

# Package ‘survHE’

January 15, 2026

**Title** Survival Analysis in Health Economic Evaluation

**Version** 2.0.51

**URL** <https://github.com/giabaio/survHE>,  
<https://gianluca.statistica.it/software/survhe/>

**BugReports** <https://github.com/giabaio/survHE/issues>

**Description** Contains a suite of functions for survival analysis in health economics.

These can be used to run survival models under a frequentist (based on maximum likelihood) or a Bayesian approach (both based on Integrated Nested Laplace Approximation or Hamiltonian Monte Carlo). To run the Bayesian models, the user needs to install additional modules (packages), i.e. 'survHEinla' and 'survHEhmc'. These can be installed from

<<https://giabaio.r-universe.dev>> using  
'install.packages(``survHEhmc", repos = c(``<https://giabaio.r-universe.dev>", ``<https://cloud.r-project.org>"))'

and

'install.packages(``survHEinla", repos = c(``<https://giabaio.r-universe.dev>", ``<https://cloud.r-project.org>"))'

respectively. 'survHEinla' is based on the package INLA, which is available for download at

<<https://inla.r-inla-download.org/R/stable/>>. The user can specify a set of parametric models using a common notation and select the preferred mode of inference. The results can also be post-processed to produce probabilistic sensitivity analysis and can be used to export the output to an Excel file (e.g. for a Markov model, as often done by modellers and practitioners). <[doi:10.18637/jss.v095.i14](https://doi.org/10.18637/jss.v095.i14)>.

**License** GPL (>= 3)

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.2.9000

**Biarch** true

**Depends** methods, R (>= 4.1.0), flexsurv, dplyr, ggplot2

**Imports** rms, xlsx, tools, tibble, tidyverse

**Suggests** survHEinla, survHEhmc, INLA, rstan, testthat (>= 3.0.0)

**Config/testthat.edition** 3

**Additional\_repositories** <https://inla.r-inla-download.org/R/stable/>,  
<https://giabaio.r-universe.dev/>

**SystemRequirements** GNU make

**NeedsCompilation** no

**Author** Gianluca Baio [aut, cre],  
 Andrea Berardi [ctb],  
 Philip Cooney [ctb],  
 Andrew Jones [ctb],  
 Nathan Green [ctb]

**Maintainer** Gianluca Baio <g.baio@ucl.ac.uk>

**Repository** CRAN

**Date/Publication** 2026-01-15 15:30:02 UTC

## Contents

data	3
digitise	3
fit.models	4
make.ipd	7
make.surv	8
make.transition.probs	9
make_data_multi_state	10
make_newdata	12
markov_trace	13
model.fit.plot	14
msmdata	15
plot.survHE	16
plot_transformed_km	17
print.survHE	19
psa.plot	20
summary.survHE	21
ta174	23
theme_survHE	23
three_state_mm	24
write.surv	26

## Index

28

---

data	<i>A fictional survival trial.</i>
------	------------------------------------

---

## Description

A dataset containing fictional data from a trial, where the main outcome is in terms of time-to-event and censoring indicator and with additional covariates.

## Usage

```
data
```

## Format

A data frame with 367 rows and 8 variables:

**ID\_patient** The individual level identifier  
**time** The observed time at which the event happens  
**censored** An indicator to describe whether the event is fully observed or censored  
**arm** An indicator for the treatment arm, with 0 = control and 1 = active treatment  
**sex** An indicator for the individual's sex, with 0 = male and 1 = female  
**age** A numeric variable with the individual's age  
**imd** A categorical variable representing a measure of area-level social deprivation  
**ethnic** A categorical variable representing the individual's ethnic group, as measured from a Census

---

digitise	<i>Format digitised data for use in survival analysis</i>
----------	---

---

## Description

Produces txt files with Kaplan Meier and individual level survival data from digitised Kaplan Meier curves obtained by DigitizeIT

## Usage

```
digitise(  
  surv_inp,  
  nrisk_inp,  
  km_output = "KMdata.txt",  
  ipd_output = "IPDdata.txt"  
)
```

## Arguments

surv_inp	a txt file obtained for example by DigitizeIT and containing the input survival times from graph reading. This file contains 3 columns 'ID' = the row-ID 'time' = the vector of times captured by the digitisation process 'survival' = the vector of survival probabilities captured by the digitisation process
nrisk_inp	a txt file obtained by DigitizeIT and containing the reported number at risk. This contains the following columns: 'Interval' = the ID of the various intervals included in the analysis ( eg 1, 2, 3, ...) 'Time' = the actual time shown on the x-axis in the digitsed graph 'Lower' = the row of the extracted co-ordinates that the time corresponds to 'Upper' = the row of the extracted co-ordinates for which the time is less than the following time at which we have a number at risk 'nrisk' = the actual number at risk as specified in the original data
km_output	the name of the file to which the KM data will be written
ipd_output	the name of the file to which the individual level data data will be written

## Author(s)

Patricia Guyot and Gianluca Baio

## References

G Baio (2019). survHE: Survival analysis for health economic evaluation and cost-effectiveness modelling. Journal of Statistical Software (2020). vol 95, 14, 1-47. [doi:10.18637/jss.v095.i14](https://doi.org/10.18637/jss.v095.i14)

## Examples

```
## Not run:
# Defines the txt files to be used as inputs
surv.inp <- system.file("extdata", "survival.txt", package = "survHE")
nrisk.inp <- system.file("extdata", "nrisk.txt", package = "survHE")
# Runs 'digitise' to create the relevant output files
digitise(surv.inp, nrisk.inp)

## End(Not run)
```

## Description

Runs the survival analysis with several useful options, using either MLE (via flexsurv) or a Bayesian approach (via R-INLA or rstan)

## Usage

```
fit.models(formula = NULL, data, distr = NULL, method = "mle", ...)
```

## Arguments

formula	a formula specifying the model to be used, in the form <code>Surv(time, event)~treatment[+covariates]</code> for <code>flexsurv</code> , or <code>inla.surv(time, event)~treatment[+covariates]</code> for INLA
data	A data frame containing the data to be used for the analysis. This must contain data for the 'event' variable. In case there is no censoring, then event is a column of 1s.
distr	a (vector of) string(s) containing the name(s) of the model(s) to be fitted. Available options are: <code>flexsurv</code> : "exponential", "gamma", "genf", "gengamma", "gompertz", "weibull", "weibullPH", "loglogistic", "lognormal" INLA: "exponential", "weibull", "lognormal", "loglogistic" hmc: "exponential", "gamma", "genf", "gengamma", "gompertz", "weibull", "weibullPH", "loglogistic", "lognormal"
method	A string specifying the inferential method (' <code>mle</code> ', ' <code>inla</code> ' or ' <code>hmc</code> '). If <code>method</code> is set to ' <code>hmc</code> ', then <code>survHE</code> will write suitable model code in the Stan language (according to the specified distribution), prepare data and initial values and then run the model.
...	Additional options (for INLA or HMC). <b>INLA</b> specific options <code>dz</code> = defines the step length for the grid search over the hyperparameters space (default = 0.1) <code>diff.logdens</code> = defines the difference in the log-density for the hyperparameters to stop integration (default = 5) <code>control.fixed</code> = defines the default for the priors, unless specified by the user. Default values are <code>prior mean = 0</code> for <i>all</i> fixed effects <code>prior var = 1000</code> for <i>all</i> fixed effects <code>prior mean = 0</code> for the intercept <code>prior prec -&gt; 0</code> for the intercept <code>control.family</code> = a list of options. If <code>distr</code> is a vector, then can be provided as a named list of options, for example something like this: <code>control.family=list(weibull=list(param=c(.1,.1)),lognormal=list(initial=2))</code> the names of the elements of the list need to be the same as those given in the vector <code>distr</code> <b>HMC</b> specific options <code>chains</code> = number of chains to run in the HMC (default = 2) <code>iter</code> = total number of iterations (default = 2000) <code>warmup</code> = number of warmup iterations (default = <code>iter</code> /2) <code>thin</code> = number of thinning (default = 1) <code>control</code> = a list specifying Stan-related options, eg <code>control=list(adapt_delta=0.85)</code> (default = <code>NULL</code> ) <code>seed</code> = the random seed (to make things replicable) <code>pars</code> = a vector of parameters (string, default = <code>NA</code> ) <code>include</code> = a logical indicator (if <code>FALSE</code> , then the <code>pars</code> are not saved; default = <code>TRUE</code> ) <code>priors</code> = a list (of lists) specifying the values for the parameters of the prior distributions in the models <code>save.stan</code> = a logical indicator (default = <code>FALSE</code> ). If <code>TRUE</code> , then saves the data list for Stan and the model file(s)

## Details

On object in the class `survHE` containing the following elements

## Value

<code>models</code>	A list containing the fitted models. These contain the output from the original inference engine ( <code>flexsurv</code> , INLA or <code>rstan</code> ). Can be processed using the methods
---------------------	---

	specific to the original packages, or via survHE-specific methods (such as <code>plot</code> , <code>print</code> ) or other specialised functions (eg to extrapolate the survival curves, etc).
<code>model.fitting</code>	A list containing the output of the model-fit statistics (AIC, BIC, DIC). The AIC and BIC are estimated for all methods, while the DIC is only estimated when using Bayesian inference.
<code>method</code>	A string indicating the method used to fit the model, ie ' <code>mle</code> ', ' <code>inla</code> ' or ' <code>hmc</code> '.
<code>misc</code>	A list containing the time needed to run the model(s) (in seconds), the formula used, the results of the Kaplan-Meier analysis (which is automatically performed using <code>npsurv</code> ) and the original data frame.

## Author(s)

Gianluca Baio

## References

G Baio (2019). survHE: Survival analysis for health economic evaluation and cost-effectiveness modelling. Journal of Statistical Software (2020). vol 95, 14, 1-47. [doi:10.18637/jss.v095.i14](https://doi.org/10.18637/jss.v095.i14)

## See Also

`make.surv`

## Examples

```
## Not run:
# Loads an example dataset from 'flexsurv'
data(bc)

# Fits the same model using the 3 inference methods
mle = fit.models(formula=Surv(recyrs,censrec)~group,data=bc,
  distr="exp",method="mle")
inla = fit.models(formula=Surv(recyrs,censrec)~group,data=bc,
  distr="exp",method="inla")
hmc = fit.models(formula=Surv(recyrs,censrec)~group,data=bc,
  distr="exp",method="hmc")

# Prints the results in comparable fashion using the survHE method
print(mle)
print(inla)
print(hmc)

# Or visualises the results using the original packages methods
print(mle,original=TRUE)
print(inla,original=TRUE)
print(hmc,original=TRUE)

# Plots the survival curves and estimates
plot(mle)
plot(mle,inla,hmc,labs=c("MLE","INLA","HMC"),colors=c("black","red","blue"))
```

```
## End(Not run)
```

---

make.ipd

*Create an individual level dataset from digitised data*

---

## Description

Piles in the simulated IPD resulting from running digitise for more than one treatment arm

## Usage

```
make.ipd(ipd_files, ctr = 1, var.labs = c("time", "event", "arm"))
```

## Arguments

ipd_files	a list including the names of the IPD files created as output of digitise
ctr	the index of the file associated with the control arm (default, the first file). This will be coded as 0
var.labs	a vector of labels for the column of the resulting data matrix. NB these should match the arguments to the formula specified for fit.models. The user can specify values. These should be 4 elements (ID, TIME, EVENT, ARM)

## Author(s)

Gianluca Baio

## References

Something will go here

## See Also

Something will go here

## Examples

```
## Not run:  
## Defines the txt files to be used as inputs  
surv.inp <- system.file("extdata", "survival.txt", package = "survHE")  
nrisk.inp <- system.file("extdata", "nrisk.txt", package = "survHE")  
# Runs 'digitise' to create the relevant output files  
digitise(surv.inp, nrisk.inp, ipd_output = "IPD.txt")  
# Now uses 'make.ipd' to create the pseudo-data  
make.ipd("IPD.txt", ctr = 1, var.labs = c("time", "event", "arm"))  
  
## End(Not run)
```

---

**make.surv***Engine for Probabilistic Sensitivity Analysis on the survival curves*

---

**Description**

Creates the survival curves for the fitted model(s)

**Usage**

```
make.surv(fit, mod = 1, t = NULL, newdata = NULL, nsim = 1, ...)
```

**Arguments**

fit	the result of the call to the <code>fit.models</code> function, containing the model fitting (and other relevant information)
mod	the index of the model. Default value is 1, but the user can choose which model fit to visualise, if the call to <code>fit.models</code> has a vector argument for <code>distr</code> (so many models are fitted & stored in the same object)
t	the time vector to be used for the estimation of the survival curve
newdata	a list (of lists), specifying the values of the covariates at which the computation is performed. For example <code>list(list(arm=0),list(arm=1))</code> will create two survival curves, one obtained by setting the covariate <code>arm</code> to the value 0 and the other by setting it to the value 1. In line with <code>flexsurv</code> notation, the user needs to either specify the value for <i>all</i> the covariates or for none (in which case, <code>newdata=NULL</code> , which is the default). If some value is specified and at least one of the covariates is continuous, then a single survival curve will be computed in correspondence of the average values of all the covariates (including the factors, which in this case are expanded into indicators).
nsim	The number of simulations from the distribution of the survival curves. Default at <code>nsim=1</code> , in which case uses the point estimate for the relevant distributional parameters and computes the resulting survival curve
...	Additional options

**Author(s)**

Gianluca Baio

**References**

G Baio (2019). survHE: Survival analysis for health economic evaluation and cost-effectiveness modelling. *Journal of Statistical Software* (2020). vol 95, 14, 1-47. [doi:10.18637/jss.v095.i14](https://doi.org/10.18637/jss.v095.i14)

**See Also**

[fit.models](#), [psa.plot](#), [write.surv](#)

## Examples

```

## Not run:
# Loads an example dataset from 'flexsurv'
data(bc)

# Fits the same model using the 3 inference methods
mle <- fit.models(formula=Surv(recyrs,censrec) ~ group, data=bc,
                    distr="exp", method="mle")
p.mle <- make.surv(mle)
psa.plot(p.mle)

# Can also use the main 'plot' function to visualise the survival curves
# and include uncertainty by using a number 'nsim' of simulations
plot(mle, nsim=10)

## End(Not run)

```

---

make.transition.probs *make.transition.probs*

---

## Description

Computes the transition probabilities (to be passed to a Markov model) from the cumulative hazard curves obtained using `fit.models`, using the formula  $p(t)=1-\exp(H(t-k)/H(t))$ , where  $k$  is the Markov model cycle length (or the difference across two consecutive times) and  $t$  is a generic time

## Usage

```
make.transition.probs(fit, labs = NULL, ...)
```

## Arguments

<code>fit</code>	an object obtained as output of the call to <code>fit.models</code>
<code>labs</code>	a vector with labels to identify the 'profiles' ie the combination of covariates that have been passed onto the model formula. If 'NULL' (default), then figures it out from the 'survHE' object.
<code>...</code>	additional arguments. Includes the standard inputs to the call to <code>make.surv</code> , so <code>mod</code> (the index of the possibly many models stored in the 'survHE' object), <code>t</code> (the vector of times over which to compute the survival curves), <code>newdata</code> (a list that defines the profile of covariates) and <code>nsim</code> (the number of simulations to use - default is <code>nsim=1</code> )

## Value

A tibble 'lambda' with an indicator for the treatment arm, the times at which the probabilities have been computed and `nsim` columns each with a simulation of the transition probabilities for all the times specified by the user

**Note**

Something will go here

**Author(s)**

Gianluca Baio

**References**

Something will go here

**See Also**

[make.surv](#)

**Examples**

```
## Not run:
# Something will go here

## End(Not run)
```

---

**make\_data\_multi\_state** *make\_data\_multi\_state*

---

**Description**

Takes as input an individual-level dataset including data on both progression and death time (**jointly**) and manipulates it using `dplyr` functions to create a full "multi-state" dataset, in which all the transitions are tracked. This can then be used to fit survival models and compute all the estimates for the whole set of transition probabilities

**Usage**

```
make_data_multi_state(
  data,
  id = "id",
  prog = "prog",
  death = "death",
  prog_t = "prog_t",
  death_t = "death_t",
  keep = NULL,
  ...
)
```

## Arguments

data	dataset containing the full ILD with information on both progression and death. Can be a data.frame or a tibble
id	The column with the individual identifier. Can be NULL (in which case, it will be created from scratch)
prog	The progression indicator: takes value 1 if the individual has progressed and 0 otherwise. Defaults to the column named 'prog' in the dataset
death	The death indicator: takes value 1 if the individual has died and 0 otherwise. Defaults to the column named 'death' in the dataset
prog_t	The progression time. Defaults to the column named 'prog_t' in the dataset
death_t	The death time. Defaults to the column named 'death_t' in the dataset
keep	A vector of strings with the names of the additional variables from the original dataset to keep into the multistate dataset. If 'NULL' (default), then keeps all additional arguments.
...	

## Value

A tibble containing the event history for each individual and with the following variables: id = Patients ID; from = Initial state (1=Pre-progression, 2=Progression, 3=Death); to = End state (1=Pre-progression, 2=Progression, 3=Death); trans = Transition ID: 1=Pre-progression -> Progression; 2=Pre-Progression -> Death; 3=Progression -> Death; Tstart = Entry time (either entry or progression); Tstop = Exit time (time of event or censoring time); status = Event indicator (1=yes, 0=censored), **for the specific event under consideration**; treat = Treatment indicator All the other original variables are appended to these, but can be removed

## Note

Something will go here

## Author(s)

Gianluca Baio

## References

Something will go here

## See Also

Something will go here

## Examples

```
## Not run:
# Something will go here

## End(Not run)
```

---

make_newdata	<i>Creates a 'newdata' list to modify the plots for specific individual profiles (with respect to the covariates)</i>
--------------	---

---

## Description

Creates a 'newdata' list to modify the plots for specific individual profiles (with respect to the covariates)

## Usage

```
make_newdata(data, vars, conts = NULL)
```

## Arguments

data	The original dataset that has been used as input to the call to 'fit.models'
vars	A vector of strings, including the names of the variables that are to be used to construct specific profiles of individual covariates
conts	A subset of 'vars', which include the named covariates that are continuous. These will be averaged over, while for the remaining covariates (assumed to be factors), the specific profiles will be listed. Defaults to NULL

## Value

newdata	The list 'newdata' to be passed as optional argument to a call to the 'plot' method
labs	A vector of labels (say to use in the plot, for each profile)

## Note

Something will go here

## Author(s)

Gianluca Baio

## Examples

```
## Not run:
data(bc)

# Fits a model using the 'bc' data
mle = fit.models(formula=Surv(recyrs,censrec)~group,data=bc,
                 distr="exp",method="mle")
# Now makes the default plot
plot(mle)
# Now creates a 'newdata' list to modify the plot for selected profiles
newdata=make_newdata(data=bc,vars="group")
# And can plot, say, only two of the three treatment arms
```

```
plot(mle,newdata=newdata$newdata[c(1,3)],lab.profile=newdata$lab[c(1,3)])  
## End(Not run)
```

---

markov\_trace

*Markov trace*

---

## Description

Plots the Markov Trace from an object generated using `three_state_mm`

## Usage

```
markov_trace(mm, interventions = NULL, ...)
```

## Arguments

<code>mm</code>	The output of a call to <code>three_state_mm</code>
<code>interventions</code>	A vector of labels for the interventions
<code>...</code>	additional arguments.

## Value

Plot

## Note

Something will go here

## Author(s)

Gianluca Baio

## References

Something will go here

## See Also

`make.surv`, `three_state_mm`

## Examples

```
## Not run:  
# Something will go here  
  
## End(Not run)
```

---

model.fit.plot	<i>Graphical representation of the measures of model fitting based on Information Criteria</i>
----------------	--

---

## Description

Plots a summary of the model fit for all the models fitted

## Usage

```
model.fit.plot(..., type = "aic", scale = "absolute", stacked = FALSE)
```

## Arguments

...	Optional inputs. Must include at least one survHE object.
type	should the AIC, the BIC or the DIC plotted? (values = "aic", "bic" or "dic")
scale	If scale='absolute' (default), then plot the absolute value of the *IC. If scale='relative' then plot a rescaled version taking the percentage increase in the *IC in comparison with the best-fitting model
stacked	Should the bars be stacked and grouped by survHE object? (default=F)

## Details

Something will go here

## Value

A plot with the relevant model fitting statistics

## Author(s)

Gianluca Baio

## References

G Baio (2019). survHE: Survival analysis for health economic evaluation and cost-effectiveness modelling. Journal of Statistical Software (2020). vol 95, 14, 1-47. [doi:10.18637/jss.v095.i14](https://doi.org/10.18637/jss.v095.i14)

## See Also

`fit.models`

## Examples

```
## Not run:
data(bc)

mle = fit.models(formula=Surv(recyrs,censrec)~group,data=bc,
  distr=c("exp","wei","lno"),method="mle")
model.fit.plot(mle)

## End(Not run)
```

---

msmdata

*NICE TA174 dataset in multi-state format.*

---

## Description

These are the same data contained in NICE TA174, as made publicly available as part of the supplementary material for Williams et al (2017). Medical Decision Making, 37;427-439. However, the data have been restructured (by using the function `make_data_multi_state()`) to be used for multi-state analysis

## Usage

`msmdata`

## Format

A tibble with 1868 rows and 16 variables:

- id** A numeric patient identifier
- from** An indicator of the starting state. 1=Pre-progression; 2=Progression; 3=Death
- to** An indicator for the arriving state
- trans** A code for the actual transition considered. 1=Pre-progression -> Progression; 2=Pre-progression -> Death; 3=Progression -> Death
- Tstart** The time of entry into the observation
- Tstop** The time of exit from observation
- time** The observed time until even (progression or death), or censoring occurs
- status** The event indicator; takes value 1 if the underlying event (which varies depending on which transition is being considered) happens and 0 otherwise
- treat** The treatment indicator. 1=rituximab in combination with fludarabine andcyclophosphamide (RFC); 0=fludarabine and cyclo-phosphamide alone (FC)
- patid** The original numeric patient identifier
- prog** The original indicator to describe whether the patient has experience a progression
- death** The original indicator to describe whether the patient has experience death

**prog\_t** The original observed time at progression, or the time at which the patient has been censored; measured in months

**death\_t** The original observed time at death, or the time at which the patient has been censored; measured in months

**prog\_ty** The original observed time at progression, or the time at which the patient has been censored; measured in years

**death\_ty** The original observed time at death, or the time at which the patient has been censored; measured in years

---

plot.survHE

*Plot survival curves for the models fitted using fit.models*

---

## Description

Plots the results of model fit.

## Usage

```
## S3 method for class 'survHE'
plot(...)
```

## Arguments

...

Must include at least one result object saved as the call to the `fit.models` function. May include other optional parameters. These include whether the KM curve should be added `add.km` and whether the user specifies a profile of covariates (in the list `newdata`). Other possibilities are additional (mainly graphical) options. These are:

- `xlab` = a string with the label for the x-axis (default = "time")
- `ylab` = a string with the label for the y-axis (default = "Survival")
- `lab.profile` = a (vector of) string(s) indicating the labels associated with the strata defining the different survival curves to plot. Default to the value used by the Kaplan Meier estimate given in `fit.models`.
- `newdata` = a list (of lists) providing the values for the relevant covariates If `NULL`, then will use the mean values for the covariates if at least one is a continuous variable, or the combination of the categorical covariates.
- `xlim` = a vector determining the limits for the x-axis
- `colors` = a vector of characters defining the colours in which to plot the different survival curves
- `what` = a string indicating whether the survival, hazard or cumulative hazard curve should be plotted. Defaults to 'survival', but the other two options can be specified as 'hazard' or 'cumhazard'
- `lab.profile` = a vector of characters defining the names of the models fitted

- `add.km` = TRUE (whether to also add the Kaplan Meier estimates of the data)
- `annotate` = FALSE (whether to also add text to highlight the observed vs extrapolated data)
- `legend.position` = a vector of proportions to place the legend. Default to '`c(.75,.9)`', which means 75% across the x-axis and 90% across the y-axis
- `legend.title` = suitable instructions to format the title of the legend; defaults to '`element_text(size=15,face="bold")`' but there may be other arguments that can be added (using 'ggplot' facilities)
- `legend.text` = suitable instructions to format the text of the legend; defaults to '`element_text(colour="black", size=14, face="plain")`' but there may be other arguments that can be added (using 'ggplot' facilities)

### Author(s)

Gianluca Baio

### References

G Baio (2019). survHE: Survival analysis for health economic evaluation and cost-effectiveness modelling. Journal of Statistical Software (2020). vol 95, 14, 1-47. [doi:10.18637/jss.v095.i14](https://doi.org/10.18637/jss.v095.i14)

### See Also

[fit.models](#), [write.surv](#)

### Examples

```
## Not run:
data(bc)

mle = fit.models(formula=Surv(recyrs,censrec)~group,data=bc,
                 distr="exp",method="mle")
inla = fit.models(formula=Surv(recyrs,censrec)~group,data=bc,
                  distr="exp",method="inla")
plot(MLE=mle,INLA=inla)

## End(Not run)
```

---

plot\_transformed\_km     *Plot to assess suitability of parametric model*

---

### Description

Perform an exploratory investigation for linearity of transformed survival models.

**Usage**

```
plot_transformed_km(fit, mod = 1, add_legend = FALSE, graph = "base", ...)
```

**Arguments**

fit	An object of class survHE.
mod	Index or name of a model in fit. Defaults to 1.
add_legend	If TRUE, labels assumptions. Defaults to FALSE.
graph	Type of plot: base or ggplot2.
...	Further arguments, passed on to plot.

**Details**

For the Weibull, twice taking logs of the survivor function

$$\log(-\log S(t)) = \log \lambda + \gamma \log t$$

A plot of  $\log(-\log S(t))$  against  $\log(t)$  would give an approximately straight line if the Weibull assumption is reasonable. The plot could also be used to give a rough estimate of the parameters.

Similarly, for the log-logistic distribution

$$\log S(t)/(1 - S(t)) = \theta - \kappa \log t$$

For the log-normal distribution

$$\Phi^{-1}(1 - S(t)) = (\log t - \mu)/\sigma$$

We can also check the assumption made with using the Cox regression model of proportional hazards by inspecting the log-cumulative hazard plot.

$$\log H_i(t) = \beta x_i + \log H_0(t)$$

The transformed curves for different values of the explanatory variables will be parallel if PH holds.

**Value**

Diagnostic plot

**Author(s)**

William Browne, Nathan Green

**References**

Collett (2015) Modelling Survival Data in Medical Research, CRC Press

## Examples

```

data(bc)
form <- formula("Surv(recyrs, censrec) ~ group")

# exponential distribution
fit_exp <- fit.models(form, data = bc,
                      distr = "exp", method = "mle")
plot_transformed_km(fit_exp)
plot_transformed_km(fit_exp, graph = "ggplot2")

# weibull distribution
fit_wei <- fit.models(form, data = bc,
                      distr = "weibull", method = "mle")
plot_transformed_km(fit_wei)
plot_transformed_km(fit_wei, graph = "ggplot2")

# loglogistic distribution
fit_llog <- fit.models(form, data = bc,
                      distr = "loglogistic", method = "mle")
plot_transformed_km(fit_llog)
plot_transformed_km(fit_llog, graph = "ggplot2")

# lognormal distribution
fit_lnorm <- fit.models(form, data = bc,
                      distr = "lognormal", method = "mle")
plot_transformed_km(fit_lnorm)
plot_transformed_km(fit_lnorm, graph = "ggplot2")

## for only one group
form <- formula("Surv(recyrs, censrec) ~ 1")

fit_exp <- fit.models(form, data = bc,
                      distr = "exp", method = "mle")
plot_transformed_km(fit_exp)
plot_transformed_km(fit_exp, graph = "ggplot2")

```

---

print.survHE

*Print a summary of the survival model(s) fitted by fit.models*

---

## Description

Prints the summary table for the model(s) fitted, with the estimate of the parameters

## Usage

```

## S3 method for class 'survHE'
print(x, mod = 1, ...)

```

## Arguments

- x the survHE object (the output of the call to `fit.models`)
- mod is the index of the model. Default value is 1, but the user can choose which model fit to visualise, if the call to `fit.models` has a vector argument for `distr` (so many models are fitted & stored in the same object)
- ... additional options, including: `digits` = number of significant digits to be shown in the summary table (default = 6); `original` = a flag to say whether the *original* table from either `flexsurv` or `INLA` or `rstan` should be printed; `print_priors` = a flag to say whether the distributional assumptions for the HMC model (only)

## Author(s)

Gianluca Baio

## References

G Baio (2019). survHE: Survival analysis for health economic evaluation and cost-effectiveness modelling. *Journal of Statistical Software* (2020). vol 95, 14, 1-47. [doi:10.18637/jss.v095.i14](https://doi.org/10.18637/jss.v095.i14)

## Examples

```
## Not run:
data(bc)

mle = fit.models(formula=Surv(recyrs,censrec)~group,data=bc,
                 distr="exp",method="mle")
print(mle)

## End(Not run)
```

---

psa.plot

*Graphical depiction of the probabilistic sensitivity analysis for the survival curves*

---

## Description

Plots the survival curves for all the PSA simulations. The function is actually deprecated - similar graphs can be obtained directly using the `plot` method (with options), which allows a finer depiction of the results.

## Usage

```
psa.plot(psa, ...)
```

## Arguments

- psa the result of the call to the function `make.surv`  
 ... Optional graphical parameters, such as:
  - `xlab` = label for the x-axis
  - `ylab` = label for the y-axis
  - `col` = (vector) of colours for the lines to be plotted
  - `alpha` = the level of transparency for the curves (default = 0.2)

## Author(s)

Gianluca Baio

## References

G Baio (2019). survHE: Survival analysis for health economic evaluation and cost-effectiveness modelling. Journal of Statistical Software (2020). vol 95, 14, 1-47. [doi:10.18637/jss.v095.i14](https://doi.org/10.18637/jss.v095.i14)

## See Also

[make.surv](#), [write.surv](#)

## Examples

```
## Not run:
data(bc)

# Fits the same model using the 3 inference methods
mle = fit.models(formula=Surv(recyrs,censrec)~group,data=bc,
  distr="exp",method="mle")
p.mle = make.surv(mle,nsim=100)
psa.plot(p.mle)

## End(Not run)
```

---

<code>summary.survHE</code>	<i>Prints a summary table for the distribution the mean survival time for a given model and data</i>
-----------------------------	--

---

## Description

Calculates the mean survival time as the area under the survival curve

## Usage

```
## S3 method for class 'survHE'
summary(object, mod = 1, t = NULL, nsim = 1000, ...)
```

## Arguments

object	a survHE object (resulting from the call to <code>fit.models</code> )
mod	the model to be analysed (default = 1)
t	the vector of times to be used in the computation. Default = NULL, which means the observed times will be used. NB: the vector of times should be: i) long enough so that $S(t)$ goes to 0; and ii) dense enough so that the approximation to the AUC is sufficiently precise
nsim	the number of simulations from the survival curve distributions to be used (to compute interval estimates)
...	Additional options

## Details

A list comprising of the following elements

## Value

<code>mean.surv</code>	A matrix with the simulated values for the mean survival times
<code>tab</code>	A summary table

## Author(s)

Gianluca Baio

## References

G Baio (2019). survHE: Survival analysis for health economic evaluation and cost-effectiveness modelling. *Journal of Statistical Software* (2020). vol 95, 14, 1-47. [doi:10.18637/jss.v095.i14](https://doi.org/10.18637/jss.v095.i14)

## See Also

`fit.models`, `make.surv`

## Examples

```
## Not run:
data(bc)

mle = fit.models(formula=Surv(recyrs,censrec)~group,data=bc,
  distr="exp",method="mle")
summary(mle,nsim=100)

## End(Not run)
```

---

`ta174`

*NICE TA174 dataset.*

---

## Description

A dataset containing the data used for NICE TA174, as made publicly available as part of the supplementary material for Williams et al (2017). Medical Decision Making, 37;427-439.

## Usage

`ta174`

## Format

A tibble with 810 rows and 8 variables:

**patid** A numeric patient identifier

**treat** The treatment indicator. 1=rituximab in combination with fludarabine andcyclophosphamide (RFC); 0=fludarabine and cyclo-phosphamide alone (FC)

**prog** An indicator to describe whether the patient has experience a progression

**death** An indicator to describe whether the patient has experience death

**prog\_t** The observed time at progression, or the time at which the patient has been censored; measured in months

**death\_t** The observed time at death, or the time at which the patient has been censored; measured in months

**prog\_ty** The observed time at progression, or the time at which the patient has been censored; measured in years

**death\_ty** The observed time at death, or the time at which the patient has been censored; measured in years

---

`theme_survHE`

*A Custom ggplot2 Theme for Survival Plots*

---

## Description

This theme is designed for use with survival analysis plots, particularly those created using the `survHE` package. It builds on `theme_bw()` and customizes axis text, titles, plot background, and legend styling.

## Usage

`theme_survHE()`

## Details

Note: To position the legend inside the plot, use an additional call to `theme(legend.position = c(x, y), legend.justification = c("left", "top"))`.

## Value

A ggplot2 theme object that can be added to a ggplot.

## Examples

```
library(ggplot2)
library(survHE)
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  theme_survHE() +
  theme(legend.position = c(0.6, 0.8), legend.justification = c("left", "top"))
```

`three_state_mm`

*three\_state\_mm*

## Description

General purpose function to run a standard three-state Markov model (typically used in cancer modelling). The states are typically 'Pre-progression', 'Progressed' and 'Death'. No backward transition from 'Progressed' to 'Pre-progression' is allowed and 'Death' is obviously an absorbing state. All other transitions are possible. The crucial assumption is that *individual-level data* are available recording an indicator and the time of progression and death for each individual. The function returns the full transition matrix

## Usage

```
three_state_mm(
  m_12,
  m_13,
  m_23,
  nsim = 1,
  start = c(1000, 0, 0),
  basecase = FALSE,
  ...
)
```

## Arguments

`m_12` A 'survHE' object (output to a call to `fit.models`) estimating the parameters of a model for the transition from 'Pre-progression' (state 1) to 'Progressed' (state 2). Given the individual level data with the complete event history (in the object 'data'), can be done with a call like `'x=make_data_multi_state(data)'` and then `fit.models(Surv(time, status)~..., data=x %>% filter(trans==1), ...)`

m_13	A 'survHE' object (output to a call to <code>fit.models</code> ) estimating the parameters of a model for the transition from 'Pre-progression' (state 1) to 'Death' (state 3). Given the individual level data with the complete event history (in the object 'data'), can be done with a call like <code>'x=make_data_multi_state(data)'</code> and then <code>fit.models(Surv(time,status)~...,data=x %&gt;% filter(trans==2),...)</code>
m_23	A 'survHE' object (output to a call to <code>fit.models</code> ) estimating the parameters of a model for the transition from 'Progressed' (state 2) to 'Death' (state 3). Given the individual level data with the complete event history (in the object 'data'), can be done with a call like <code>'x=make_data_multi_state(data)'</code> and then <code>fit.models(Surv(time,status)~...,data=x %&gt;% filter(trans==3),...)</code>
nsim	The number of simulations for the model parameters that are used to compute the survival curves. Defaults to <code>nsim=1</code> , which simply creates one survival curve for each treatment arm.
start	A vector of initial state occupancy. By default assumes 1000 individuals, all initially allocated to 'Pre-progression'
basecase	Should the base case be computed as well, based on the point estimate of the underlying model parameters? (Default=FALSE)
...	additional arguments.

### Value

A list including the state occupancy simulations in an object 'm'. This is a tibble with the number of individuals in each of the 3 states at each of the times specified by the user. If `nsim>1`, then the tibble also contains a simulation index to keep track of that. The list also includes the computation time to obtain the state occupancy tibble (in the object 'running\_time'). If `basecase==TRUE`, then the function also computes the "base case scenario" (based on 1 simulation from of the underlying survival curves, i.e. the point estimate of the model parameters) and stores it in the object 'base\_case'

### Note

Something will go here

### Author(s)

Gianluca Baio

### References

Something will go here

### See Also

`make.transition.probs` `make_data_multi_state`

## Examples

```
## Not run:
# Something will go here

## End(Not run)
```

---

write.surv

*write.surv*

---

## Description

Writes the survival summary to an excel file (helpful to then call the values in the Markov model)

## Usage

```
write.surv(object, file, sheet = NULL, what = "surv")
```

## Arguments

object	a <code>summary.flexsurvreg</code> object containing the survival curves (with times, estimates and interval limits)
file	a string with the full path to the file name to be saved
sheet	a string with the name of the sheet to be created
what	a string to describe what to be exported. Can either be 'surv' (default), which outputs the simulation(s) for the survival curves or 'sim', which outputs the simulation(s) for the underlying model parameters. If there are several 'profiles', they get written in separate spreadsheets and a clear indication is given as the name of the spreadsheet

## Details

Something will go here

## Value

A spreadsheet file with the simulation(s) of the relevant quantity

## Author(s)

Gianluca Baio

## References

G Baio (2019). survHE: Survival analysis for health economic evaluation and cost-effectiveness modelling. *Journal of Statistical Software* (2020). vol 95, 14, 1-47. [doi:10.18637/jss.v095.i14](https://doi.org/10.18637/jss.v095.i14)

**See Also**`make.surv`**Examples**

```
## Not run:  
# Loads an example dataset from 'flexsurv'  
data(bc)  
  
# Fits the same model using the 3 inference methods  
mle = fit.models(formula=Surv(recyrs,censrec)~group,data=bc,  
      distr="exp",method="mle")  
p.mle = make.surv(mle)  
write.surv(p.mle,file="test.xlsx")  
  
## End(Not run)
```

# Index

- \* **Approximation**
  - fit.models, 4
- \* **Bayesian**
  - fit.models, 4
- \* **Bootstrap**
  - make.surv, 8
  - psa.plot, 20
- \* **Carlo**
  - fit.models, 4
- \* **Digitized**
  - digitise, 3
  - make.ipd, 7
- \* **Excel**
  - write.surv, 26
- \* **Hamiltonian**
  - fit.models, 4
- \* **Integrated**
  - fit.models, 4
- \* **Kaplan**
  - digitise, 3
  - make.ipd, 7
- \* **Laplace**
  - fit.models, 4
- \* **Markov**
  - make.transition.probs, 9
  - make\_data\_multi\_state, 10
  - markov\_trace, 13
  - three\_state\_mm, 24
- \* **Mean**
  - summary.survHE, 21
- \* **Meier**
  - digitise, 3
  - make.ipd, 7
- \* **Model**
  - model.fit.plot, 14
- \* **Monte**
  - fit.models, 4
- \* **Multistate**
  - make\_data\_multi\_state, 10
- \* **Nested**
  - fit.models, 4
- \* **PSA**
  - write.surv, 26
- \* **Parametric**
  - fit.models, 4
  - make\_newdata, 12
  - model.fit.plot, 14
  - plot.survHE, 16
  - print.survHE, 19
  - summary.survHE, 21
- \* **Probabilistic**
  - make.surv, 8
  - psa.plot, 20
- \* **Survival**
  - make.surv, 8
  - psa.plot, 20
- \* **Three-state**
  - three\_state\_mm, 24
- \* **Transition**
  - make.transition.probs, 9
  - make\_data\_multi\_state, 10
  - markov\_trace, 13
  - three\_state\_mm, 24
- \* **analysis**
  - make.surv, 8
  - psa.plot, 20
- \* **cancer**
  - three\_state\_mm, 24
- \* **curve**
  - digitise, 3
  - make.ipd, 7
- \* **datasets**
  - data, 3
  - msmdata, 15
  - ta174, 23
- \* **fitting**
  - model.fit.plot, 14
- \* **hplot**

plot\_transformed\_km, 17  
\* **inference**  
  fit.models, 4  
\* **models**  
  fit.models, 4  
  make.surv, 8  
  make.transition.probs, 9  
  make\_data\_multi\_state, 10  
  make\_newdata, 12  
  markov\_trace, 13  
  model.fit.plot, 14  
  msmdata, 15  
  plot.survHE, 16  
  plot\_transformed\_km, 17  
  print.survHE, 19  
  psa.plot, 20  
  summary.survHE, 21  
  three\_state\_mm, 24  
\* **model**  
  three\_state\_mm, 24  
\* **probabilities**  
  make.transition.probs, 9  
  make\_data\_multi\_state, 10  
  markov\_trace, 13  
  three\_state\_mm, 24  
\* **sensitivity**  
  make.surv, 8  
  psa.plot, 20  
\* **survival**  
  fit.models, 4  
  make\_newdata, 12  
  model.fit.plot, 14  
  plot.survHE, 16  
  plot\_transformed\_km, 17  
  print.survHE, 19  
  summary.survHE, 21  
\* **time**  
  summary.survHE, 21  
\* **trace**  
  markov\_trace, 13  
\* **via**  
  fit.models, 4

data, 3  
digitise, 3

fit.models, 4, 8, 17

make.ipd, 7  
make.surv, 8, 10, 21  
make.transition.probs, 9

make\_data\_multi\_state, 10  
make\_newdata, 12  
markov\_trace, 13  
model.fit.plot, 14  
msmdata, 15

plot.survHE, 16  
plot\_transformed\_km, 17  
print.survHE, 19  
psa.plot, 8, 20

summary.survHE, 21

ta174, 23  
theme\_survHE, 23  
three\_state\_mm, 24

write.surv, 8, 17, 21, 26