Package ‘expertsurv’

January 21, 2026

Title Incorporate Expert Opinion with Parametric Survival Models

Version 1.4.1

Date 2026-01-21

Description Enables users to incorporate expert opinion with parametric survival analysis using a Bayesian or frequentist approach. Expert Opinion can be provided on the survival probabilities at certain time-point(s) or for the difference in mean survival between two treatment arms. Please reference it's use as Cooney, P., White, A. (2023) <doi:10.1177/0272989X221150212>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.2

Biarch true

Depends R (>= 3.5.0), survival

Imports abind, assertthat, dplyr, generics, ggplot2, magrittr (>= 2.0), Matrix, numDeriv, purrr, Rcpp (>= 0.12.0), Rdpack, rlang, rms, rstan (>= 2.26.0), rstantools (>= 2.2.0), rstpm2, scales, SHELF, statmod, stats, stringr, tibble, tidyr, tidyselect, utils, broom, sn, muhaz, mvtnorm, quadprog,

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppArmadillo, RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.26.0), splines2, StanHeaders (>= 2.26.0)

SystemRequirements GNU make

Suggests rmarkdown, bookdown, knitr, rjags, R2jags, loo, xlsx, shiny, shinyWidgets, shinycssloaders, shinyjs, shinyMatrix, shinybusy, ggpubr, splines2, survminer, colorspace, testthat, covr, eha

VignetteBuilder knitr

RdMacros Rdpack

Config/testthat/edition 3

NeedsCompilation yes

Author Philip Cooney [aut, cre] (ORCID:
<https://orcid.org/0000-0001-9649-0195>),
Arthur White [ths] (ORCID: <https://orcid.org/0000-0002-7268-5163>)

Maintainer Philip Cooney <phcooney@tcd.ie>

Repository CRAN

Date/Publication 2026-01-21 20:20:02 UTC

Contents

`expertsurv-package` *Incorporating Expert Opinion with Parametric Survival Models*

Description

Contains functions to include expert opinion with the parametric models commonly used in health economic modelling. Theoretical details are described elsewhere (Cooney and White 2023). Borrows many functions from the `survHE` package (Baio 2020) and `flexsurv` package (Jackson 2016).

Details

Package:	expertsurv
Type:	Package
Version:	1.4.0
Date:	2023-09-22
License:	MIT + file LICENSE

LazyLoad: yes

Integrate expert opinions on survival and mean differences in survival with common parametric survival models using either a Bayesian or frequentist framework.

Author(s)

Philip Cooney Package Creator, Maintainer

Arthur White Thesis Supervisor

References

P Cooney (2023). expertsurv: Incorporating expert opinion into parametric survival models.

Baio G (2020). "survHE: Survival Analysis for Health Economic Evaluation and Cost-Effectiveness Modeling." *Journal of Statistical Software*, **95**(14), 1–47. doi:10.18637/jss.v095.i14. Cooney P, White A (2023). "Direct Incorporation of Expert Opinion into Parametric Survival Models to Inform Survival Extrapolation." *Medical Decision Making*, **1**(1), 0272989X221150212. doi:10.1177/0272989X221150212, PMID: 36647200, https://doi.org/10.1177/0272989X221150212, `https://doi.org/10.1177/0272989X221150212`. Jackson C (2016). "flexsurv: A Platform for Parametric Survival Modeling in R." *Journal of Statistical Software*, **70**(8), 1–33. doi:10.18637/jss.v070.i08.

Examples

```
#Define expert opinion
require("dplyr")
param_expert_example1 <- list()
#1 timepoint and 2 experts with equal weight,
#first a normal distribution, second a non-standard t-distribution with
#3 degrees of freedom

param_expert_example1[[1]] <- data.frame(dist = c("norm","t"),
                               wi = c(0.5,0.5), # Ensure Weights sum to 1
                               param1 = c(0.1,0.12),
                               param2 = c(0.05,0.05),
                                param3 = c(NA,3))


timepoint_expert <- 14

data2 <- data %>% rename(status = censored) %>% mutate(time2 = ifelse(time > 10, 10, time),
                  status2 = ifelse(time> 10, 0, status))

example1  <- fit.models.expert(formula=Surv(time2,status2)~1,data=data2,
                              distr=c("wph", "gomp"),
                              method="mle",
                              pool_type = "log pool",
                              opinion_type = "survival",
                              times_expert = timepoint_expert,
                              param_expert = param_expert_example1)
```

```
#Visualize the goodness of fit
model.fit.plot(example1, type = "aic")
#Visualize the survival curve
plot(example1, add.km = TRUE, t = 0:30)
```

`bc` *Breast cancer survival data*

Description

Survival times of 686 patients with primary node positive breast cancer. See **flexsurv** for details.

Usage

```
bc
```

Format

An object of class `data.frame` with 686 rows and 4 columns.

`bos` *Bronchiolitis obliterans syndrome after lung transplants*

Description

See `flexsurv` for details.

Format

A data frame containing a sequence of observed or censored transitions to the next stage of severity or death. It is grouped by patient and includes histories of 204 patients. All patients start in state 1 (no BOS) at six months after transplant, and may subsequently develop BOS or die.

Source

Papworth Hospital, U.K.

References

Heng. D. et al. (1998). Bronchiolitis Obliterans Syndrome: Incidence, Natural History, Prognosis, and Risk Factors. Journal of Heart and Lung Transplantation 17(12)1255–1263.

`compiled_models_saved` *Object with Compiled Stan Code*

Description

Pre-compiled stan models that can be generated by the compile_stan function. These compiled stan models can be supplied to the `fit.models.expert` function with the argument `compile_mods`. This greatly facilitates the estimation of the parametric survival models run by stan.

Source

Generated from compile_stan

`compile_stan` *Compile Specified Stan Models for Bayesian Survival Analysis*

Description

The `compile_stan` function pre-compiles specified Stan models used in the `expertsurv` package for Bayesian survival analysis. By compiling the models ahead of time, you can significantly reduce computation time during model fitting, as the models won't need to be compiled each time they're used.

Usage

```
compile_stan(dist_stan = c("exp", "wei", "wph", "rps", "llo", "lno"))
```

Arguments

`dist_stan`	A character vector specifying the distributions to compile. Options include: `"exp"` Exponential distribution `"wei"` Weibull distribution `"wph"` Weibull Proportional Hazards `"rps"` Restricted Piecewise Survival `"llo"` Log-Logistic distribution `"lno"` Log-Normal distribution Defaults to `c("exp", "wei", "wph", "rps", "llo", "lno")`.

Details

Pre-compiling Stan models is recommended when working with Bayesian methods in survival analysis, as it avoids the overhead of compiling models during each function call. This is particularly beneficial when running multiple models or iterations.

The function internally calls `rstan::stan_model()` for each specified distribution, compiling the Stan code associated with that model.

Value

A named list of compiled Stan models corresponding to the specified distributions.

See Also

compile_stan

Examples

```
library(dplyr)
# Prepare the data
# Assume 'data' is your dataset containing 'time' and 'censored' variables
data2 <- data %>%
  rename(status = censored) %>%
  mutate(
    time2 = ifelse(time > 10, 10, time),
    status2 = ifelse(time > 10, 0, status)
  )

# Pre-compile Stan models (ideally after installing the package)
# This step can be time-consuming but only needs to be done once per session
compiled_models_saved <- compile_stan()

# Fit the survival models using Bayesian methods
example1 <- fit.models.expert(
  formula = Surv(time2, status2) ~ 1,
  data = data2,
  distr = c("wei", "gom"),  # Weibull and Gompertz distributions
  method = "bayes",
  compile_mods = compiled_models_saved)

# Examine the results
summary(example1)
plot(example1)
```

`cred_int` *Credible interval for pooled distribution*

Description

Returns the interval based on defined quantiles. The approach used only provides an approximate (although quite accurate) integral.

Usage

```
cred_int(plt_obj, val = "linear pool", interval = c(0.025, 0.975))
```

Arguments

`plt_obj`	A plot object from `plot_expert_opinion`.
`val`	The name of the opinion for which the interval will be generated.
`interval`	A vector of the upper and lower probabilities. Default is the standard 95% interval

Value

Credible interval based on the pooled distribution

Examples

```
param_expert_example1 <- list()
param_expert_example1[[1]] <- data.frame(dist = c("norm","t"),
    wi = c(0.5,0.5), # Ensure Weights sum to 1
    param1 = c(0.1,0.12),
    param2 = c(0.005,0.005),
    param3 = c(NA,3))

plot_opinion1<- plot_expert_opinion(param_expert_example1[[1]],
              weights = param_expert_example1[[1]]$wi)
cred_int(plot_opinion1,val = "linear pool", interval = c(0.025, 0.975))
```

`data` *A fictional survival trial taken directly from survHE.*

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

An object of class `data.frame` with 367 rows and 8 columns.

References

Baio G (2020). "survHE: Survival Analysis for Health Economic Evaluation and Cost-Effectiveness Modeling." *Journal of Statistical Software*, **95**(14), 1–47. doi:10.18637/jss.v095.i14.

`elicit_surv` *Elicit survival judgements interactively and estimate survival models*

Description

Opens up a web browser (using the shiny package), from which you can specify judgements and fit distributions for multiple timepoints and experts. Plots of the fitted density functions are provided overlayed on the survival data (where appropriate).

Usage

```
elicit_surv(compile_mods = NULL)
```

Arguments

`compile_mods`	list of compiled stan models generated by compile_stan. Supplying the compiled stan models will greatly speed up the computation of the Bayesian analysis (otherwise each time a stan model is run it will be compiled (and not reused between runs)).

Details

Once the elicitation is complete the analysis can be run. Click "Download R objects" to download the `expertsurv` object generated from the analysis. Click "Download report" to generate a report including plots and parameter values for the parametric survival models. For detailed instructions use `browseVignettes("expertsurv")`

Value

If "Download R objects" selected an `expertsurv` object containing the results of the elicitation and analysis. The object includes:

- `model`: The fitted survival models.
- `parameters`: The estimated parameters for each model.

If "Download report" selected either .html, .pdf or .docx document is downloaded with:

- `plot`: Plots of the fitted survival overlayed on the Kaplan Meier data and plot of expert opinion as probability distributions.
- `summary`: A summary report of the analysis including goodness of fit and parameter values.

Author(s)

Philip Cooney phcooney@tcd.ie

Examples

```
if (interactive()) {
elicit_surv()
}
```

`fit.models.expert` *Fit Parametric Survival Models Incorporating Expert Opinion*

Description

The `fit.models.expert` function extends the capabilities of the `survHE` package by allowing users to fit parametric survival models that incorporate expert opinion. Expert opinions can be on survival probabilities at specific time points or on expected differences in survival between groups. This function is particularly useful when empirical data is scarce or incomplete, and expert knowledge can help inform the analysis.

Usage

```
fit.models.expert(
  formula = NULL,
  data,
  distr = NULL,
  method = "bayes",
  opinion_type = "survival",
  param_expert = NULL,
  ...
)
```

Arguments

`formula` An object of class `formula` specifying the survival model to be fitted, as per `fit.models` in the **survHE** package. The left-hand side must be a `Surv` object, and the right-hand side specifies the covariates.

`data` A data frame containing the variables specified in the `formula`, as per `fit.models`.

`distr` A character vector specifying the distribution(s) to be used for the survival model(s), as per `fit.models`. Options include, but are not limited to:

- `"exp"`: Exponential distribution
- `"wei"`: Weibull distribution
- `"gom"`: Gompertz distribution
- `"gengamma"`: Generalized Gamma distribution
- `"genf"`: Generalized F distribution (not available for `method = "bayes"`)
- `"rps"`: Royston-Parmar spline model (not available with `opinion_type = "mean"`)

Note: The Generalized F model is not available when `method = "bayes"`, and the Royston-Parmar model is not available with expert opinion on the mean survival.

`method` The estimation method to be used. Options are:

- `"mle"`: Maximum Likelihood Estimation
- `"bayes"`: Bayesian estimation using either Stan or JAGS

Note: The `"inla"` method is not included. For Bayesian analysis, specify `method = "bayes"` (do not use `"hmc"` as in **survHE**).

`opinion_type` A character string specifying the type of expert opinion provided:

- `"survival"`: Expert opinion on the survival function at specific time points
- Other values (e.g., `"mean"`): Expert opinion on differences in expected survival (area under the survival curve)

`param_expert` A list where each element corresponds to a time point (if applicable) and contains a data frame of expert opinions. Each data frame should have the following columns, with each row representing an expert:

`dist` Name of the distribution assigned to each expert's opinion. Options include `"norm"`, `"t"`, `"lnorm"`, `"gamma"`, `"beta"`.

`wi` Weight of the expert's opinion. Weights must sum to 1 across all experts for each time point.

`param1` First parameter of the specified distribution (e.g., mean for normal distribution). Parameters follow the conventions of the **SHELF** package.

`param2` Second parameter of the specified distribution (e.g., standard deviation for normal distribution).

`param3` Third parameter of the distribution, if applicable (e.g., degrees of freedom for the t-distribution); otherwise, set to `NA`.

`...` Other arguments required depending on the analysis. Important ones include:

`id_St` Required if the model includes covariates (e.g., treatments) and expert opinion on survival probabilities. Specifies the row number in the data frame representing the covariate pattern for which the expert opinion is provided.

`id_trt` Required if including expert opinion about differences in expected survival. Specifies the row number representing the treatment group.

`id_comp` Required if including expert opinion about differences in expected survival. Specifies the row number representing the comparator group.

`times_expert` A numeric vector of time points at which expert opinion on survival probabilities is provided.

`compile_mods` For Bayesian analysis, a list of pre-compiled Stan models can be supplied to speed up computation. Pre-compiling models is recommended and can be done using `compile_stan`.

Details

This function enables the integration of expert opinion into parametric survival models. Expert opinion can be particularly valuable when data is limited or censored, as it allows for informed estimates of survival functions or differences between treatment groups.

The function supports both Maximum Likelihood Estimation (MLE) and Bayesian methods. For Bayesian estimation, models are fitted using Stan or JAGS, depending on the distribution. Pre-compiling Stan models using `compile_stan` is highly recommended to reduce computation time.

Value

An object of class `expertsurv` containing the fitted models, parameter estimates, and other relevant information. This object can be used with plotting and summary functions for further analysis.

Examples

```
## Not run:
library(dplyr)

# Example 1: Incorporating Expert Opinion on Survival Probabilities Using MLE

# Define expert opinion as a normal distribution centered at 0.1 with sd 0.05
param_expert_example1 <- list()
param_expert_example1[[1]] <- data.frame(
  dist = "norm",
  wi = 1,  # Ensure weights sum to 1 across experts
  param1 = 0.1,
  param2 = 0.05,
  param3 = NA
)

# Time point at which expert opinion is provided
timepoint_expert <- 14  # For example, 14 months

# Prepare the data
# Assume 'data' is your dataset containing 'time' and 'censored' variables
data2 <- data %>%
  rename(status = censored) %>%
  mutate(
    time2 = ifelse(time > 10, 10, time),
    status2 = ifelse(time > 10, 0, status)
  )

# Fit the survival models using MLE, incorporating expert opinion
example1 <- fit.models.expert(
  formula = Surv(time2, status2) ~ 1,
  data = data2,
  distr = c("wei", "gom"),  # Weibull and Gompertz distributions
  method = "mle",
  opinion_type = "survival",
  times_expert = timepoint_expert,
  param_expert = param_expert_example1
)

# Plot the fitted survival curves along with the Kaplan-Meier estimate
plot(example1, add.km = TRUE, t = 0:20)

# Compare models using Akaike Information Criterion (AIC)
model.fit.plot(example1, type = "aic")

# Example 2: Incorporating Expert Opinion Using Bayesian Estimation

# Pre-compile Stan models (ideally after installing the package)
# This step can be time-consuming but only needs to be done once per session
compiled_models_saved <- compile_stan()

# Fit the survival models using Bayesian estimation with expert opinion
```

```
example1_bayes <- fit.models.expert(
  formula = Surv(time2, status2) ~ 1,
  data = data2,
  distr = c("wei", "gom"),
  method = "bayes",
  opinion_type = "survival",
  times_expert = timepoint_expert,
  param_expert = param_expert_example1,
  iter = 2000,  # Set to a high number for convergence (e.g., 2000 or more)
  compile_mods = compiled_models_saved
)

# Summarize the Bayesian model results
summary(example1_bayes)

# Plot the Bayesian fitted survival curves
plot(example1_bayes, add.km = TRUE, t = 0:20)

## End(Not run)
```

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

Description

Creates the survival curves for the fitted model(s) - Original code from `survHE`

Usage

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

Arguments

`fit`	the result of the call to the `fit.models` 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 fit.models has a vector argument for distr (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), specifiying the values of the covariates at which the computation is performed. For example `list(list(arm=0),list(arm=1))` will create two survival curves, one obtained by setting the covariate `arm` to the value 0 and the other by setting it to the value 1. In line with `flexsurv` notation, the user needs to either specify the value for *all* the covariates or for none (in which case, `newdata=NULL`, 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 `nsim=1`, in which case uses the point estimate for the relevant distributional parameters and computes the resulting survival curve
`...`	Additional options

Value

A list with survival times for the fitted models

Author(s)

Gianluca Baio

References

Baio G (2020). "survHE: Survival Analysis for Health Economic Evaluation and Cost-Effectiveness Modeling." *Journal of Statistical Software*, **95**(14), 1–47. doi:10.18637/jss.v095.i14.

See Also

`psa.plot`(for example)

`model.fit.plot` *Graphical representation of the measures of model fitting based on Information Criteria*

Description

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

Usage

```
model.fit.plot(..., type = "dic")
```

Arguments

`...`	Optional inputs. Must include an `expertsurv` object.
`type`	should the DIC, WAIC, PML be plotted (AIC, BIC also allowed but only valid for frequentist approach).

Value

A plot with the relevant model fitting statistics plotted in order of fit.

Examples

```
require("dplyr")
param_expert_example1 <- list()
param_expert_example1[[1]] <- data.frame(dist = c("norm"),
                                         wi = c(1), # Ensure Weights sum to 1
                                         param1 = c(0.1),
                                         param2 = c(0.05),
                                         param3 = c(NA))
timepoint_expert <- 14 # Expert opinion at t = 14


data2 <- expertsurv::data %>% rename(status = censored) %>%
mutate(time2 = ifelse(time > 10, 10, time),
status2 = ifelse(time> 10, 0, status))
example1  <- fit.models.expert(formula=Surv(time2,status2)~1,data=data2,
                              distr=c("wei", "gomp"),
                              method="mle",
                              pool_type = "linear pool",
                              opinion_type = "survival",
                              times_expert = timepoint_expert,
                              param_expert = param_expert_example1)


model.fit.plot(example1, type = "aic")
```

`plot.expertsurv` *Plot survival curves for the models fitted using* `fit.models`

Description

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

Usage

```
## S3 method for class 'expertsurv'
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: • `add.km`: Whether the KM curve should be added. • `newdata`: Specifies a profile of covariates (in the list `newdata`). Other possibilities are additional (mainly graphical) options: – `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. Defaults to the value used by the Kaplan Meier estimate given in `fit.models`.
  - `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.
  - `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 other arguments 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 other arguments can be added (using 'ggplot' facilities).
- `plot_opinion`: TRUE will provide an illustration of the expert opinion at each time-point.
- `plot_ci`: Statistical uncertainty can be plotted using the `plot_ci = TRUE` argument and by specifying `nsim` equal to the number of desired simulations (for Bayesian models, this must be less than the total number of simulations from the posterior). By default, the confidence/credible intervals are plotted as dashed lines. If an area/ribbon plot is preferred, set `ci_plot_ribbon = TRUE`.
- `nsim`: Even if statistical uncertainty is not required in the plots, it is recommended that `nsim` is set to a reasonable number. If `nsim = 1` by default, the maximum likelihood estimates or the posterior mean of the parameters will be used to plot the results. In most cases, this should suffice (particularly for maximum likelihood). However, the expected survival estimated by the full sampling distribution may be different from the estimate at its expectation/maximum likelihood estimate.

Value

A ggplot2 object of the survival curves.

Author(s)

Gianluca Baio

References

Baio G (2020). “survHE: Survival Analysis for Health Economic Evaluation and Cost-Effectiveness Modeling.” *Journal of Statistical Software*, **95**(14), 1–47. doi:10.18637/jss.v095.i14.

See Also

`fit.models.expert`

Examples

```
require("dplyr")
param_expert_example1 <- list()
param_expert_example1[[1]] <- data.frame(dist = c("norm","t"),
                                         wi = c(0.5,0.5), # Ensure Weights sum to 1
                                         param1 = c(0.1,0.12),
                                         param2 = c(0.15,0.5),
                                         param3 = c(NA,3))

timepoint_expert <- 14
data2 <- data %>% rename(status = censored) %>% mutate(time2 = ifelse(time > 10, 10, time),
                                                       status2 = ifelse(time> 10, 0, status))
example1_mle <- fit.models.expert(formula=Surv(time2,status2)~1,data=data2,
                                  distr=c("wph", "exp"),
                                  method="mle",
                                  pool_type = "log pool",
                                  opinion_type = "survival",
                                  times_expert = timepoint_expert,
                                  param_expert = param_expert_example1)


plot(example1_mle, add.km = TRUE, t = 0:30,plot_opinion = TRUE)
```

`plot_expert_opinion` *Plotting Pooled Expert Opinion*

Description

Returns a ggplot with the individual expert opinions along with the pooled distributions (both linear and logarithmic).

Usage

```
plot_expert_opinion(
  object,
  xl_plt = NULL,
  xu_plt = NULL,
  weights = NULL,
  St_indic = 0
)
```

Arguments

`object`	Either a object of class elicitation (from `SHELF`) or a dataframe with parameters of the distribution (see Example below).
`xl_plt`	Optionally set the lower bound for the plot
`xu_plt`	Optionally set the upper bound for the plot
`weights`	A vector with the weight of each expert. If omitted, set to equal weights.
`St_indic`	Set to 1 if you want to truncate the distributions to be between 0 and 1.

Value

A ggplot with pooled distributions.

Examples

```
 expert_df <- data.frame(dist = c("norm","t"), #Distribution Name
                         wi = c(1/3,2/3), #Expert weights
                         param1 = c(0.3,0.40), #Parameter 1
                         param2 = c(0.05,0.05),# Parameter 2
                         param3 = c(NA,3)) #Parameter 3: Only t-distribution

plot_expert_opinion(expert_df , weights = expert_df$wi)
```

`print.expertsurv` *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 - ported from `survHE`.

Usage

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

Arguments

`x`	the `expertsurv` 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 `rstan/JAGS` should be printed

Value

Printed message (no object returned) providing estimates of the survival models.

Author(s)

Gianluca Baio

References

Baio G (2020). "survHE: Survival Analysis for Health Economic Evaluation and Cost-Effectiveness Modeling." *Journal of Statistical Software*, **95**(14), 1–47. doi:10.18637/jss.v095.i14.

Examples

```
require("dplyr")
param_expert_example1 <- list()
param_expert_example1[[1]] <- data.frame(dist = c("norm","t"),
                                         wi = c(0.5,0.5), # Ensure Weights sum to 1
                                       param1 = c(0.1,0.12),
                                      param2 = c(0.15,0.5),
                                       param3 = c(NA,3))
timepoint_expert <- 14
data2 <- data %>% rename(status = censored) %>% mutate(time2 = ifelse(time > 10, 10, time),
                                                   status2 = ifelse(time> 10, 0, status))
mle = example1 <- fit.models.expert(formula=Surv(time2,status2)~1,data=data2,
                   distr=c("wph", "gomp"),
                   method="mle",
                   pool_type = "log pool",
                   opinion_type = "survival",
                   times_expert = timepoint_expert,
                   param_expert = param_expert_example1)
print(mle)
```

`psa.plot` *Graphical depiction of the probabilistic sensitivity analysis for the survival curves - ported from* `survHE`

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 colors for the lines to be plotted `alpha` = the level of transparency for the curves (default = 0.2)

Value

ggplot2 object of the survival curve including parameter uncertainty

Author(s)

Gianluca Baio

References

Baio G (2020). "survHE: Survival Analysis for Health Economic Evaluation and Cost-Effectiveness Modeling." *Journal of Statistical Software*, **95**(14), 1–47. doi:10.18637/jss.v095.i14.

Examples

```
require("dplyr")
param_expert_example1 <- list()
param_expert_example1[[1]] <- data.frame(dist = c("norm","t"),
                                         wi = c(0.5,0.5), # Ensure Weights sum to 1
                                         param1 = c(0.1,0.12),
                                         param2 = c(0.15,0.5),
                                         param3 = c(NA,3))

timepoint_expert <- 14
data2 <- data %>% rename(status = censored) %>% mutate(time2 = ifelse(time > 10, 10, time),
                                                  status2 = ifelse(time> 10, 0, status))
example1 <- fit.models.expert(formula=Surv(time2,status2)~1,data=data2,
                              distr=c("wph", "gomp"),
                              method="mle",
                              pool_type = "log pool",
                              opinion_type = "survival",
                              times_expert = timepoint_expert,
                              param_expert = param_expert_example1)

p.mle = make.surv(example1,mod= 2,t = 1:30, nsim=1000) #Plot the Gompertz model
psa.plot(p.mle , name_labs = "PSA", labs = "Gompertz", col ="blue")
```

summary.expertsurv — *Prints a summary table for the distribution the mean survival time for a given model and data*

Description

Calculates the mean survival time as the area under the survival curve - ported from `survHE`

Usage

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

Arguments

`object`	a `expertsurv` object (resulting from the call to `fit.models`)
`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

Value

A list comprising of the following elements:

`mean.surv`	A matrix with the simulated values for the mean survival times
`tab`	A summary table

Author(s)

Gianluca Baio

References

Baio G (2020). "survHE: Survival Analysis for Health Economic Evaluation and Cost-Effectiveness Modeling." *Journal of Statistical Software*, **95**(14), 1–47. doi:10.18637/jss.v095.i14.

See Also

`fit.models.expert`, `make.surv`

Examples

```
require("dplyr")
data2 <- data %>% rename(status = censored) %>% mutate(time2 = ifelse(time > 10, 10, time),
                                                         status2 = ifelse(time> 10, 0, status))
mle = example1 <- fit.models.expert(formula=Surv(time2,status2)~1,data=data2,
                   distr=c("wph", "gomp"),
                   method="mle")
summary(mle)
```

Index