Improved curve fits to summary survival data: using and extending the Hoyle & Henley spreadsheet in R

Frederick Thielen

2021-02-10

Background

The spreadsheet accompanying the Hoyle and Henley paper on improving curve firs to summary survival data (here), is a trusted tool to digitalise published Kaplan-Meier curves and using the parameters of the parametric model for economic models in Excel.

The Excel spreadsheet (here referred to as "Hoyle and Henley spreadsheet") provides some R code to fit several parametric models. However, R code for the other standard models (as defined by NICE DSU Technical Support Document 14) like the *Gompertz* and *generalised gamma* function are not included. The most likely reason for this is probably that the **survival** package (which is used in the Hoyle and Henley spreadsheet) does not support these two functions.

Here, I provide the R code to estimate both the *Gompertz* and *generalised gamma* function using the **flexsurv** package. In addition, I show how to use the Hoyle and Henley sheet directly, without the need of exporting data from the "R data" sheet into a separate .txt file.

Assumptions

I assume that:

- you have a very basic understanding of R
- you know the Hoyle and Henley spreadsheet and also how to use it

Preparations

To get started with the following code, you should follow steps 1 and 2 explained in the Hoyle and Henley spreadsheet and save one spreadsheet for each arm of the Kaplan-Meier curve. Make sure to give each Hoyle and Henley spreadsheet a unique and identifiable name.

Next, open RStudio and create a new project in a directory of your choice. If you don't know what a project in RStudio is read here.

Within your project, create a new folder called "data". Put all your Hoyle and Henley spreadsheets in this folder.

Now you can open a new R script. If you don't know what a script is read here.

The following steps come *after* step 1-2 of the Hoyle and Henley spreadsheet.

Step 3: read the data

There is no need to store the "R data" sheet in a separate text (.txt) file! Make sure you have the following packages installed. If you don't know how this works, read here.

We need the following packages:

```
library("readxl")
library("here")
library("survival")
library("flexsurv")
library("dplyr")
```

Now, we can read-in the data. Copy/paste the following code into you script and run it. Make sure to replace my file path with the one to the Excel sheet of you choice. If you followed the steps above, this should be: `path = here::here("data/HoyleHenleySheetYouWant.xls")`.

```
# This is the actual R data sheet
spreadsheet_data <- readxl::read_xls(
  path = here::here("data/posts/2021/hoyle_henley/HoyleHenleyVersion_1.2.xls"),
  sheet = "R data",
  col_types = "numeric") %>%
  na.omit(.)

# This is needed too
meta_data <- readxl::read_xls(
  path = here::here("data/posts/2021/hoyle_henley/HoyleHenleyVersion_1.2.xls"),
  sheet = "Number events & censored",
  range = "O14:O16",
  col_names = FALSE)
```

Step 4: adding time for patients at risk at last time point

Copy/paste the following code into you script and run it.

```
# Prepare times_start
times_start <-c(
  rep(spreadsheet_data$start_time_censor,
      spreadsheet_data$n_censors),
  rep(spreadsheet_data$start_time_event,
      spreadsheet_data$n_events))

if(meta_data[1, 1] == "Yes"){
  times_start <- c(times_start,
                   rep(as.numeric(last(spreadsheet_data$end_time_event)),
                       as.numeric(meta_data[2, 1]))
                   )
}

# Prepare times_end
times_end <-c(
  rep(spreadsheet_data$end_time_censor,
      spreadsheet_data$n_censors),
  rep(spreadsheet_data$end_time_event,
      spreadsheet_data$n_events))
```

```
if(meta_data[1, 1] == "Yes"){
  times_end <- c(times_end,
                 rep(as.numeric(last(spreadsheet_data$end_time_censor)),
                     as.numeric(meta_data[2, 1]))
                 )
}
```

Step 5: curve fitting

Now we can fit the parametric models. Other than in the Hoyle and Henley spreadsheet, we do not need to comment out each line of code in Step 5 or run each line separately. Here, all models have distinct names:

- `fit1` = exponential model
- `fit2` = weibull model
- `fit3` = lognormal model
- `fit4` = log logistic model
- `fit5` = Gompertz model
- `fit6` = generalised gamma model

```
# Here we define all possible distributions once so we don't need to type them each time
possible_dist <- c("exponential", "weibull", "lognormal", "llogis", "gompertz",
                   "gengamma")

#### Fitting with the survival package (like in the Hoyle and Henley spreadsheet) ####
fit1_survival <- survival::survreg(
  Surv(times_start, times_end, type = "interval2") ~ 1,
  data = spreadsheet_data,
  dist = possible_dist[1])

fit2_survival <- survreg(
  Surv(times_start, times_end, type = "interval2") ~ 1,
  data = spreadsheet_data,
  dist = possible_dist[2])

fit3_survival <- survreg(
  Surv(times_start, times_end, type = "interval2") ~ 1,
  data = spreadsheet_data,
  dist = possible_dist[3])

fit4_survival <- survreg(
  Surv(times_start, times_end, type = "interval2") ~ 1,
  data = spreadsheet_data,
  dist = "loglogistic")

#### Fitting with the flexsurv package ####
fit1 <- flexsurv::flexsurvreg(
  Surv(times_start, times_end, type = "interval2") ~ 1,
  data = spreadsheet_data,
  dist = possible_dist[1])

fit2 <- flexsurvreg(
  Surv(times_start, times_end, type = "interval2") ~ 1,
  data = spreadsheet_data,
  dist = possible_dist[2])
```

```
## Warning in flexsurvreg(Surv(times_start, times_end, type = "interval2") ~ :
## Optimisation has probably not converged to the maximum likelihood - Hessian is
## not positive definite.
```

```
fit3 <- flexsurvreg(
  Surv(times_start, times_end, type = "interval2") ~ 1,
  data = spreadsheet_data,
  dist = possible_dist[3])

fit4 <- flexsurvreg(
  Surv(times_start, times_end, type = "interval2") ~ 1,
  data = spreadsheet_data,
  dist = possible_dist[4])
```

```
## Warning in flexsurvreg(Surv(times_start, times_end, type = "interval2") ~ :
## Optimisation has probably not converged to the maximum likelihood - Hessian is
## not positive definite.
```

```
fit5 <- flexsurvreg(
  Surv(times_start, times_end, type = "interval2") ~ 1,
  data = spreadsheet_data,
  dist = possible_dist[5])
```

```
## Warning in flexsurvreg(Surv(times_start, times_end, type = "interval2") ~ :
## Optimisation has probably not converged to the maximum likelihood - Hessian is
## not positive definite.
```

```
fit6 <- flexsurvreg(
  Surv(times_start, times_end, type = "interval2") ~ 1,
  data = spreadsheet_data,
  dist = possible_dist[6])
```

You see that there are several warning message for the example data. If this is also the case for your data, you should consider using the **survival** package only.

Step 6: AIC/BIC values

In the next step, we can extract the AIC and BIC values.

```
aicbic <- data.frame(distribution = possible_dist,
                             AIC = c(fit1$AIC, fit2$AIC, fit3$AIC,
                                     fit4$AIC, fit5$AIC, fit6$AIC),
                             BIC = c(BIC(fit1), BIC(fit2), BIC(fit3),
                                     BIC(fit4), BIC(fit5), BIC(fit6)),
                             stringsAsFactors = FALSE) %>%
          mutate(AIC_rank = rank(AIC),
                 BIC_rank = rank(BIC)) %>%
          select(distribution, AIC, AIC_rank, BIC, BIC_rank)

aicbic
```

```
##   distribution       AIC AIC_rank       BIC BIC_rank
## 1  exponential  702.5167        5  705.7262        5
## 2      weibull  663.9915        1  670.4105        1
## 3    lognormal  670.4610        3  676.8799        3
## 4       llogis  690.7154        4  697.1343        4
```

```
## 5     gompertz 1495.6940        6 1502.1130        6
## 6     gengamma  665.7108        2  675.3393        2
```

Step 7: Cholesky matrix

With flexsurv

In the next step we can extract the Cholesky matrix for **flexsurv** package. Be aware that this only works when the models converge. So if you received a warning message while fitting the model, you do not even have to try getting the Cholesky matrix.

```
chol(fit1$cov)
```

```
##           rate
## rate 0.1058511
```

```
chol(fit2$cov)
```

```
## Error in chol.default(fit2$cov): the leading minor of order 1 is not positive definite
```

```
chol(fit3$cov)
```

```
##            meanlog      sdlog
## meanlog 0.07206188 0.02748009
## sdlog   0.00000000 0.07278072
```

```
chol(fit4$cov)
```

```
## Error in chol.default(fit4$cov): the leading minor of order 1 is not positive definite
```

```
chol(fit5$cov)
```

```
## Error in chol.default(fit5$cov): the leading minor of order 1 is not positive definite
```

```
chol(fit6$cov)
```

```
##               mu       sigma          Q
## mu    0.09082912 -0.08105322  0.2359362
## sigma 0.00000000  0.12629652 -0.1600355
## Q     0.00000000  0.00000000  0.1321308
```

With survival

In case you fit the models with the **survival** package, you should use the original code from the Hoyle and Henley spreadsheet below:

```
chol(summary(fit1_survival)$var)
```

```
##             (Intercept)
## (Intercept)    0.104273
```

```
chol(summary(fit2_survival)$var)
```

```
##             (Intercept) Log(scale)
## (Intercept)  0.05790985 0.01753581
## Log(scale)   0.00000000 0.08144150
```

```
chol(summary(fit3_survival)$var)
```

```
##             (Intercept) Log(scale)
## (Intercept)  0.07206187 0.02748010
```

```
## Log(scale)   0.00000000 0.07278075
```

```
chol(summary(fit4_survival)$var)
```

```
##             (Intercept) Log(scale)
## (Intercept)  0.06486216 0.02030419
## Log(scale)   0.00000000 0.08452755
```

Known issues

For the example data, several parametric models of **flexsurv** do not converge to the maximum likelihood, where the the same models do with **survival** (e.g. the weibull model). This is a recognised issue which is due to different optimisation methods employed by **felxsurv** and **survival** (see here).