How to use rstanbmcm

Introduction

This is a basic introduction to how to use `rstanbmcm` to fit Bayesian mixture cure models in Stan.

Data

We will use the Checkmate 067 study data set. The data have already been arranged in to the correct format and saved within the package so we can load it as follows.

```
data("surv_input_data", package = "rstanbmcm")
```

Required fields include event times (`os`, `pfs`) and censoring indicators (`os_event`, `pfs_event`) for both OS and PFS. There should also be a treatment label column (`TRTA`). Additional patient-level covariates can also be included. At present only age at event (`OSage`, `PFSage`) is used.

This looks like this.

```
head(surv_input_data)
#>   OSage PFSage        os os_event       pfs pfs_event                 TRTA SEX COUNTRY    ACOUNTRY P
#> 1    57     56 60.024641        0 59.663244         0 NIVOLUMAB+IPILIMUMAB   M     NLD NETHERLANDS
#> 2    78     77 19.449692        1  2.628337         1            NIVOLUMAB   F     NLD NETHERLANDS
#> 3    67     67  2.069815        1  2.069815         1           IPILIMUMAB   M     NLD NETHERLANDS
#> 4    48     47 60.188912        0 59.958932         0            NIVOLUMAB   F     NLD NETHERLANDS
#> 5    76     73 64.262834        0 32.295688         1            NIVOLUMAB   M     NLD NETHERLANDS
#> 6    78     76 34.891170        1  2.562628         1           IPILIMUMAB   M     NLD NETHERLANDS
```

Example

First of all attach all of the libraries we are going to need.

```
library(purrr)
library(reshape2)
library(dplyr)
library(rstan)
library(shinystan)
library(dplyr)
library(ggplot2)
library(rstanbmcm)
```

For demonstration purposes we will select a single treatment and fit Exponential distributions to both OS and PFS.

```
i <-  "exp"
k <- "exp"
j <- "IPILIMUMAB"
```

To use the Stan engine we set some options to use all-but-one of the available cores and not to over-write pre-complied code.

```
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores() - 1)
```

Now we are ready to do the model fitting. There are 2 options to use.

- `bmcm_joint_stan_file()`: calls the Stan file directly from R without pre-compiling. This is useful for development.
- `bmcm_joint_stan()`: uses the pre-compiled Stan code.

An example call to `bmcm_joint_stan_file` is given below.

```
out <-
  bmcm_joint_stan_file(
    input_data = surv_input_data,
    model_os = i,
    model_pfs = k,
    tx_name = j,
    params_pfs = list(mu_0 = c(-3, 0),
                      sigma_0 = c(0.5, 0.01)),
    params_os = list(mu_0 = c(-3, 0),
                     sigma_0 = c(0.4, 1)),
    params_cf = list(mu_cf_os = array(-0.8, 1),
                     mu_cf_pfs = array(-0.8, 1),
                     sd_cf_os = array(0.5, 1),
                     sd_cf_pfs = array(0.5, 1)),
    cf_model = 2,
    joint_model = FALSE,
    warmup = 100,
    iter = 1000,
    thin = 10)
```

Explanation of function arguments

- The first thing to note is that we supply the study data as the first argument `input_data`. We then define which distributions we want to fit to the OS anf PFS data, followed by the particular treatment subset of data to use from `input_data`.
- The next 3 arguments `params_pfs`, `params_os` and `params_cf` are the prior parameters for the PFS, OS and cure fraction distributions respectively. These must be supplied as a list. The two values for each parameter corresponds to the intercept and age effect in the linear equation component of the rate regression. For the parameters that are optional we have to wrap them with `array(.,1)` because Stan expects an array object even when it is of dimension (1,1). The cure fraction parameters here are optional because there are alternative ways of defining its prior i.e. using a Beta distribution or using the same prior for both OS and PFS. This example is for the separate cure fraction model.
- `cf_model` defines whether this is a pooled (1), separate (2) or hierarchical (3) cure fraction model.
- `joint_model` is a logical argument defining whether we model the OS and PFS event times jointly. If `joint_model = TRUE` then we must also pass the prior parameters using the `params_joint` argument.
- Finally, the remaining arguments are passed directly to the Stan engine.

Results

There are several very good packages available to view Stan output, including `shinystan` and `coda`. Below we give some basic output specific to `rstanbmcm`.

Output data

We can view the raw output from running the Stan model using

```
res <- extract(stan_exp_exp_IPI_)
```

The available posterior samples are

```
names(res)
#>  [1] "beta_os"       "beta_pfs"      "beta_bg"       "lp_cf_os"      "lp_cf_pfs"     "lp_os"
#> [12] "lambda_os_bg"  "lambda_pfs_bg" "cf_os"         "cf_pfs"        "mean_os"       "mean_pfs"
#> [23] "S_pfs_pred"    "pmean_os"      "pmean_pfs"     "pmean_bg"      "pmean_cf_os"   "pmean_cf_pfs"
#> [34] "pbeta_os"      "pbeta_pfs"     "pbeta_bg"      "lp__"
```

- Parameters prefixed with `p` are prior predicted distribution samples.
- Parameters prefixed with `lp_` (except `lp__`) are the linear predictors used for the Exponential distributions rate parameters `lambda_`.

Plots

The function `plot_S_joint()` takes a list of multiple Stan runs and creates `ggplot2` grid object of survival curves for cured, uncured and mixed with 95% credible intervals. In our example we simply create a list of one element.

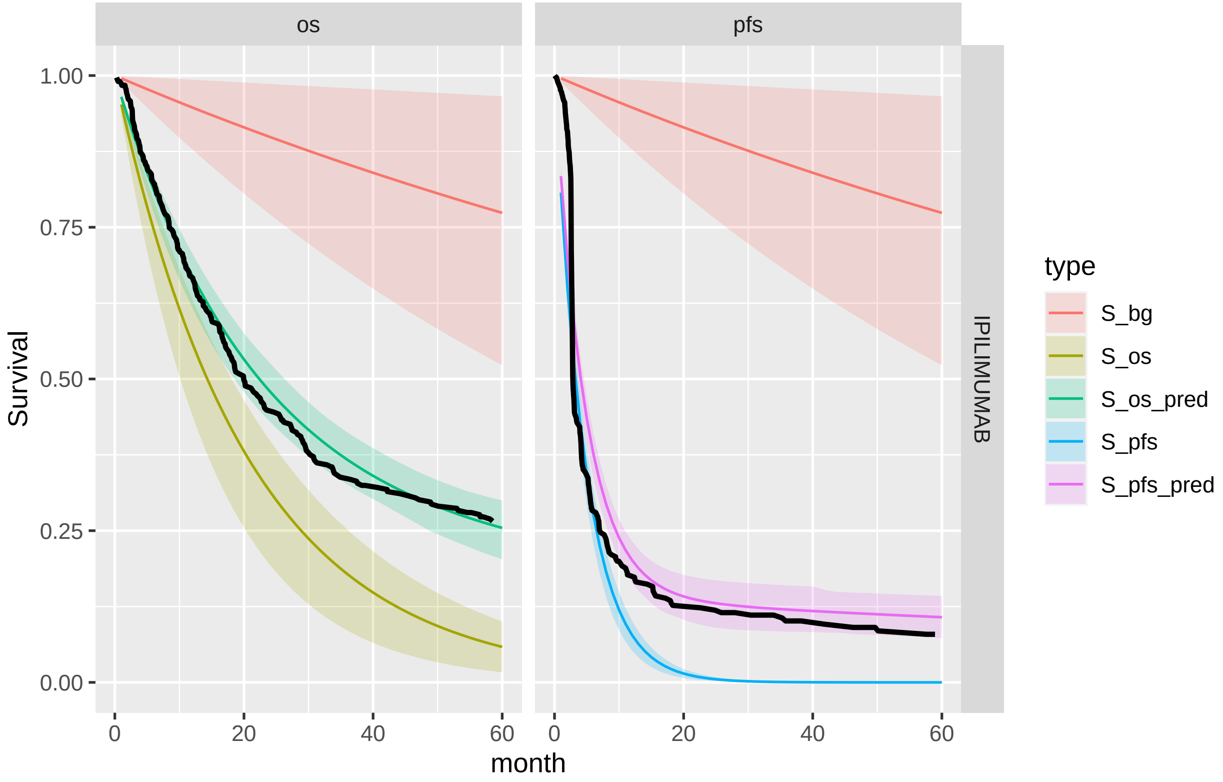
```
stan_list <- list("IPILIMUMAB" = stan_exp_exp_IPI_)
gg <- rstanbmcm:::plot_S_joint(stan_list = stan_list)
```

In addition, we overlay the Kaplan-Meier curves for the original data using the `survival` package. Note that this will always be slightly different since this is for the trial case-mix and the posterior survival curves are for the *average* patient.

```
library(survival)

trta <- "IPILIMUMAB"
fit_os <- survfit(Surv(os, os_event) ~ 1,
                  data = filter(surv_input_data, TRTA == trta))
fit_pfs <- survfit(Surv(pfs, pfs_event) ~ 1,
                   data = filter(surv_input_data, TRTA == trta))
km_data <-
  rbind(
    data.frame(Tx = trta,
               event_type = "os",
               time = fit_os$time,
               surv = fit_os$surv),
    data.frame(Tx = trta,
               event_type = "pfs",
               time = fit_pfs$time,
               surv = fit_pfs$surv))

gg + geom_line(aes(x = time, y = surv),
               data = km_data,
               lwd = 1,
               inherit.aes = FALSE) +
  xlim(0, 60)
```


os
pfs
IPILIMUMAB
Survival
1.00
0.75
0.50
0.25
0.00
0
20
40
60
0
20
40
60
month
type
S_bg
S_os
S_os_pred
S_pfs
S_pfs_pred