Practical 8. Survival analysis — SOLUTIONS

Lecture 8 PDF version

Survival analysis using `survHE`

Preliminaries

This practical assumes that you have installed `survHE`, a `R` package specifically designed to perform survival analysis in health economic evaluation and with advanced facilities for Bayesian modelling.

You can install `survHE` you can either use the "official" CRAN version, or the most-updated, "development". This can take a little time, as there are several "dependencies" (i.e. packages that are required for `survHE` to work properly).

```
# Install survHE from CRAN
install.packages("survHE")

# Or the development version fro GitHub
devtools::install.github("giabaio/survHE")
```

If `survHE` is installed, you simply need to load it into your `R` session, as usual and then we can also load some data.

```
# Loads survHE in the session
library(survHE)
```

```
Loading required package: flexsurv
```

```
Loading required package: survival
```

```
load("survival_data.Rdata")
```

This loads a dataset called `dat`, which contains some survival data. In particular, the dataset includes the patients ID; the time of progression to a more severe stage of cancer; an indicator for the event of interest (mortality); an indicator for the treatment arm (coded as 0 = control and 1 = active treatment); an indicator for the patients' sex (0 = male; 1 = female); the patients' age (in years); and the Index of Multiple Deprivation (IMD) score (this is a census-based, area-level measure of socio-economic circumstances. It is coded as categorical variable taking values in the interval $[1;5]$, where 1 indicates the least deprived and 5 indicates the most deprived areas). We can inspect it as usual, using built-in `R` functions.

```
# Loads survHE in the session
head(dat)
```

```
  ID_patient time event arm sex age imd
1          1 0.03     0   0   1  32   2
2          2 0.03     0   0   1  43   2
3          3 0.92     0   0   1  25   4
4          4 1.48     0   0   0  36   3
5          5 1.64     0   0   1  38   5
6          6 1.64     0   0   0  35   1
```

```
table(dat$arm)
```

```

  0   1
189 178
```

```
table(dat$arm,dat$event)
```

```
    0   1
0  90  99
1 105  73
```

There are 189 individuals in arm `0` (controls) and 178 in the arm `1` (some active drug). The data include a patient ID, the time at which the event has been observed (e.g. progression to a worse disease state) and an indicator for censoring. Individuals who are not fully observed are associated with `censored=1`.

Model fitting in a frequentist setting

We are instructed to fit both the Exponential and the Weibull model to the data, assuming a linear predictor of the form

$$g(\mu_i) = \log(\mu_i) = \beta_0 + \beta_1 \texttt{arm}_i.$$

In order to analyse the data, we first need to define the model we want to use and the distributions we want to use. We can simply set this out using the following `R` commands.

```
# Defines the model formula and the distributions
formula=Surv(time,event)~as.factor(arm)
mods=c("exp","weibull")

# Then runs survHE to estimate the two models
m1=fit.models(formula=formula,data=dat,distr=mods)
```

The `formula` specifies the model in terms of regression for the generalised linear predictor, which in this case only depends on the treatment arm (notice that, because `arm` is a categorical variable, we include it in our analysis as a `R` "`factor`"; the first value `arm=0` will be used as reference category). Notice also that we need to use the specific notation `Surv(time=time, event=event)` to tell `R` and `survHE` that our data are in survival analysis form. Then we set up a vector `mods` in which we include some string text identifying the Exponential and Weibull models (more details are available in the `survHE` documentation). Finally, we are ready to run the function `fit.models`, which is used by `survHE` to perform the analysis and estimate the model parameters.

The results of the models are stored in an object `m1`, which contains several elements.

```
# Explores the model output
names(m1)
```

```
[1] "models"        "model.fitting" "method"        "misc"
```

```
lapply(m1,names)
```

```
$models
[1] "Exponential"   "Weibull (AFT)"

$model.fitting
[1] "aic" "bic" "dic"

$method
NULL

$misc
[1] "time2run"   "formula"    "data"       "model_name" "km"
```

The `R` command `lapply` can be used to "apply" the function `names` to all the elements of the list `m1`, to provide details of each of its elements. So, for example, the object `m$models` contains two objects (`Exponential` and `Weibull (AFT)`), in which the estimates are stored.

The output for the modelling can be visualised using the `print` method, as follows.

```
# Defines the model formula and the distributions
print(m1)
```

```
Model fit for the Exponential model, obtained using Flexsurvreg
(Maximum Likelihood Estimate). Running time: 0.020 seconds

                      mean         se       L95%      U95%
rate              0.0824203 0.00828355  0.0676839  0.100365
as.factor(arm)1 -0.4656075 0.15427131 -0.7679738 -0.163241

Model fitting summaries
Akaike Information Criterion (AIC)....: 1274.576
Bayesian Information Criterion (BIC)..: 1282.387
```

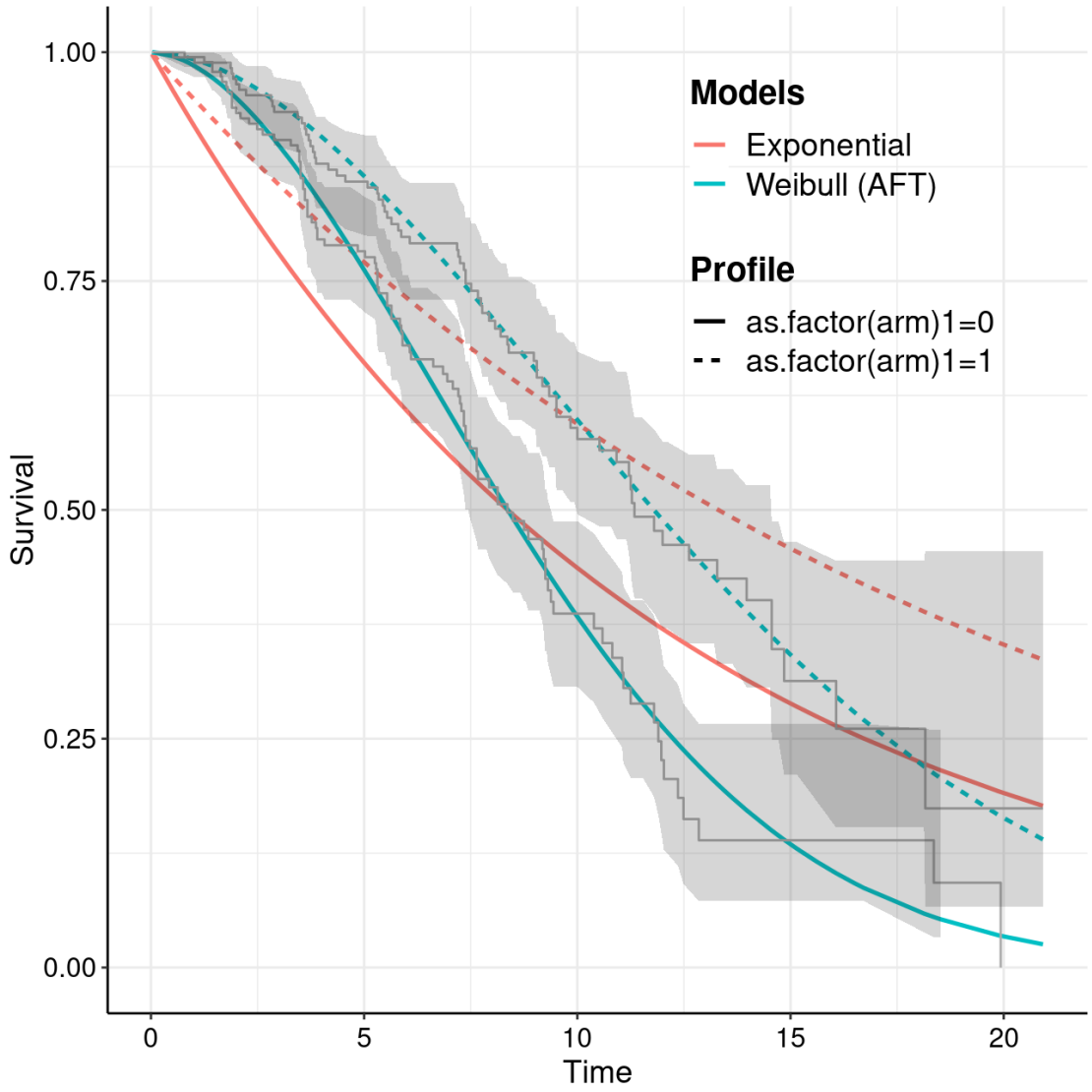
```
print(m1,2)
```

```
Model fit for the Weibull AF model, obtained using Flexsurvreg
(Maximum Likelihood Estimate). Running time: 0.012 seconds

                     mean        se     L95%      U95%
shape            1.816383 0.1098390 1.613371  2.044941
scale           10.220953 0.5705218 9.161747 11.402616
as.factor(arm)1  0.342019 0.0855445 0.174355  0.509683

Model fitting summaries
Akaike Information Criterion (AIC)....: 1203.130
Bayesian Information Criterion (BIC)..: 1214.846
```

This takes an optional argument, which allows to specify which model should be printed, in case more than one distribution has been selected (e.g. in this case). Notice that, by default, `survHE` uses maximum likelihood as the "`method`" to perform the estimation (as reported by the output of the `print` function).

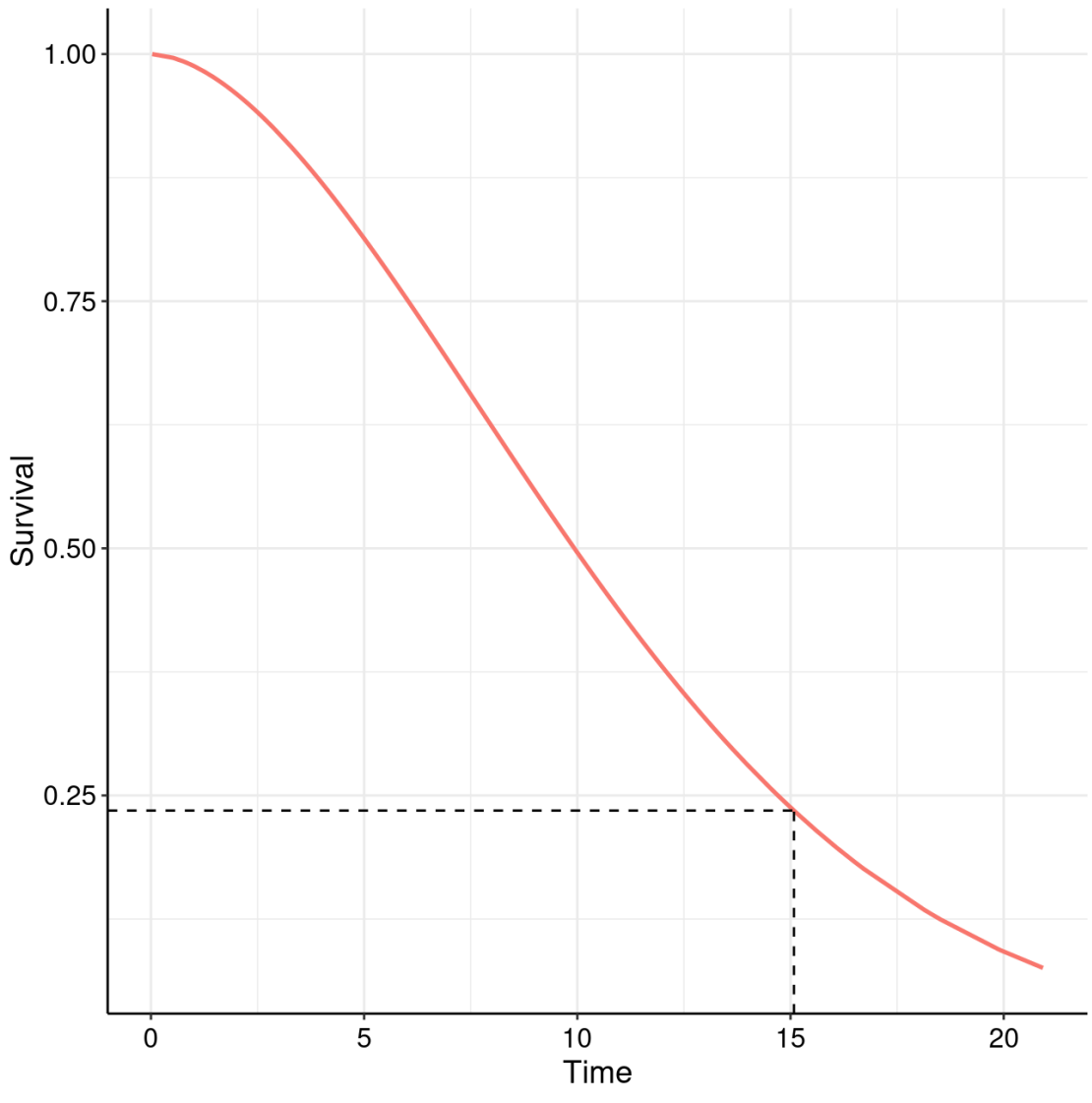
The model output can also be plotted in terms of the resulting survival curves, on top of the Kaplan-Meier estimate. This can be done using the `plot` command, using the option `add.km=TRUE`.

```
# Defines the model formula and the distributions
plot(m1,add.km=TRUE)
```



The resulting graph shows the survival curves within the observed time-frame (0.03—20.92), for all the models fitted in `m`. As expected from the theory, the Exponential model does not do a good job at following the observed shape of the data, as it is not flexible enough. The Weibull model is much closer to the empirical estimate provided by the Kaplan-Meier curve. This is confirmed by the analysis of the Information Criterion statistics (AIC and BIC): for the Exponential model they are both greater than the equivalent values obtained for the Weibull model, indicating that the latter fit the observed data better.

In general terms, the survival curves (which are just 1 — the cumulative probability curves) can be used to read off the relevant probability at a given time. For example, if we consider the following graph, it would be fairly easy to read off that at time $t = 15$, the survival probability is roughly about 0.25 (in fact, to be precise, it can be computed with some algebra to be 0.3797412). Similar (approximate) computations can be made on a grid of values (as represented in the graph) for different times and or probability values.



It is often useful to compute, at least in an approximate ways, (survival) probabilities using this method.

Bayesian modelling

As mentioned in the lecture, standard MCMC algorithms may struggle with survival data, especially when they are characterised by a large number of censored observations. Thus, `survHE` implements Bayesian analysis using two alternative Bayesian computation methods. The first one is based on Integrated Nested Laplace Approximation (INLA), while the second uses a variant of MCMC called Hamiltonian Monte Carlo (HMC).

Without going too much into the details (some of which are described in the `survHE` manual, INLA is very fast (almost as fast as the MLE procedure) and produces precise results, but is only available (at present) for a limited set of distributions. On the other hand, HMC is a little slower, but is perhaps a little more flexible and allows for more distributional assumptions.

In `survHE`, it is very simple to specify what "method" of inference should be used, by simply setting the option `method` to either `mle` (the default), or `inla`, or `hmc`. So, for example, we could replicate the analysis above using INLA by simply using the following command.

```
# Runs survHE to estimate the two models using INLA
m2=fit.models(formula=formula,data=dat,distr=mods,method="inla")

# Shows the output for the Exponential model
print(m2)
```

```
Model fit for the Exponential model, obtained using INLA (Bayesian inference via
Integrated Nested Laplace Approximation). Running time: 0.62141 seconds

                      mean         se       L95%      U95%
rate              0.0828715 0.00836297  0.0669431  0.100186
as.factor(arm)1 -0.4665097 0.14721656 -0.7474321 -0.177349

Model fitting summaries
Akaike Information Criterion (AIC)....: 1276.583
Bayesian Information Criterion (BIC)..: 1288.299
Deviance Information Criterion (DIC)..: 1277.371
```

```
# And then for the Weibull model
print(m2,2)
```

```
Model fit for the Weibull AF model, obtained using INLA (Bayesian inference via
Integrated Nested Laplace Approximation). Running time: 1.284 seconds

                     mean        se     L95%      U95%
shape            1.764107 0.1114979 1.552269  1.973406
scale           10.281869 0.6121163 9.186980 11.575483
as.factor(arm)1  0.344241 0.0893014 0.182237  0.532154

Model fitting summaries
Akaike Information Criterion (AIC)....: 1205.359
Bayesian Information Criterion (BIC)..: 1220.981
Deviance Information Criterion (DIC)..: 1206.669
```

As is possible to see, many of the results are very similar to the MLE analysis above. This is because, by default, both the INLA and HMC implementation use relatively weak prior distributions for both the location $\mu_i = g^{-1}(\bm\beta)$ and the ancillary parameters $\bm\alpha$ (see lecture slides). These priors can be modified, but this requires some changes to the call to the `fit.models` function (see the manual for more details). Because INLA specifies a Bayesian model, there is an additional Information Criterion available, the DIC, which is also printed in the summary tables. Once again, the Weibull model is preferable as it is associated with lower values of the AIC, BIC and DIC.

In a very similar way, we can specify the models using HMC as the inferential engine, by using the following command.

```
# Runs survHE to estimate the two models using HMC
m3=fit.models(formula=formula,data=dat,distr=mods,method="hmc")
```

and we can still use the `print` method to visualise the results.

```
# Shows the output for the Exponential model
print(m3)
```

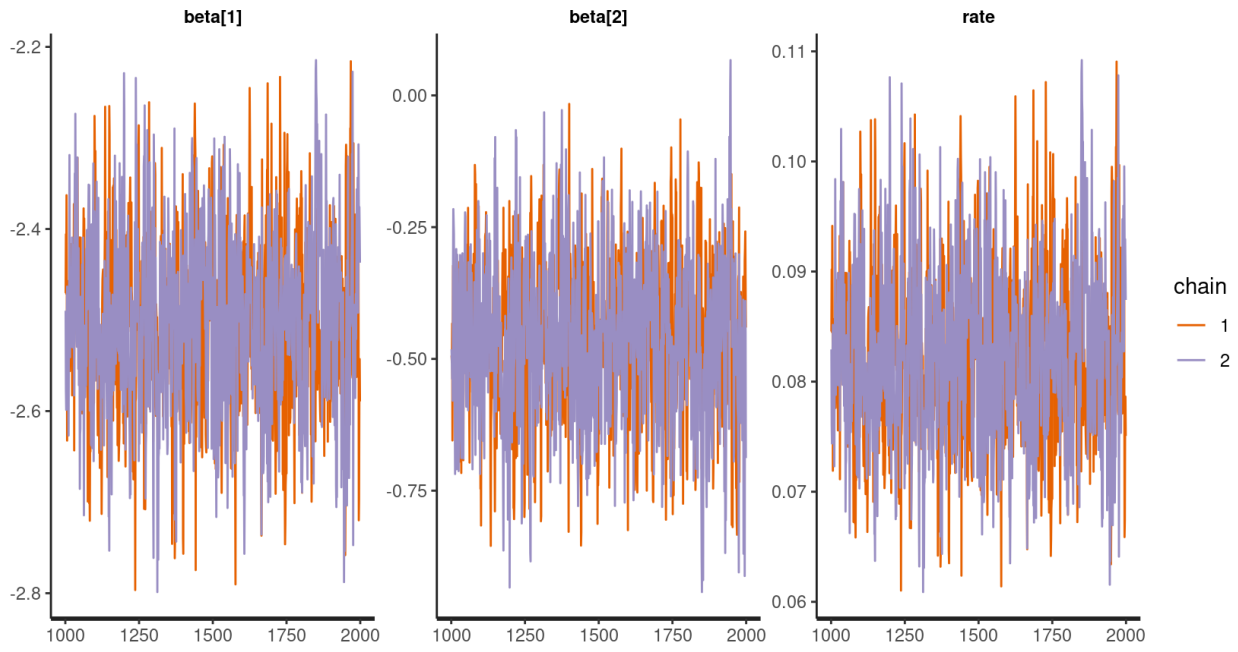
```
Model fit for the Exponential model, obtained using Stan (Bayesian inference via
Hamiltonian Monte Carlo). Running time: 1.1656 seconds

                      mean         se       L95%       U95%
rate              0.0821905 0.00792447  0.0676933  0.0988463
as.factor(arm)1 -0.4626672 0.15001521 -0.7682862 -0.1793789

Model fitting summaries
Akaike Information Criterion (AIC)....: 1276.579
Bayesian Information Criterion (BIC)..: 1288.295
Deviance Information Criterion (DIC)..: 1274.295
```

```
# And then for the Weibull model
print(m3,2)
```

```
Model fit for the Weibull AF model, obtained using Stan (Bayesian inference via
Hamiltonian Monte Carlo). Running time: 2.6113 seconds

                     mean        se     L95%      U95%
shape            1.804606 0.1106481 1.594570  2.018583
scale           10.273537 0.5923679 9.243481 11.492342
as.factor(arm)1  0.346925 0.0892387 0.177872  0.518907

Model fitting summaries
Akaike Information Criterion (AIC)....: 1205.143
Bayesian Information Criterion (BIC)..: 1220.765
Deviance Information Criterion (DIC)..: 1203.313
```

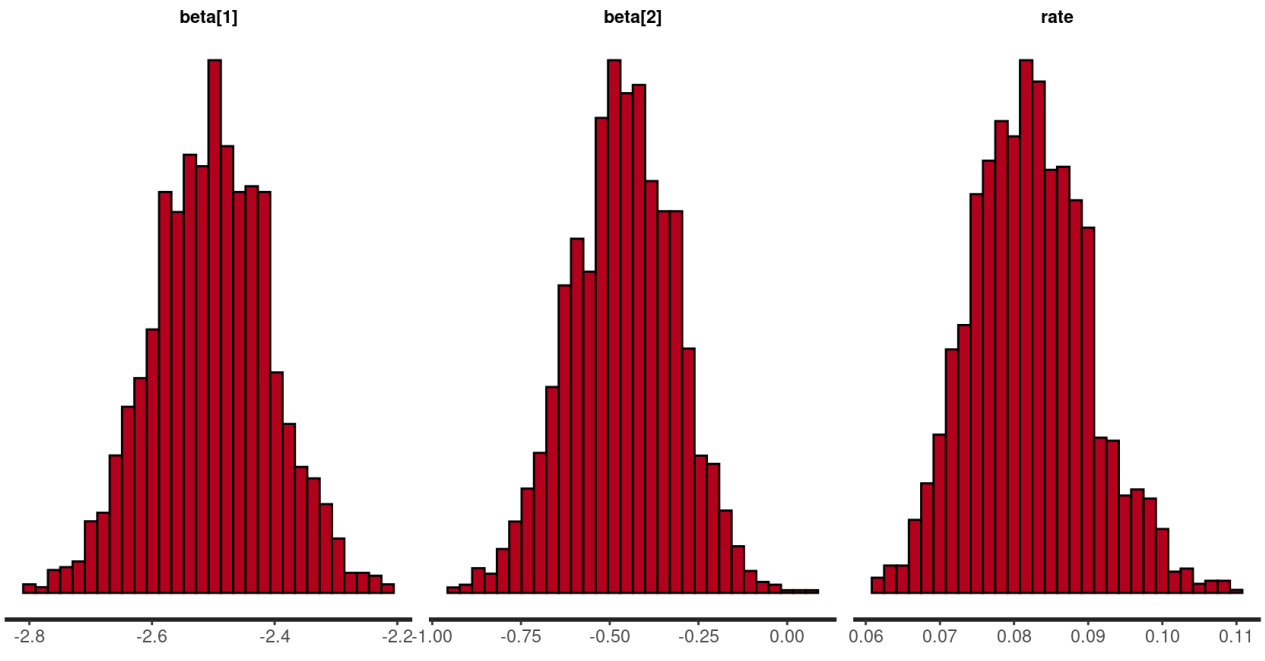
Once again, the results are fairly similar, numerically, due to the fact that the priors are relatively weak and there are enough data to consistently inform the posterior distributions for the parameters. Again, the `survHE` manual explains in more details how the priors can be modified in order to include genuine information. Because HMC is an MCMC algorithm, we can check the convergence diagnostics, much as we had done for the `BUGS` output in the previous practicals. In particular, we could check the traceplots and histograms for the posterior distributions of the parameters using built-in functions in the `rstan` package, which `survHE` uses to perform the HMC analysis, as in the following.

```
# Traceplots for the parameters of the Exponential model (the first element of m3$models)
rstan::traceplot(m3$models[[1]])
```




```
# Histograms for the parameters of the Weibull model (the second element of m3$models)
rstan::stan_hist(m3$models[[1]])
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



A more familiar version of the summary statistics table for the HMC output can be obtained by adding another optional argument to the call to `print`, as follows.

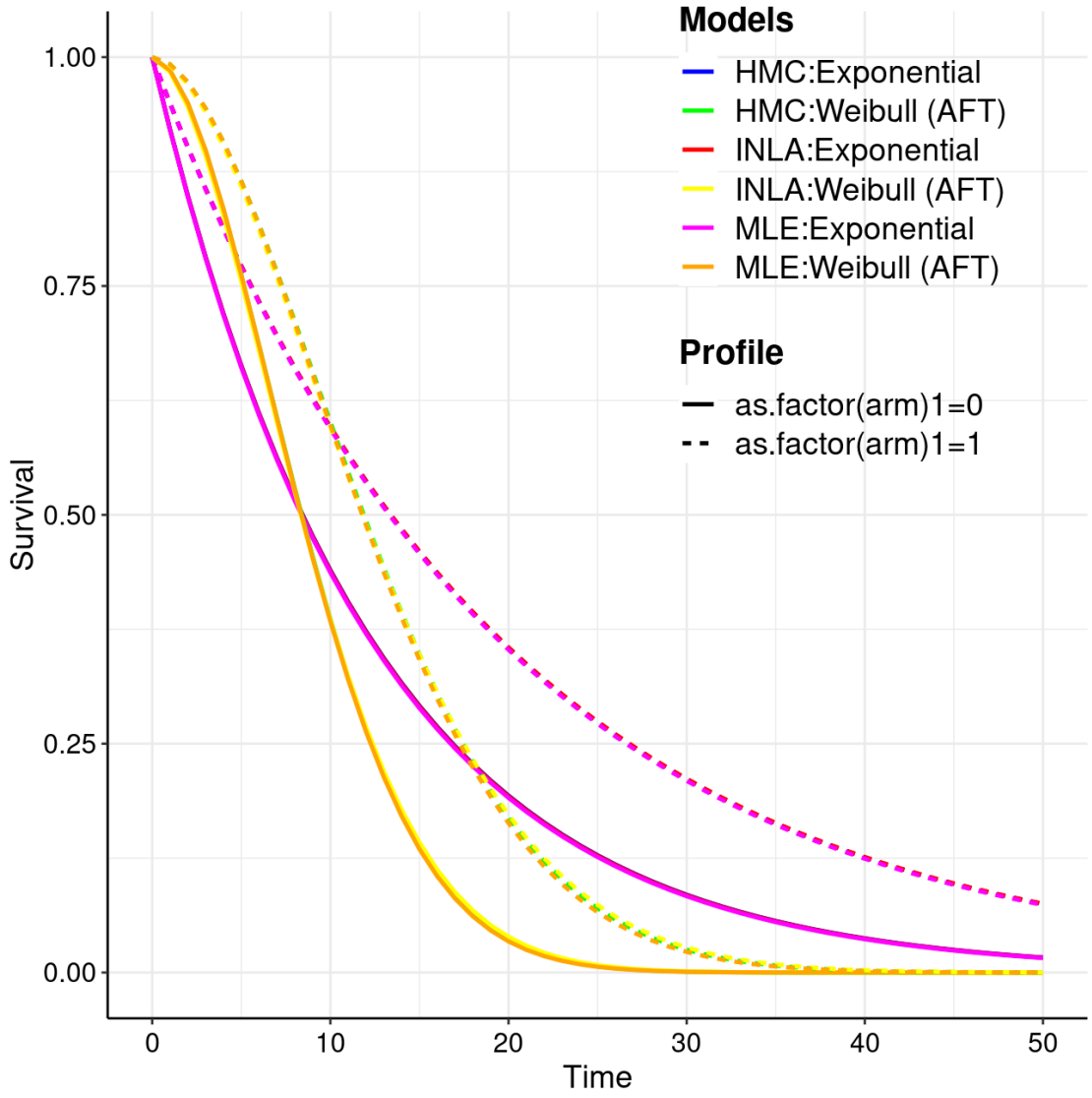
```
# Shows the output for the Exponential model
print(m3,2,original=TRUE)
```

```
Inference for Stan model: WeibullAF.
2 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=2000.

              mean  se_mean       sd        2.5%         25%         50%
beta[1]    2.327924 0.001666 0.057311    2.223919    2.286785    2.325146
beta[2]    0.346925 0.002579 0.089239    0.177872    0.286620    0.343483
alpha      1.804606 0.002958 0.110648    1.594570    1.726319    1.803580
scale     10.273537 0.017270 0.592368    9.243481    9.843244   10.228174
lp__    -600.346661 0.039879 1.251980 -603.624141 -600.881002 -600.046654
               75%       97.5% n_eff     Rhat
beta[1]    2.366375    2.441681  1184 0.999171
beta[2]    0.407953    0.518907  1197 0.999553
alpha      1.883740    2.018583  1399 0.999804
scale     10.658682   11.492342  1177 0.999187
lp__    -599.475113 -598.892282   986 1.002001

Samples were drawn using NUTS(diag_e) at Mon Jun  6 14:39:34 2022.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

We can easily see that the $\hat{R}$ statistic is below the arbitrary threshold of 1.1 for all the nodes and that the effective sample size `n_eff` is also rather close to the nominal sample size of 2000, indicating that convergence is reached and autocorrelation is not an issue.

We can plot the results of all the model, selectively, by specifying a more complex call to the `plot` function, for example as in the following.

```
plot(MLE=m1,INLA=m2,HMC=m3,
     # Selects which models from the three fitted objects
     mods=c(1,2,3,4,5,6),
     # Specifies colours to plot the curves
     colour=c("blue","green","red","yellow","magenta","orange"),
     # Defines the time horizon over which to make the plot
     t=seq(0,50)
)
```



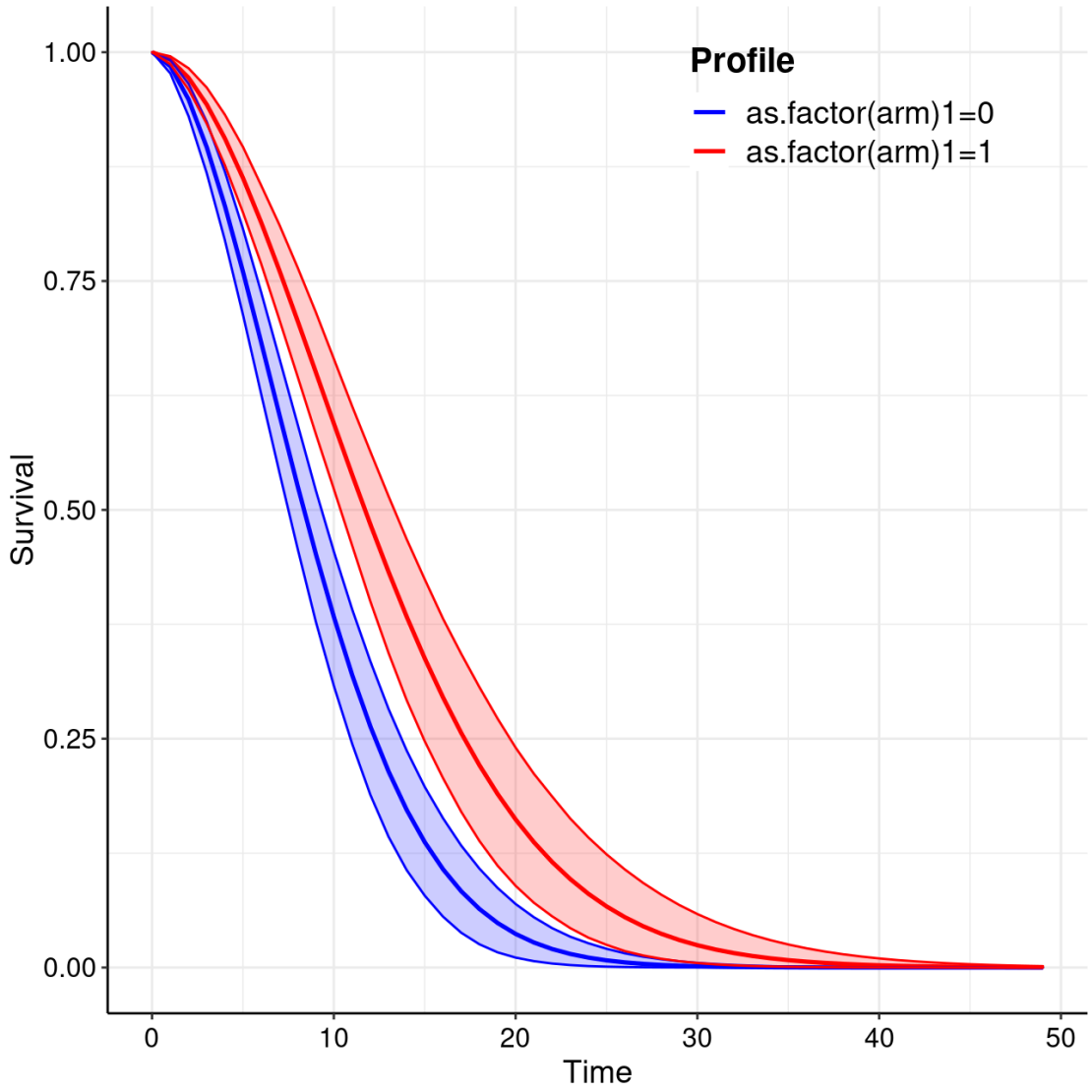
Here, the option `MLE=m1,INLA=m2,HMC=m3,mod=c(1,2,3,4,5,6)` instructs `R` to first stack together the three objects `m1`, `m2` and `m3` (and give them the names `MLE`, `INLA` and `HMC`) and the to select the models 1 to 6 (in this case, all of them, because in each method we have fitted two distributions). Then we specify colours and labels. As is possible to see, there is virtually no difference in the estimates for the Exponential model, while there are some minor ones for the Weibull. We can also set an option `t=seq(0,50)`, which instructs `R` to extrapolate the survival curves beyond the observed data and up to time = 50.

Probabilistic sensitivity analysis

`survHE` is designed to perform automatically PSA on the survival curves, based on the underlying uncertainty in the model parameters. Irrespective of the inferential engine (MLE or Bayesian), the function `make.surv` uses a simulation approach (based either on boostrap in the case of MLE, or simulations from the posterior distributions in the case of the Bayesian models) to then reconstruct the entire probability distribution of the survival curves, in a specified time range.

For example, the following code constructs an object `psa1` in which `nsim=1000` simulations for the survival curves of `mod=2` (the Weibull specification) in `m1` (the MLE analysis) are stored.

```
# Performs PSA on the survival curves for the Weibull model (under MLE)
psa1=make.surv(m1,mod=2,t=seq(0.01,50),nsim=1000)
psa.plot(psa1,offset=2.5,col=c("blue","red"))
```



The specialised function `psa.plot` can be used to visualise the resulting survival curves and the underlying uncertainty. `psa.plot` can be customised, e.g. by specifying the colour with which the curves need to be plotted, or the distance between the terms of the label, which appears in the top part of the graph. These describe the combination of covariates associated with each curve — in this case, the blue curve is associated with a value of the intercept of 1 and a value of the treatment arm of 0 (i.e. the control arm), while the red curve is associated with a value of 1 for the treatment arm (i.e. the active treatment).

In fact, the most recent (and current) version of `survHE` can use the simpler function `plot` to perform the extrapolation and PSA (see here).

Without getting into the technical details, the process can be replicated for the Bayesian models — the main difference here is in the fact that in this case (and particularly under HMC), the resulting simulations will be a better approximation of the underlying joint probability distribution of all the model parameters. As mentioned in the classes, in cases where there is substantial correlation among the parameters of the survival model (α, β), then this is likely to give results that may differ from the rougher approximation based on bootstrap.
