Missing and censored data in Bayesian models: practical exercises

Christopher Jackson

The practical exercises for Session 6 "Modelling missing and censored data" are provided in this PDF file.

1. Missing covariate data.
2. Survival modelling.

Full worked solutions are provided at the back of this document. Feel free to consult these as you are going along, if you are stuck on any of the exercises.

The document and embedded code are also provided as an R Markdown document in `missingcens_practical.Rmd`. This allows the R code blocks to be run conveniently from RStudio. If you want to copy-and-paste code from the practicals into your own documents, then this should be copied from the R Markdown document rather from the PDF, otherwise some special characters will fail to copy properly.

1 Missing covariate data

```
library(tidyverse)
library(rjags)
library(posterior)
ELSAmiss <- read.csv("../03_regression/ELSA.csv") %>%
  slice(1:100)
```

The logistic regression example from Session 3 (regression) will be extended to include the observations where some covariates are missing.

The outcome is diabetes status at the next 4-yearly visit (`diab_4yr`), and the covariates are current age (`age`) and current (total) cholesterol level (`chol_curr`). The data are subsetted to the first 100 observations (for computational convenience in this example). Cholesterol is missing in 25% of cases, whereas age is fully observed.

(a) Modify the example in session 3 to include a fully Bayesian missing data model.

Alongside the model of interest for predicting diabetes, this should include a model with current cholesterol as an outcome, from which missing values for cholesterol can be predicted given the observed values of other variables. Include age and gender (`male`) as predictors in the model for cholesterol. Vague priors are OK for the purpose of this example.

(b) Compare the key results of the model of interest with and without a missing data model. Are the results similar - has the missing data model contributed any useful information? Explain the reasons for this.

Note: to answer this question, the same JAGS code can be used, but with two different datasets, with and without the cases with missing cholesterol included.

(c) How might we extend the model to include another covariate with intermittently-missing values (e.g. systolic blood pressure) in the model of interest, while including information from the cases where current cholesterol is missing, but not blood pressure? There is no need to actually implement this model, just sketch how it might be constructed.

(Hint: as discussed in the lectures, in a fully Bayesian model, you should not include a regression of X on Y and a regression of Y on X in the same model, for *any* variables Y and X)

1.1 Multiple imputation: demonstration

The following code demonstrates how to do multiple imputation of missing data where the model of interest is Bayesian. This is a more computationally efficient alternative to fully Bayesian modelling of missing data.

There are no exercises to do here - this is just included so you have a working example of this method.

1. Fit the imputation model. In our example, this is a simple non-Bayesian linear regression for current cholesterol in terms of age and gender.

```
imp.lm <- lm(chol_curr ~ age + male, data=ELSAmiss)
mischol <- ELSAmiss[is.na(ELSAmiss$chol_curr),]
mean_imp <- predict(imp.lm, newdata = mischol)
sd_imp <- summary(imp.lm)$sigma
```

2. Create 10 imputed datasets by predicting the missing cholesterol values from the imputation model.

```
nimp <- 10
impdata <- vector(nimp, mode="list")
set.seed(1)
for (i in 1:nimp){
  impdata[[i]] <- ELSAmiss %>% select(chol_4yr, chol_curr, age, male)
  chol_imp <- rnorm(mean_imp, sd_imp)
  impdata[[i]]$chol_curr[is.na(ELSAmiss$chol_curr)] <- chol_imp
}
```

3. Define the model of interest in JAGS code, and fit it to each of the 10 imputed datasets, using a `for` loop in R to repeatedly call JAGS.

```
moi_mod <- "
model {
  for (i in 1:n){
    ## Model of interest: predict cholesterol at next 4 year visit, based on
    ## current cholesterol, age and gender
    chol_4yr[i] ~ dnorm(mu_cn[i], prec_cn)
    mu_cn[i] <- acn + bcnmale*male[i] + bcnage*age[i] + bcnchol*chol_curr[i]
  }
  acn ~ dnorm(5, 0.1)
  bcnmale ~ dnorm(0, 1)
  bcnage ~ dnorm(0, 0.01)
  bcnchol ~ dnorm(0, 1)
  prec_cn ~ dgamma(0.001, 0.001)
}
"

draws_allimps <- vector(nimp, mode="list")
for (i in 1:nimp){
  dat <- as.list(impdata[[i]])
  dat$chol_curr <- dat$chol_curr - mean(dat$chol_curr)
  # be careful here - mean() doesn't automatically exclude NAs,
  ## so any NAs should have been imputed or excluded before calling mean()
  dat$age <- dat$age - mean(dat$age)
  dat$n <- length(dat$chol_4yr)
  moi.jag <- jags.model(textConnection(moi_mod), dat, quiet = TRUE)
  vars <- c("bcnchol","bcnage","bcnmale")
  sam <- coda.samples(moi.jag, var=vars, n.iter=1000)
  draws_allimps[[i]] <- subset_draws(as_draws_df(sam), iteration = 101:1000)
```

```
}
```

```
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
```

4. Pool the posterior samples from all imputations. There are multiple ways to do this in R. Clearest way, but involves most code:

```
draws <- draws_allimps[[1]]
for (i in 2:10){
      draws <- bind_draws(draws, draws_allimps[[i]], along="iteration")
}
```

   Or a one-line call using base R,

```
draws <- do.call(function(...) bind_draws(..., along="iteration"), draws_allimps)
```

   Or a slightly neater one-liner which uses a tidyverse function, `exec` from the `rlang` package.

```
draws <- exec("bind_draws", !!!draws_allimps, along="iteration")
```

   The object `draws` can then be summarised as if it were a sample from a Bayesian posterior distribution - though it is an approximation to a full Bayesian model.

```
summary(draws)
```

```
## # A tibble: 3 x 10
##   variable    mean  median     sd    mad      q5      q95  rhat
##   <chr>      <num>   <num>  <num>  <num>   <num>    <num> <num>
## 1 bcnage   -0.0391 -0.0390 0.0183 0.0182 -0.0694 -0.00943  1.00
## 2 bcnchol   0.139   0.139  0.0756 0.0730  0.0154  0.264    1.02
## 3 bcnmale  -0.208  -0.201  0.330  0.327  -0.752   0.324    1.00
## # i 2 more variables: ess_bulk <num>, ess_tail <num>
```

2 Survival models in JAGS

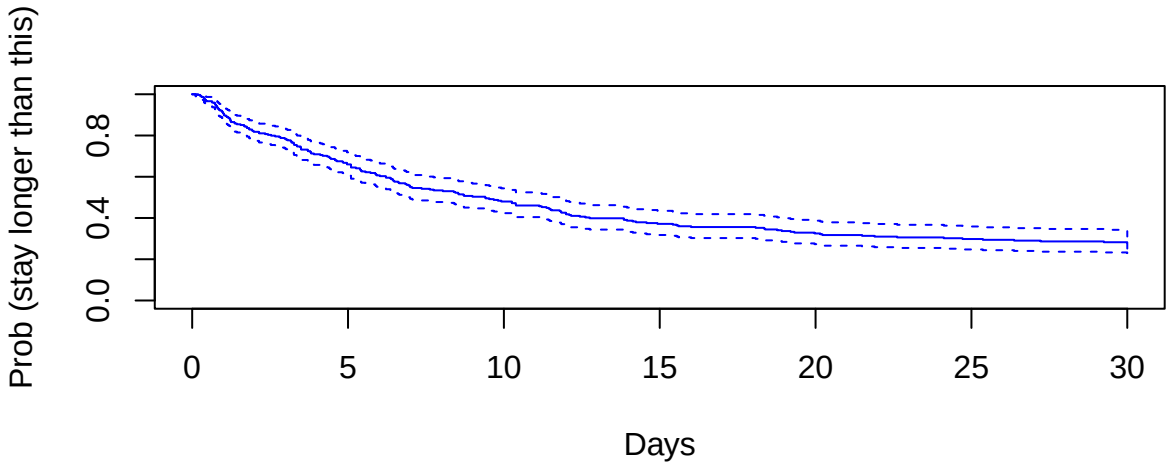
In this practical exercise, the JAGS syntax for a parametric survival model is demonstrated. You will then modify the JAGS code to answer some practical questions.

The example is based on a simulated dataset `hosp`, based on a real example of lengths of stay in hospital for COVID-19. There are 300 observations. The variables are `days` (number of days in hospital) and `event` (1 if the person was discharged from hospital at that day, and 0 for a censored time to discharge). The first four rows of the data, and a Kaplan-Meier estimate of the distribution of length of stay in hospital, are shown below.

```
library(survival)
hosp <- read.csv("hosp_surv_sim.csv")
head(hosp, 4)
```

```
##        days event
## 1  2.167328     0
## 2 30.000000     0
## 3 30.000000     0
## 4 30.000000     0
```

```
fit <- survfit(Surv(days, event) ~ 1, data=hosp)
plot(fit, col="blue", xlab="Days",
     ylab="Prob (stay longer than this)")
```



See, e.g. `help(plot.survfit)`, `help(lines.survfit)`, `help(survfit)` for customising Kaplan-Meier estimates and how these are plotted, e.g. how to change the colour, line type and whether confidence intervals are plotted. Multiple curves on the same plot can be produced, e.g. with `lines()`, or by replacing `~1` in the example above with the name of a factor covariate - see the examples in the help pages.

Note the original analysis also accounted for the competing risk of death before discharge from hospital, but competing risks are not covered in this course. For the sake of this exercise, suppose that these are standard censored time-to-event data, so that `event` equal to 0 means that the length of stay in hospital is known to be longer than the corresponding time in `days`.

2.1 JAGS syntax for censored data

JAGS has an unusual syntax for censored data, shown in the following model code.

- The vector `y` includes observed survival times for those that were observed, and `NA` for those that are censored. This is given a parametric distribution, which in this example is a Weibull (`dweib`).
- `c` includes censoring times for those that are censored, and `Inf` for the observed survival times

- `is.censored` is 1 if the outcome is censored and 0 otherwise
- Initial values are supplied for the unobserved survival times (the `y` component of `ini` below). A reasonable way to define these is to add one day to the censoring time.

Do not worry about why this strange JAGS syntax is the way it is. If you are interested, then the JAGS user manual (at https://sourceforge.net/projects/mcmc-jags/files/Manuals/) explains how the `dinterval` distribution is defined, but you are not required to understand it for this course. The important thing is to be able to implement it.

```
surv_jagsmod <- "
model {
 for (i in 1:n) {
  y[i] ~ dweib(alpha, lambda)            # NA for censored observations
  is.censored[i] ~ dinterval(y[i], c[i]) # observation is censored if
                                         # y[i] is greater than c[i]
 }
 lambda <- 1 / exp(loginvlam)
 alpha ~ dgamma(1, 1)    # priors for Weibull parameters
                         # discussed in lecture
 loginvlam  ~ dnorm(log(10), (1.96/(log(30) - log(10)))^2)
}"
y <- hosp$days; y[hosp$event==0] <- NA
c <- hosp$days; c[hosp$event==1] <- Inf
dat <- list(y = y, c = c,
            is.censored = 1 - hosp$event,
            n=nrow(hosp))
ini <- list(y = ifelse(is.na(y), c+1, NA),
            alpha = 1, loginvlam = 0)
```

The model above can then be run, checked for convergence and summarised in the standard way (note just one chain is run in this demonstration, but it is usually sensible in practice to run two parallel chains, to help with convergence checking).

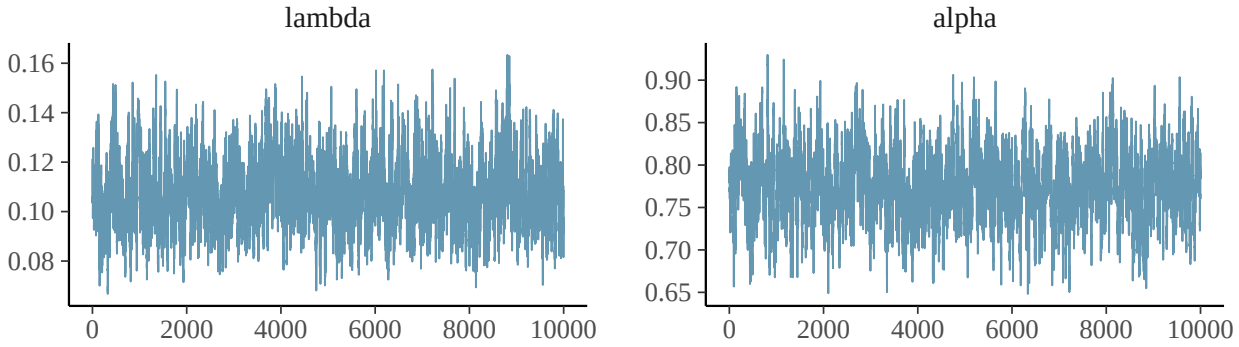
```
library(rjags)
library(bayesplot)
library(posterior)
surv.jag <- jags.model(textConnection(surv_jagsmod), dat, inits=ini, quiet = TRUE)
update(surv.jag, 1000)
```

```
##   |                                                              |
```

```
sam <- coda.samples(surv.jag, var=c("lambda","alpha"), n.iter=10000)
```

```
##   |                                                              |
```

```
mcmc_trace(sam, c("lambda","alpha"))
```




```
summary(sam)
```

```
##
## Iterations = 2001:12000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean      SD  Naive SE Time-series SE
## alpha  0.7751 0.04175 0.0004175      0.0018812
## lambda 0.1069 0.01438 0.0001438      0.0006128
##
## 2. Quantiles for each variable:
##
##           2.5%    25%    50%    75%  97.5%
## alpha  0.69447 0.7465 0.7754 0.8033 0.8570
## lambda 0.08117 0.0969 0.1058 0.1162 0.1376
```

In this Weibull model, the hazard at time `t` is $\lambda\alpha t^{\alpha-1}$. Therefore it could be extended to a proportional hazards regression model, by replacing `lambda` by an individual-specific variable `lambda[i]`, and expressing `log(lambda[i])` as a linear function of covariates, as in a standard regression model. The coefficients of these covariates will be log hazard ratios.

2.2 Exercise

(a) Suppose we expect 500 people to be admitted to hospital in the next month. Modify the JAGS code to calculate the expected total number of days spent in hospital for this population. To do this, note the following:

- The mean of the Weibull distribution with shape α and scale λ (in the parameterisation used by JAGS) is $\lambda^{-1/\alpha}Gamma(1+1/\alpha)$.
- The log of the Gamma function in JAGS is called `loggam()`.

(b) Modify the JAGS code to calculate the probability that an individual stays in hospital for longer than 20 days.

Hint: the cumulative distribution function of the Weibull is available in JAGS as `pweib(t, alpha, lambda)` - see Table 6.2 in the JAGS user manual.

Then run the model to calculate the posterior distributions of the quantities defined in (a) and (b). Calculate the posterior probability that over 15000 hospital bed-days are required in the next month.

(c) (advanced!) The maximum follow-up in the `hosp` dataset is 30 days. Suppose we want to supply prior information about the probabilities of longer hospital stays. One way we can do this is as follows. Suppose our prior probability of being still in hospital by 50 days is 0.01 (lower credible limit 0.001, upper credible limit 0.05). The `fitdist` function from the `SHELF` package (described in the first lecture) can be used to find a Beta distribution which is roughly compatible with this belief, as follows.

```
betas <- SHELF::fitdist(vals=c(0.001, 0.01, 0.05), probs=c(0.025, 0.5, 0.975),
                        lower=0, upper=1)$Beta
prior_y <- round(betas$shape1 + 1)
prior_n <- round(betas$shape1 + betas$shape2 + 2)
```

The information in this Beta distribution is converted to "pseudo-data" consisting of `prior_y` $= 2$ people out of `prior_n` $= 112$ still in hospital by 50 days, using the Beta/Binomial relation explained in lectures 1-2.

- Use this "pseudo-data" in the JAGS model to provide an additional source of information about α and λ, by modelling it with a binomial distribution with the appropriate probability.
- Compare the estimates of $P(T > 50)$, the probability of being still in hospital at 50 days, with and without this prior information.
- Check that the difference in the results makes sense, given the information that was provided.
- As well as the prior distribution and the original observed data, what is the third major piece of information or assumption which influences the estimate of $P(T > 50)$ in this example?

Solutions: missing covariate data

(a) Examples of JAGS code for missing data model

```
lr_miss_jagsmod <- "model {
for (i in 1:n) {
  # Model of interest
  diab_4yr[i] ~ dbern(p[i])
  logit(p[i]) <- alpha + beta_age*(age[i] - 50)/10 + beta_chol*chol_curr[i]

  # Covariate model: for predicting missing values of current cholesterol
  chol_mu[i] <- ac + bc*(age[i] - mean(age))/10 + cc*male[i]
  chol_curr[i] ~ dnorm(chol_mu[i], cprec)
}

 # Priors
 alpha ~ dlogis(0, 1)
 beta_age   ~ dnorm(0, 1/2.5^2)
 beta_chol  ~ dnorm(0, 1/2.5^2)
 or_age <- exp(beta_age)
 or_chol <- exp(beta_chol)
 ac ~ dlogis(0, 1)
 bc ~ dnorm(0, 1/2.5^2)
 cc ~ dnorm(0, 1/2.5^2)
 cprec ~ dgamma(0.001, 0.001)
}"
```

(b)

```
library(rjags)
library(bayesplot)
library(posterior)
ini <- list(list(alpha=0, beta_age=0, beta_chol=0, ac=0, bc=0, cc=0, cprec=1),
        list(alpha=1, beta_age=1, beta_chol=1, ac=1, bc=1, cc=1, cprec=2))
```

Create two different datasets: one with cases included even if cholesterol is missing,

```
dat <- list(diab_4yr = ELSAmiss$diab_4yr,
        age = ELSAmiss$age, male = ELSAmiss$male,
        chol_curr = ELSAmiss$chol_curr, n = nrow(ELSAmiss))
```

. . . and one where the cases with missing cholesterol are excluded

```
ELSAnomiss <- ELSAmiss %>% filter(!is.na(chol_curr))
datn <- list(diab_4yr = ELSAnomiss$diab_4yr,
         age = ELSAnomiss$age, male = ELSAnomiss$male,
         chol_curr = ELSAnomiss$chol_curr, n = nrow(ELSAnomiss))
```

Age and gender are fully observed in both of these datasets.

Now run JAGS, check convergence, and summarise the parameters of interest with both of these datasets. There should be enough iterations run that the comparison between the two analyses is not confused by Monte Carlo error.

```
diab_jag <- jags.model(textConnection(lr_miss_jagsmod),
                   data=dat, inits=ini, n.chains = 2)
```

```
## Compiling model graph
##    Resolving undeclared variables
```

```
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 172
##    Unobserved stochastic nodes: 35
##    Total graph size: 961
##
## Initializing model
##
##   |                                                              |
```

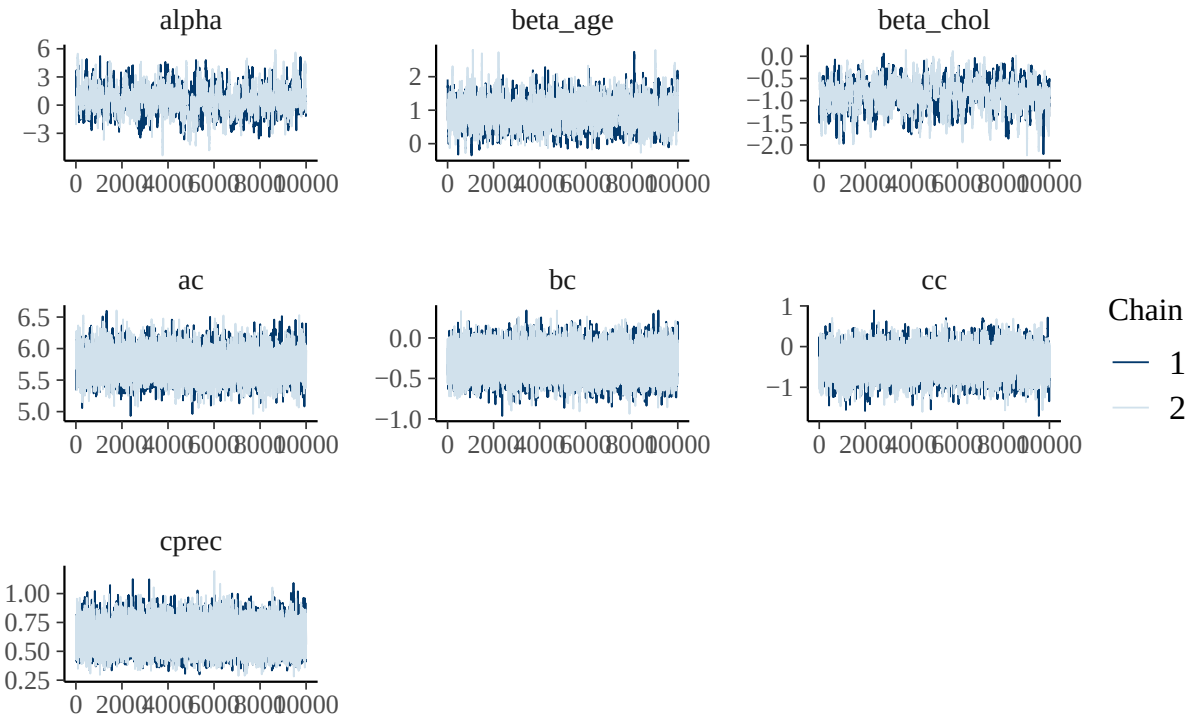
```
update(diab_jag, 1000)
```

```
##   |                                                              |
```

```
sam <- coda.samples(diab_jag,
                c("alpha","or_age", "or_chol","beta_age","beta_chol",
                  "ac","bc","cc", "cprec"), n.iter=10000)
```

```
##   |                                                              |
```

```
mcmc_trace(sam, c("alpha","beta_age", "beta_chol","ac","bc","cc", "cprec"))
```




```
draws <- as_draws(sam)
```

```
diabn_jag <- jags.model(textConnection(lr_miss_jagsmod), data=datn, inits=ini, n.chains = 2)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 144
##    Unobserved stochastic nodes: 7
##    Total graph size: 733
##
```

```
## Initializing model
##
##   |                                                                      |
```

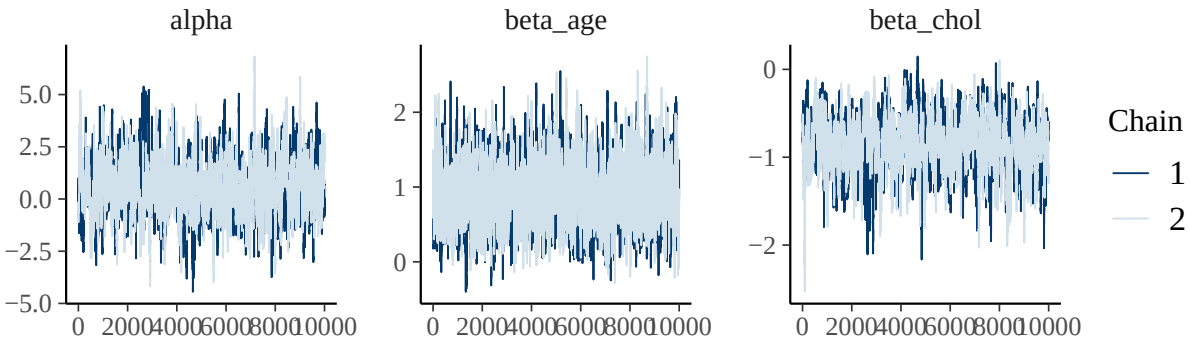
```
update(diabn_jag, 1000)
```

```
##   |                                                                      |
```

```
samn <- coda.samples(diab_jag,
                     c("alpha","beta_age", "beta_chol","or_age","or_chol"),
                     n.iter=10000)
```

```
##   |                                                                      |
```

```
mcmc_trace(samn, c("alpha","beta_age", "beta_chol"))
```




```
drawsn <- as_draws(samn)
```

```
summary(draws, median, quantile2, mcse_median, mcse_quantile)
```

```
## # A tibble: 9 x 7
##   variable  median     q5     q95 mcse_median mcse_q5 mcse_q95
##   <chr>      <num>  <num>   <num>       <num>   <num>    <num>
## 1 ac          5.76   5.42   6.11       0.00417 0.00535  0.00713
## 2 alpha       0.534 -1.60   2.99       0.0658  0.105    0.127
## 3 bc         -0.293 -0.555 -0.0354     0.00204 0.00276  0.00314
## 4 beta_age    0.908  0.320  1.57       0.0115  0.0133   0.0220
## 5 beta_chol -0.838 -1.39  -0.378       0.0147  0.0298   0.0184
## 6 cc         -0.386 -0.891  0.116      0.00498 0.00951  0.00554
## 7 cprec       0.611  0.448  0.807      0.00158 0.00150  0.00266
## 8 or_age      2.48   1.38   4.81       0.0285  0.0184   0.106
## 9 or_chol     0.432  0.248  0.685      0.00634 0.00740  0.0126
```

```
summary(drawsn, median, quantile2, mcse_median, mcse_quantile)
```

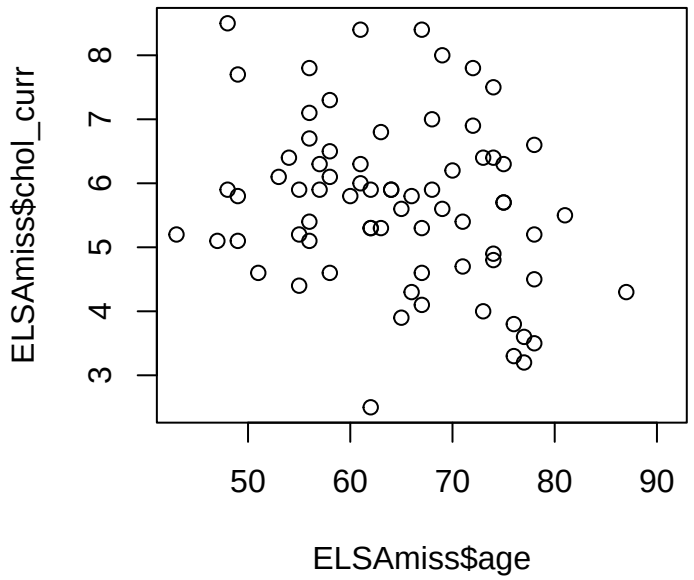
```
## # A tibble: 5 x 7
##   variable  median     q5    q95 mcse_median mcse_q5 mcse_q95
##   <chr>      <num>  <num>  <num>       <num>   <num>    <num>
## 1 alpha       0.558 -1.57   2.80       0.0546  0.105    0.105
## 2 beta_age    0.897  0.310  1.59       0.0113  0.0129   0.0238
## 3 beta_chol -0.839 -1.37  -0.393       0.0148  0.0262   0.0179
## 4 or_age      2.45   1.36   4.92       0.0278  0.0175   0.117
## 5 or_chol     0.432  0.255  0.675      0.00637 0.00668  0.0121
```

Focusing on the odds ratios in the model of interest (for current age and cholesterol, `or_age` and `or_chol`), these are substantively the same between the two analyses.

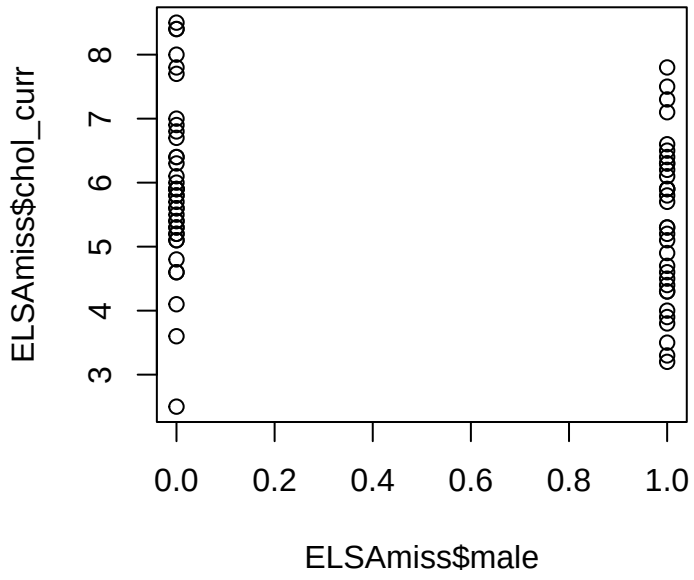
This is probably because the fully-observed variables (age and gender) are not strong predictors of

cholesterol. This can be seen from a quick scatterplot of these variables against each other, or from examining the posterior distributions of the coefficients `bc` and `cc`. Although men in this dataset have one unit lower cholesterol on average, the between-individual variability is very high by comparison.

```
plot(ELSAmiss$age, ELSAmiss$chol_curr)
```




```
plot(ELSAmiss$male, ELSAmiss$chol_curr)
```



Hence these variables are not very useful for predicting what the missing values of cholesterol might have been. So any increased precision (from including predicted values for the missing data in the model of interest) is balanced out by the reduction in precision due to uncertainty about what the missing values might have been.

(c) We would add a third regression model to the JAGS code, with blood pressure as an outcome, and age and gender as predictors.

We could then include blood pressure as a predictor in the covariate model with current cholesterol as an outcome, to improve the prediction of the missing cholesterol values.

In theory, we could also have had just age and gender in the model for current cholesterol, and age, gender and current cholesterol in the model for blood pressure. The information about blood pressure would have propagated "in reverse" to the missing cholesterol values via the regression model for blood pressure in terms of cholesterol.

There is no single correct answer, except that the important restrictions are to

- not include the outcome of interest as a predictor in the covariate model
- model each variable conditionally on the previous unmodelled variables, so that we don't have, e.g. one model for cholesterol given blood pressure, and another model for blood pressure given cholesterol.

(A useful way to visualise this kind of model would be as a directed acyclic graph (see Session 1). There should be no loops or cycles in the graph.)

Solutions: survival models

(a) and (b)

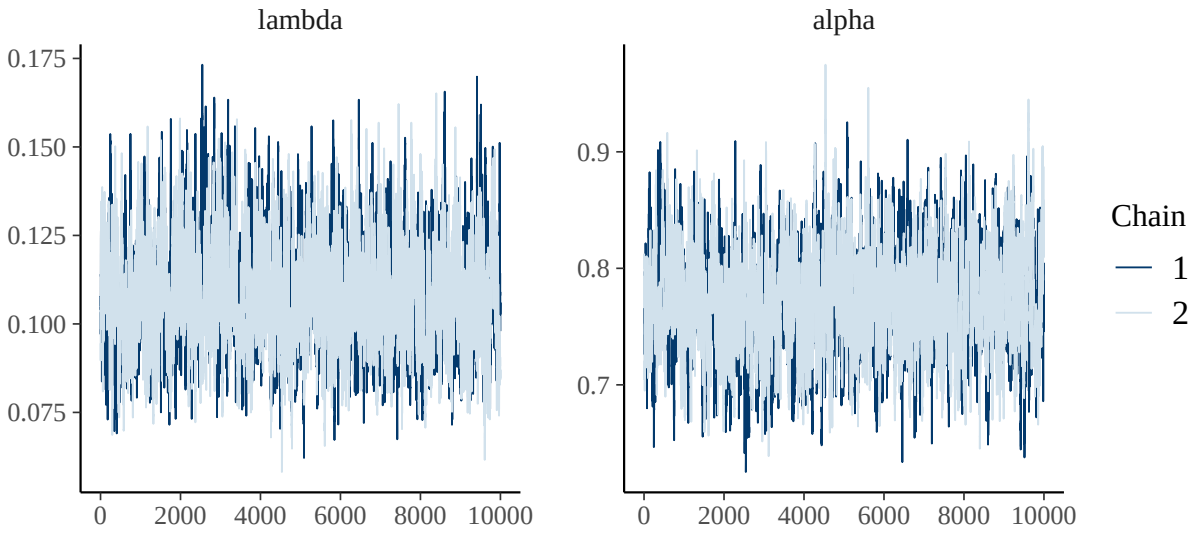
```
hosp <- read.csv("hosp_surv_sim.csv")
y <- hosp$days; y[hosp$event==0] <- NA
c <- hosp$days; c[hosp$event==1] <- Inf
dat <- list(y = y, c = c,
        is.censored = 1 - hosp$event,
        n=nrow(hosp))

surv_jagsmod <- "
model {
 for (i in 1:n) {
  y[i] ~ dweib(alpha, lambda)            # NA for censored observations
  is.censored[i] ~ dinterval(y[i], c[i]) # observation is censored
                                         # if y[i] is greater than c[i]
 }
 lambda <- 1 / exp(loginvlam)
 alpha ~ dgamma(1, 1)
 loginvlam  ~ dnorm(log(10), (1.96/(log(30)-log(10)))^2)

 n_pts <- 500
 mean_hospdays <- n_pts * lambda^(- 1 / alpha) * exp(loggam(1 + 1/alpha)) # (a)
 prob_20 <- 1 - pweib(20, alpha, lambda)                                  # (b)
 prob_50 <- 1 - pweib(50, alpha, lambda)                                  # for (c)
}"

ini <- list(list(alpha=1, loginvlam=0), list(alpha=1.1, loginvlam=1))
surv.jag <- jags.model(textConnection(surv_jagsmod), dat, n.chains=2, inits=ini,
                   quiet = TRUE)
update(surv.jag, 1000)
```

```
##   |                                                              |
```

```
sam <- coda.samples(surv.jag,
                var=c("lambda","alpha","mean_hospdays","prob_20","prob_50"),
                n.iter=10000)
```

```
##   |                                                              |
```

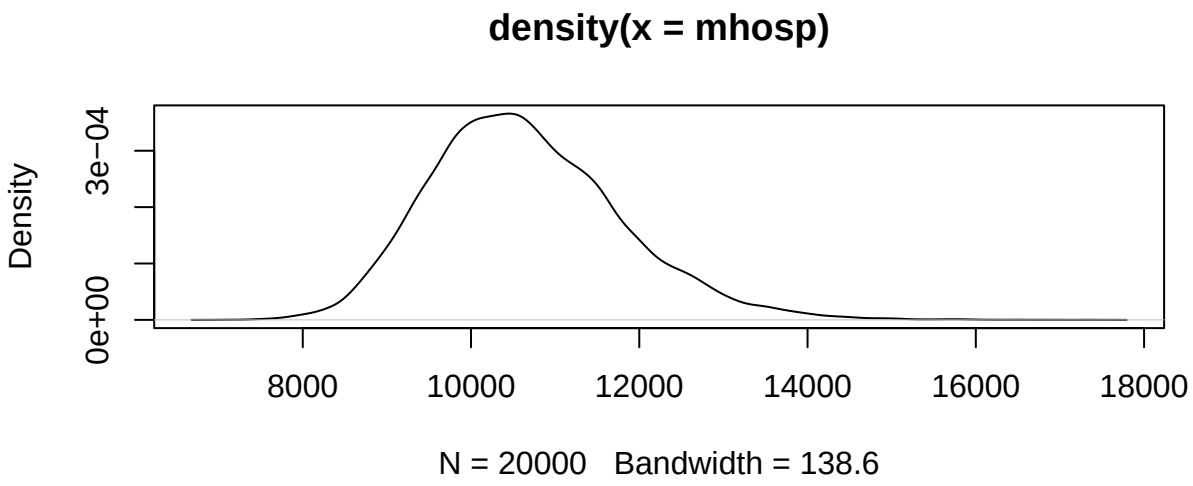
```
mcmc_trace(sam, c("lambda","alpha"))
```




```
summary(as_draws(sam))
```

```
## # A tibble: 5 x 10
##   variable           mean  median      sd     mad      q5     q95  rhat
##   <chr>             <num>   <num>   <num>   <num>   <num>   <num> <num>
## 1 alpha             0.773 7.72e-1 4.30e-2 4.26e-2 7.03e-1 8.44e-1  1.00
## 2 lambda            0.108 1.07e-1 1.48e-2 1.47e-2 8.48e-2 1.33e-1  1.00
## 3 mean_hospdays 10651.    1.05e+4 1.13e+3 1.10e+3 8.99e+3 1.27e+4  1.00
## 4 prob_20           0.340 3.40e-1 2.59e-2 2.59e-2 2.99e-1 3.83e-1  1.00
## 5 prob_50           0.113 1.12e-1 2.11e-2 2.12e-2 8.09e-2 1.50e-1  1.00
## # i 2 more variables: ess_bulk <num>, ess_tail <num>
```

```
mhosp <- as.matrix(sam)[,"mean_hospdays"]
plot(density(mhosp))
```




```
mean(mhosp > 15000)
```

```
## [1] 0.0015
```

In part (c), the trick is to model the binomial outcome `prior_y` with a probability (called `prob_50` here) defined by the survivor function (1-CDF) of the Weibull distribution at a time of 50 days. This survivor function is defined with parameters α and λ, the same parameters that govern the model for the individual-level data `y[i]`. Hence the `y[i]` and `prior_y` together provide information about α and λ. This is an

example of *evidence synthesis*, which will be covered more in Session 7.

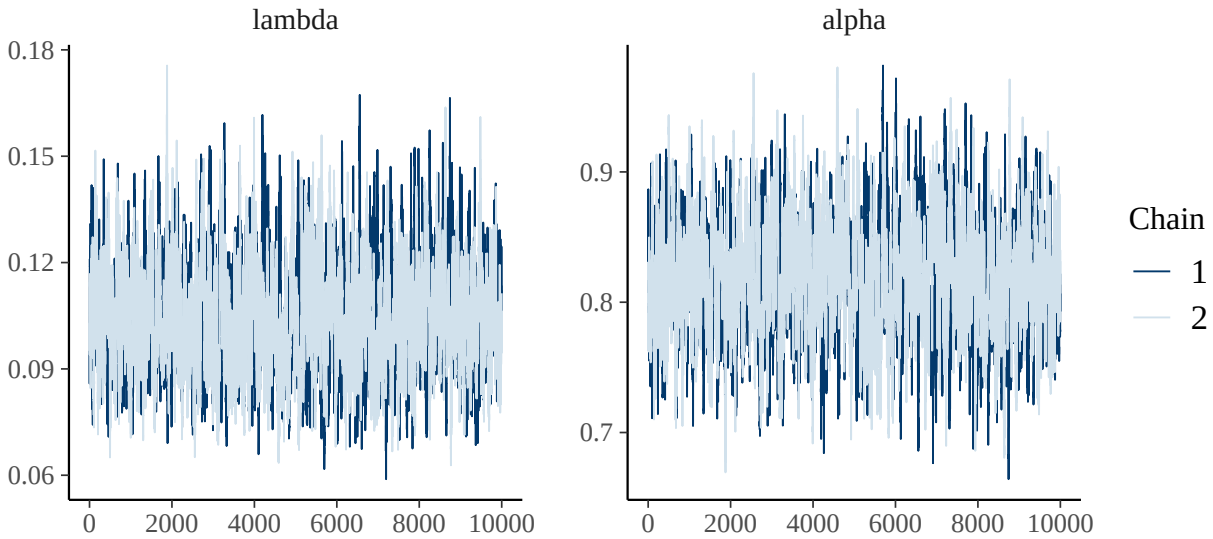
```
surv2_jagsmod <- "
model {
 for (i in 1:n) {
  y[i] ~ dweib(alpha, lambda)            # NA for censored observations
  is.censored[i] ~ dinterval(y[i], c[i]) # observation is censored
                                         # if y[i] is greater than c[i]
 }
 lambda <- 1 / exp(loginvlam)
 alpha ~ dgamma(1, 1)
 loginvlam  ~ dnorm(log(10), (1.96/(log(30)-log(10)))^2)

 prob_50 <- 1 - pweib(50, alpha, lambda)
 prior_y ~ dbin(prob_50, prior_n)
}"

betas <- SHELF::fitdist(vals=c(0.001, 0.01, 0.05), probs=c(0.025, 0.5, 0.975),
                        lower=0, upper=1)$Beta
prior_y <- round(betas$shape1 + 1)
prior_n <- round(betas$shape1 + betas$shape2 + 2)
dat2 <- list(y = y, c = c,
             is.censored = 1 - hosp$event,
             n=nrow(hosp),
             prior_y = prior_y, prior_n = prior_n)
ini <- list(list(alpha=1,  loginvlam=1),
            list(alpha=0.8, loginvlam=2))
surv2.jag <- jags.model(textConnection(surv2_jagsmod), data=dat2,
                        inits=ini, n.chains=2, quiet=TRUE)
update(surv2.jag, 1000)
```

```
##   |                                                              |
```

```
sam2 <- coda.samples(surv2.jag, var=c("lambda","alpha","prob_50"),
                     n.iter=10000)
```

```
##   |                                                              |
```

```
mcmc_trace(sam2, c("lambda","alpha"))
```



```
summary(as_draws(sam2))
```

```
## # A tibble: 3 x 10
##   variable   mean median     sd    mad     q5   q95  rhat ess_bulk
##   <chr>     <num>  <num>  <num>  <num>  <num> <num> <num>    <num>
## 1 alpha    0.817  0.817  0.0434 0.0428 0.747  0.891  1.00     779.
## 2 lambda   0.104  0.103  0.0150 0.0146 0.0806 0.130  1.00     808.
## 3 prob_50  0.0811 0.0803 0.0147 0.0145 0.0584 0.106  1.00    3429.
## # i 1 more variable: ess_tail <num>
```

The estimate of $P(T > 50)$ (posterior median and 95% credible interval) is

- 0.11 (0.08 to 0.16) without the strong prior information given by the pseudo-data `prior_y`
- 0.08 (0.05 to 0.11) with this strong prior information.

This amount of "shrinkage", from 0.11 to 0.08, towards a prior which had mean 0.01, makes sense given the relative amount of data: the prior was based on 112 "individuals", compared to 300 individuals in the data (though note given the different forms that these data take, censored time to event data, versus a binary outcome at a specific time, we would not expect the posterior to be a simple "weighted average" of the prior and data)

The predicted event probability at 50 days is based on three pieces of information: the prior, the observed data, and the assumption that the times to events follow a Weibull distribution. Nobody is observed for longer than 30 days in the original dataset `hosp`. Therefore, without any other information, the predicted probability of survival in hospital for 50 days is based on extrapolating the fitted Weibull model into the future. The results therefore depend on whether the fitted model is a representation of survival times up to 50 days - which cannot be checked from the observed data. The Bayesian model allows external information about longer-term survival probabilities to be included as priors.