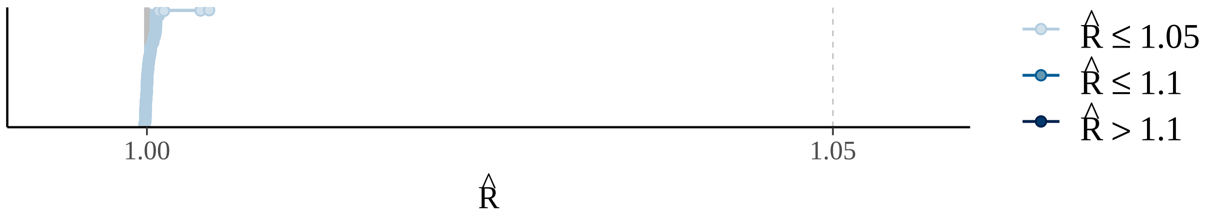
Pooled model



Censored model



Hierarchical model



3 Conclusion

3.1 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
  vector[M] beta[J];            // 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
  vector[M] beta_unif[J];       // 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].

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.

3.2 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_loo, waic parameters and PSIS diagnostic plot.

3.3 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.

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.

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.