Package 'morse'

July 3, 2014

Type Package

Title MOdelling tools for Reproduction and Survival data in Ecotoxicology

Version 1.0.2

Date 2014-06-24

Author Marie Laure Delignette-Muller [aut, cre], Philippe Ruiz [aut, cre], Sandrine Charles [aut], Wandrille Duchemin [ctb], Christelle Lopes [ctb], Philippe Veber [ctb]

Maintainer Philippe Ruiz <philippe.ruiz@univ-lyon1.fr>

Description A package for ecotoxicologists and regulators dedicated to the mathematical and statistical modelling of bioassay data. The package uses advanced and innovative methods for a valuable quantitative environmental risk assessment.

Depends R (>= 3.0.0)

SystemRequirements jags (>= 3.0.0)

Suggests dclone, ggmcmc, ggplot2, gridExtra, lattice, rjags

Imports graphics, grDevices, plyr

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-07-03 15:51:32

R topics documented:

`morse-package` *MOdelling tools for Reproduction and Survival data in Ecotoxicology*

Description

The package offers tools for ecotoxicologists and regulators based on advanced and innovative methods for a valuable quantitative environmental risk assessment. The package allows the analysis of bioassay reproduction data accounting for mortality all along the bioassay. Such data are commonly used to estimate Effective Concentration (EC_x) values from chronic toxicity tests. The aim is to fit an exposure-reponse curve to reproduction data by Bayesian inference while taking into account mortality among parents without loosing valuable data (Delignette-Muller et al., 2014). Models are characterized by a deterministic log-logistic part associated with a stochastic part. Two different stochastic parts can be chosen: Poisson or Gamma-Poisson. The package uses the `rjags` package (Plummer, 2013), an interface from R to the JAGS library for Bayesian data analysis. Note that the `rjags` package does not include a copy of the JAGS library: you must install it separately. For instructions on downloading JAGS, see the home page at `http://mcmc-jags.sourceforge.net`.

Details

Package:	morse
Type:	Package
Version:	1.0.2
Date:	2014-06-24
License:	GPL (>=2)

Author(s)

Marie Laure Delignette-Muller <marielaure.delignettemuller@vetagro-sup.fr>, Philippe Ruiz <philippe.ruiz@univ-lyon1.fr>, Sandrine Charles <sandrine.charles@univ-lyon1.fr>, Wandrille Duchemin <wandrille.duchemin@insa-lyon.fr>, Christelle Lopes <christelle.lopes@univ-lyon1.fr>, Philippe Veber <philippe.veber@univ-lyon1.fr>

Maintainer: Philippe Ruiz <philippe.ruiz@univ-lyon1.fr>

References

Delignette-Muller, M.L., Lopes, C., Veber, P. and Charles, S. (2014) *Statistical handling of reproduction data for exposure-response modelling*. http://arxiv.org/abs/1310.2733.

Plummer, M. (2013) *JAGS Version 3.4.0 user manual*. http://sourceforge.net/projects/mcmc-jags/files/Manuals/3.x/jags_user_manual.pdf/download.

See Also

rjags, lattice, ggplot

Examples

```
# (1) Load the data
data(cadmium1)

# (2) Check data
repro.check.data(cadmium1)

# (3) Plot raw data
repro.survplot(cadmium1, log.scale = TRUE)
repro.fullsurvplot(cadmium1)
repro.cumulplot(cadmium1)

## Not run:
# (4) Fit the log-logistic model
dat <- repro.data(cadmium1)
out <- repro.fit(dat)

# (5) Check the mcmc convergence
repro.convergence(out)

# (6) Summary of the results
plot(out)
summary(out)
print(out)

## End(Not run)
```

---

cadmium1 *Reproduction and survival datasets for* Daphnia magna *exposed to cadmium during 21 days*

Description

Reproduction and survival datasets of chronic laboratory bioassays with *Daphnia magna* freshwater invertebrate exposed to five concentrations of one metal contaminant (cadmium) during 21 days. Five concentrations were tested, with four replicates per concentration. Each replicate contained 10 organisms. Reproduction and survival were monitored at 10 time points.

Usage

```
data(cadmium1)
```

Format

A data frame with 200 observations of the following five variables:

`replicate` A vector of class `integer` with the replicate code (1 to 4).

`conc` A vector of class `numeric` with the cadmium concentrations in $\mu g.L^{-1}$.

`time` A vector of class `integer` with the time points (in days from the beginning of the experiment $t = 0$).

`Nsurv` A vector of class `integer` with the number of alive individuals at each time point for each concentration and each replicate.

`Nrepro` A vector of class `integer` with the number of offspring at each time point for each concentration and each replicate.

References

Billoir, E., Delhaye, H., Forfait, C., Clément, B., Triffault-Bouchet, G., Charles, S. and Delignette-Muller, M.L. (2012) Comparison of bioassays with different exposure time patterns: The added value of dynamic modelling in predictive ecotoxicology, *Ecotoxicology and Environmental Safety*, 75, 80-86.

Examples

```
# (1) Load the data
data(cadmium1)

# (2) Plot the number of survivors as a function of time for each concentration
repro.fullsurvplot(cadmium1)

# (3) Plot the number of survivors as a function of concentration
# at the end of the bioassay
repro.survplot(cadmium1, log.scale = TRUE)

# (4) Plot the cumulated number of offspring as a function of concentration
repro.cumulplot(cadmium1, log.scale = TRUE)
```

---

cadmium2 *Reproduction and survival datasets for snails exposed to cadmium during 56 days*

Description

Reproduction and survival datasets of chronic laboratory bioassays with snails exposed to six concentrations of one metal contaminant (cadmium) during 56 days. Six concentrations were tested, with six replicates per concentration. Each replicate contained five organisms. Reproduction and survival were monitored at 17 time points.

Usage

```
data(cadmium2)
```

Format

A data frame with 612 observations of the following five variables:

`replicate` A vector of class `factor` with the replicate code (`A` to `F`).

`conc` A vector of class `integer` with the cadmium concentrations in $\mu g.L^{-1}$.

`time` A vector of class `integer` with the time points (in days from the beginning of the experiment $t = 0$).

`Nsurv` A vector of class `integer` with the number of alive individuals at each time point for each concentration and each replicate.

`Nrepro` A vector of class `integer` with the number of offspring at each time point for each concentration and each replicate.

Examples

```
# (1) Load the data
data(cadmium2)

# (2) Plot the number of survivors as a function of time for each concentration
repro.fullsurvplot(cadmium2)

# (3) Plot the number of survivors as a function of concentration
# at the end of the bioassay
repro.survplot(cadmium2, log.scale = TRUE)

# (4) Plot the cumulated number of offspring as a function of concentration
repro.cumulplot(cadmium2, log.scale = TRUE)
```

---

chlordan — *Reproduction and survival datasets for* Daphnia magna *exposed to chlordan during 21 days*

Description

Reproduction and survival datasets of chronic laboratory bioassays with *Daphnia magna* freshwater invertebrate exposed to six concentrations of one organochlorine insecticide during 21 days. Six concentrations were tested, with 10 replicates per concentration. Each replicate contained one organism. Reproduction and survival were monitored at 22 time points.

Usage

```
data(chlordan)
```

Format

A data frame with 1320 observations of the following five variables:

`replicate` A vector of class `integer` with the replicate code (`1` to `10`).

`conc` A vector of class `numeric` with the chlordan concentrations in $\mu g.L^{-1}$.

`time` A vector of class `integer` with the time points (in days from the beginning of the experiment $t = 0$).

`Nsurv` A vector of class `integer` with the number of alive individuals at each time point for each concentration and each replicate.

`Nrepro` A vector of class `integer` with the number of offspring at each time point for each concentration and each replicate.

References

Manar, R., Bessi, H. and Vasseur, P. (2009) Reproductive effects and bioaccumulation of chlordan in Daphnia magna, *Environmental Toxicology and Chemistry*, 28, 2150-2159.

Examples

```
# (1) Load the data
data(chlordan)

# (2) Plot the number of survivors as a function of time for each concentration
repro.fullsurvplot(chlordan)

# (3) Plot the number of survivors as a function of concentration
# at the end of the bioassay
repro.survplot(chlordan, log.scale = TRUE)

# (4) Plot the cumulated number of offspring as a function of concentration
repro.cumulplot(chlordan, log.scale = TRUE)
```

---

copper

Reproduction and survival datasets for Daphnia magna *exposed to copper during 21 days*

Description

Reproduction and survival datasets of chronic laboratory bioassays with *Daphnia magna* freshwater invertebrate exposed to five concentrations of one metal contaminant (copper) during 21 days. Five concentrations were tested, with three replicates per concentration. Each replicate contained 20 organisms. Reproduction and survival were monitored at 16 time points.

Usage

```
data(copper)
```

Format

A data frame with 240 observations of the following five variables:

`replicate` A vector of class `factor` with the replicate code (`A` to `C`).

`conc` A vector of class `numeric` with the copper concentrations in $\mu g.L^{-1}$.

`time` A vector of class `integer` with the time points (in days from the beginning of the experiment $t = 0$).

`Nsurv` A vector of class `integer` with the number of alive individuals at each time point for each concentration and each replicate.

`Nrepro` A vector of class `integer` with the number of offspring at each time point for each concentration and each replicate.

References

Billoir, E., Delignette-Muller, M.L., Péry, A.R.R. and Charles, S. (2008) A Bayesian Approach to Analyzing Ecotoxicological Data, *Environmental Science & Technology*, 42 (23), 8978-8984.

Examples

```
# (1) Load the data
data(copper)

# (2) Plot the number of survivors as a function of time for each concentration
repro.fullsurvplot(copper)

# (3) Plot the number of survivors as a function of concentration
# at the end of the bioassay
repro.survplot(copper, log.scale = TRUE)

# (4) Plot the cumulated number of offspring as a function of concentration
repro.cumulplot(copper, log.scale = TRUE)
```

---

repro.check.data *Check the consistency of the dataset*

Description

The `repro.check.dat` function performs several tests on the integrity of the dataset (column headings, type of data...) and returns a list of error messages if data are not in the correct format. The aim of this function is to check the consistency of the dataframe before using function `repro.data`. This function highlights possible errors in the data structure that would disturb or prevent the execution of the function `repro.data`.

Usage

```
repro.check.data(data, diagnos.plot = TRUE)

## S3 method for class 'repro.check.data'
print(x, ...)
```

Arguments

`data`	Raw dataframe with five columns. See `repro.data` function for details on the required data format.
`diagnos.plot`	If `TRUE`, calls the default `repro.fullsurvplot` function if the number of survivors increases at some time points.
`x`	An object of class repro.check.data.
`...`	Further arguments to be passed to generic methods.

Details

For a given dataframe, the function checks if:

1) column headings are correct: `replicate` for the column of replicates, `conc` for the column of concentrations, `time` for the column of time points, `Nsurv` for the column of the number of alive individuals and `Nrepro` for the column of the number of collected offspring at each time point,

2) the first time point of the dataset is 0,

3) the class of column `conc` is `numeric`,

4) the classes of columns `Nsurv` and `Nrepro` are `integer`,

5) values of the dataframe are all positive,

6) the number of collected offspring is 0 at $t = 0$,

7) each replicate appears only once per concentration and per time point,

8) the number of replicates is the same at any concentration and any time point,

8) the number of alive individuals never increases with time,

9) at each time T, if the number of alive individuals is null, the number of collected offspring is also null at time $T + 1$.

Value

Returns an object of class `repro.check.data`. A dataframe with two columns of character string, `id` and `msg`. The `id` is invisible when displaying the function. Print only shows error messages `msg`.

`id` The identifier of the test, equals to:

- `missingcolumn` if one or more columns are missing or if the column headings are not `replicate`, `conc`, `time`, `Nsurv` and `Nrepro`.
- `firstTime0` if the first time point is not 0 at each concentration and each replicate.
- `concNumeric` if column `conc` does not contain values of class `numeric` only.
- `NsurvInteger` if column `Nsurv` does not contain values of class `integer` only.
- `NreproInteger` if column `Nrepro` does not contain values of class `integer` only.
- `tablePositive` if there are negative values within the data.
- `Nrepro0T0` if `Nrepro` is not 0 at time 0 for each concentration and each replicate.

`onlyReplicate` if a replicate is duplicated on different lines for the same time points and the same concentration.

`missingReplicate` if a replicate is missing for at least one time points at one concentration.

`NsurvMonotone` if `Nsurv` increases at some time points compared to the previous one.

`Nsurvt0Nreprotp1P` if at a giving time T, the number of alive individuals is null and the number of collected offspring is not null for the same replicate and the same concentration at time $T + 1$.

`msg` One or more user friendly error messages are generated:

```
The column 'colname' is missing or have a wrong name.
Data are required at time 0 for each concentration and each
   replicate.
Column 'conc' must contain only numerical values.
Column 'Nsurv' must contain only integer values.
Column 'Nrepro' must contain only integer values.
Data must contain only positive values.
'Nrepro' should be 0 at time 0 for each concentration and each
   replicate.
Replicate 'replicate' appears on different lines for the same time point at concentra
Replicate 'replicate' is missing for at least one time points at concentration 'conc'
For replicate 'replicate' and concentration 'conc', 'Nsurv'
   increases at some time points compared to the previous one.
For replicate 'replicate' and concentration 'conc', there are some
   'Nsurv' = 0 followed by 'Nrepro' > 0 at the next time point.
```

Note

If an error of type `missingcolumn` is detected, the function `repro.check.data` is stopped. When no error is detected the `repro.check.data` function returns an empty dataframe.

Author(s)

Marie Laure Delignette-Muller <marielaure.delignettemuller@vetagro-sup.fr>, Philippe Veber <philippe.veber@univ-lyon1.fr>, Philippe Ruiz <philippe.ruiz@univ-lyon1.fr>

See Also

`repro.fullsurvplot`, `repro.data`

Examples

```
# Run the check data function
data(zinc)
repro.check.data(zinc)

# Example with an error in the dataframe
```

```
# (1) Load the data
data(zinc)

# (2) Insert an error (increase the number of survivors at a certain time point compared to its
# value at the previous time point within the same remplicate)
zinc[25,"Nsurv"] <- 20
check <- repro.check.data(zinc, diagnos.plot = TRUE)

# (3) Check for potential errors in the dataframe
check
```

`repro.convergence` *Convergence check of the MCMC chains*

Description

The `repro.convergence` function checks the convergence of the MCMC chains from the JAGS estimate with the Gelman and Rubin convergence diagnostic (Gelman and Rubin, 1992). It summarizes the `mcmc` or `mcmc.list` object with a trace of the sampled output, a density estimate and an autocorrelation plot for each variable in the chain.

Usage

```
repro.convergence(out, trace = TRUE, density = TRUE,
autocorr = TRUE, type = "generic")
```

Arguments

`out`	An object of class `repro.fit`.
`trace`	If `TRUE`, the function traces the sampled output estimate for each variable in the chain.
`density`	If `TRUE`, the function plots the density estimate for each variable in the chain.
`autocorr`	If `TRUE`, the function plots the autocorrelation for each variable in each chain.
`type`	Graphical method: `generic` or `ggplot`.

Value

A list with the point estimate of the multivariate potential scale reduction factor and the point estimate of the potential scale reduction factor (Rhat) for each parameter of the Gelman and Rubin test (Gelman and Rubin, 1992). A value close to 1 is expected when convergence is reached. See the `gelman.diag` help for more details.

Note

When `type = "ggplot"`, the function calls packages `ggmcmc` and `gridExtra` and returns an object of class `ggplot`.

Author(s)

Marie Laure Delignette-Muller <marielaure.delignettemuller@vetagro-sup.fr>, Philippe Ruiz <philippe.ruiz@univ-lyon1.fr>

References

Gelman, A. and Rubin, D.B. (1992) *Inference from iterative simulation using multiple sequences*, Statistical Science, 7, 457-511.

See Also

`repro.fit` and `gelman.diag`, `plot.mcmc`, `autocorr.plot` from the `rjags` package and `ggs_traceplot`, `ggs_density` and `ggs_autocorrelation` from the ggmcmc package (http://xavier-fim.net/packages/ggmcmc)

Examples

```
# (1) Load the data
data(zinc)

# (2) Create an object of class "repro.data"
dat <- repro.data(zinc)

## Not run:
# (3) Run the repro.fit function
out <- repro.fit(dat)

# (4) Check the convergence
repro.convergence(out, trace = TRUE, density = FALSE,
autocorr = TRUE)

# (5) Check the convergence using the "ggmcmc" package
repro.convergence(out, type = "ggplot")

## End(Not run)
```

`repro.cumulplot` *Cumulative plot of reproduction data*

Description

The `repro.cumulplot` function plots the cumulative number of offspring (e.g. eggs, clutches...) at the end of the assay for a given concentration of contaminant. Replicates with and without mortality are plotted with different symbols.

Usage

```
repro.cumulplot(data, xlab, ylab, type = "generic",
log.scale = FALSE, addlegend = TRUE, ...)
```

Arguments

data	Raw dataframe with five columns: replicate, conc, time, Nsurv, Nrepro. See repro.data for details.
xlab	A label for the X-axis, by default Concentrations.
ylab	A label for the Y-axis, by default Nreprocumul.
type	Graphical method: generic or ggplot.
log.scale	If TRUE, a log-scale is used on the X-axis.
addlegend	If TRUE, a default legend is added to the plot differentiating replicates with and without mortality.
...	Furthers arguments passed to the generic plot function.

Note

When type = "ggplot", the function calls package ggplot2 and returns an object of class ggplot.

Author(s)

Marie Laure Delignette-Muller <marielaure.delignettemuller@vetagro-sup.fr>, Philippe Ruiz <philippe.ruiz@univ-lyon1.fr>

See Also

ggplot

Examples

```
# (1) Load the data
data(chlordan)

# (2) Plot the cummulative reproduction data
repro.cumulplot(chlordan, log.scale = TRUE)

# (3) Personalize legend
repro.cumulplot(chlordan, addlegend = FALSE)
legend("left",title = "Legend", pch = c(19,1), bty = "n",
legend = c("Without mortality","With mortality"))

# (4) Personalize legend with ggplot type
cum <- repro.cumulplot(chlordan, type = "ggplot", addlegend = FALSE)
cum + theme(legend.position = "left") + scale_colour_hue("Legend",
breaks = c("1","19"), labels = c("With mortality","Without mortality"))
```

repro.data *Transformed dataset for the* repro.fit *function*

Description

The repro.data function creates a repro.data object needed to run the repro.fit function. A new dataframe called transformed.data is created by adding three new columns at target time for each replicate and each concentration: the initial number of individuals (Ninit), the cumulative number of offspring (Nreprocumul) and the number of individual-days (Nindtime). The generic methods are print and summary. The use of the repro.check.data function is recommended before using the repro.data function to detect possible structural errors in raw data.

Usage

```
repro.data(data, target.time)
## S3 method for class 'repro.data'
summary(object, ...)
## S3 method for class 'repro.data'
print(x, ...)
```

Arguments

data	The raw dataframe with five columns passed to the function in argument data: **replicate** A vector of class integer or factor for replicate identification. **conc** A vector of class numeric with tested concentrations (positive values). **time** A vector of class integer with time points (positive values). The first time must be 0. **Nsurv** A vector of class integer with positive values of the number of alive individuals (positive values) at each time point for each concentration and each replicate. **Nrepro** A vector of class integer with the number of offspring (positive values) at each time point for each concentration and each replicate.
target.time	The time at which the number of individual-days and the cumulative number of offspring from the beginning of the bioassay are calculated. By default the last time point.
object	An object of class repro.data.
x	An object of class repro.data.
...	Further arguments to be passed to generic methods.

Details

The raw dataframe must have exactly five columns arranged in the correct order with the following headings: replicate, conc, time, Nsurv and Nrepro. It is recommended to use the function repro.check.data in order to check if the dataframe is in the correct format. The repro.data function builds a new dataframe called transformed.data, that is a subset of the raw dataframe

at target.time with three additional columns: Ninit, Nreprocumul, Nindtime. The number of individual-days is needed to account for the time-contribution of each individual to the cummulative reproduction (Delignette-Muller et al., 2014). See the section **Value**. The new dataframe is automatically reordered by time, concentration and replicate.

Value

Returns an object of class repro.data. A list of three objects:

raw.data The raw dataframe with five columns corresponding to the argument passed in the function.

transformed.data
A dataframe with six columns:

replicate A vector of class integer or factor for replicate identification.

conc A vector of class numeric with tested concentrations (positive values).

Ninit A vector of class integer with the number of individuals at the beginning of the bioassay (positive values).

Nsurv A vector of class integer with positive values of the number of alive individuals (positive values) at the target time for each concentration and each replicate.

Nreprocumul A vector of class integer with the cumulative number of offspring (positive values) at the target time for each concentration and each replicate.

Nindtime A vector of class numeric with the number of individual-days (positive values) at target time for each concentration and each replicate.

target.time The time at which the number of individual-days and the cumulative number of offspring from the beginning of the bioassay are calculated.

Generic functions:

summary The summary provides information about the structure of the dataset and the experimental design: the number of datapoints per replicate, concentration and time both for the raw dataset and the transformed dataset.

print Print of a repro.data object with the transformed dataframe and the value of the target time.

Author(s)

Marie Laure Delignette-Muller <marielaure.delignettemuller@vetagro-sup.fr>, Philippe Ruiz <philippe.ruiz@univ-lyon1.fr>

References

Delignette-Muller M.L., Lopes C., Veber P. and Charles S. (2014) *Statistical handling of reproduction data for exposure-response modelling*. http://arxiv.org/abs/1310.2733.

See Also

repro.check.data, repro.fit

Examples

```
# (1) Load the data
data(zinc)

# (2) Create an objet of class 'repro.data'
dat <- repro.data(zinc)

# (3) Print and summarize object dat
print(dat)
summary(dat)
```

`repro.fit` *Fit an exposure-response model to reproduction data taking mortality into account within the Bayesian framework*

Description

The `repro.fit` function estimates the parameters of an exposure-response model using Bayesian inference. One deterministic part is proposed: the log-logistic function. Two stochastic parts may be chosen by the function in order to take into account the nature of reproduction data and the inter-replicate variability (Delignette-Muller et al., 2014). The function calls the `rjags` package (Plummer, 2013) and the `dclone` package for parallelization of chains. The function returns parameter estimates of the exposure-response model and estimates of x % effective concentration for x = 5, 10, 20 and 50. The `repro.parfit` does the same thing as `repro.fit`, but chains are run on parallel workers, so that computations can be faster for long MCMC runs. Generic methods are `print`, `plot` and `summary`.

Usage

```
repro.fit(rdata, n.chains = 3, quiet = FALSE)

repro.parfit(rdata, n.chains = 3, quiet = FALSE)

## S3 method for class 'repro.fit'
summary(object, ...)

## S3 method for class 'repro.fit'
print(x, ...)

## S3 method for class 'repro.fit'
plot(x, xlab, ylab, fitcol, fitlty, fitlwd,
ci = FALSE, cicol, cilty, cilwd, addlegend = TRUE,
log.scale = FALSE, type = "generic", ...)
```

Arguments

rdata	An object of class repro.data.
n.chains	Number of MCMC chains. The minimum required number of chains is 2.
quiet	If TRUE, make silent all prints and progress bars of JAGS compilation.
object	An object of class repro.fit.
x	An object of class repro.fit.
xlab	A label for the X-axis, by default Concentrations.
ylab	A label for the Y-axis, by default Response.
fitcol	A single color to plot the fitted curve, by default red.
fitlty	A single line type to plot the fitted curve, by default 1.
fitlwd	A single numeric which controls the width of the fitted curve, by default 1.
ci	If TRUE, the 95 % credible limits of the model are plotted.
cicol	A single color to plot the 95 % credible limits, by default red.
cilty	A single line type to plot 95 % credible limits, by default 1.
cilwd	A single numeric which controls the width of the 95 % credible limits, by default 2.
addlegend	If TRUE, a default legend is added to the plot.
log.scale	If TRUE, a log-scale is used on the X-axis.
type	Graphical method: generic or ggplot.
...	Further arguments to be passed to generic methods.

Details

"log-logistic" deterministic part: the reproduction rate (expressed in number of offspring per individual-day) at concentration C_i is described by:

$$f(C_i) = \frac{d}{1 + (\frac{C_i}{e})^b}$$

where d stands for the expected number of offspring per individual-day in the control, e is the 50 % effective concentration (EC_{50}) and b is a slope parameter. The number of offspring at concentration i and replicate j is described by a Poisson distribution of mean equal to the product of the number of individual-days $Nindtime_{ij}$ by a term f_{ij} differing between the two stochastic parts.

$$N_{ij} \sim Poisson(f_{ij} \times Nindtime_{ij})$$

with $Nindtime_{ij}$ the number of individual-days at the target time for each replicate and each concentration (Delignette-Muller et al., 2014).

"Poisson" stochastic part : f_{ij} only depends on the concentration:

$$f_{ij} = f(C_i)$$

"Gamma-Poisson" stochastic part: f_{ij} is assummed to be variable between replicates at a same concentration and to follow a gamma distribution:

$$f_{ij} \sim Gamma(\frac{f(C_i)}{\omega}, \frac{1}{\omega})$$

with ω the overdispersion parameter.

DIC: The Deviance Information Criterium (DIC) as defined by Spiegelhalter et al. (2002) is provided by the `dic.samples` function.

Raftery and Lewis's diagnostic: The `raftery.diag` is a run length control diagnostic based on a criterion that calculates the appropriate number of iterations required to accurately estimate the parameter quantiles. The Raftery and Lewis's diagnostic value used in the `repro.fit` function is the `resmatrix` object. See the `raftery.diag` help for more details.

Model selection: The `repro.fit` function chooses itself between the Poisson and the Gamma-Poisson model depending on the number of MCMC samples and on the DIC values. The minimum number of MCMC samples for the pilot run is provided by the Raftery and Lewis's diagnostic (Raftery and Lewis 1992). If this number is more than 100 000 or if the DIC difference between models is small (typically less than 1), then the Poisson model is selected.

Value

Returns an obbject of class `repro.fit`. A list of 13 objects:

`DIC`	DIC value of the selected model.
`estim.ECx`	A table of the estimated 5, 10, 20 and 50 % effective concentrations and their 95 % credible intervals.
`estim.par`	A table of the estimated parameters as medians and 95 % credible intervals.
`mcmc`	An object of class `mcmc.list` with the posterior distributions.
`model`	A JAGS model object.
`model.label`	An undocumented value for internaluse only.
`n.chains`	An integer value corresponding to the number of chains used for the MCMC computation.
`n.burnin`	A numerical value corresponding to the number of discarded draws for the burn-in period.
`n.iter`	A numerical value corresponding to the number of monitored iterations.
`param.prior`	An undocumented list for internaluse only.
`n.thin`	A numerical value corresponding to the thinning interval.
`raw.data`	The raw dataframe with five columns passed to the argument of `repro.data`.
`transformed.data`	A dataframe with six columns. See `repro.data` for details.

Generic functions:

`summary` provides the following information: the type of model used, a summary of the MCMC chains with summary statistics for each variable: mean, standard deviation, naive standard error of the mean and time-series standard error based on an estimate of the spectral density

at 0, quantiles of the sample distribution using the `quantiles` argument. See `summary.mcmc`, prior quantiles, median and 2.5 % and 97.5 % quantiles of posterior distributions of estimated parameters and ECx estimates (x = 5, 10, 20, 50) as 50 %, 2.5 % and 97.5 % quantiles.

`print` shows information about the estimation method: the full JAGS model, the number of chains, the total number of iterations, the number of iterations in the burn-in period, the thin value and the DIC.

`plot` shows the fitted exposure-response curve superimposed to experimental data at target time. The response is here expressed as the cumulative number of offspring per individual-day. See `repro.data`. Two types of output are available: `generic` or `ggplot`.

Note

When the `repro.parfit` function is used, the number of clusters is automatically defined by the function. It is equal to argument `n.chains`.

Author(s)

Marie Laure Delignette-Muller <marielaure.delignettemuller@vetagro-sup.fr>, Philippe Ruiz <philippe.ruiz@univ-lyon1.fr>

References

Delignette-Muller, M.L., Lopes, C., Veber, P. and Charles, S. (2014) Statistical handling of reproduction data for exposure-response modelling. http://arxiv.org/abs/1310.2733.

Plummer, M. (2013) JAGS Version 3.4.0 user manual. http://sourceforge.net/projects/mcmc-jags/files/Manuals/3.x/jags_user_manual.pdf/download

Raftery A.E. and Lewis, S.M. (1992) One long run with diagnostics: Implementation strategies for Markov chain Monte Carlo. *Statistical Science*, 7, 493-497.

Spiegelhalter, D., N. Best, B. Carlin, and A. van der Linde (2002) Bayesian measures of model complexity and fit (with discussion). *Journal of the Royal Statistical Society*, Series B 64, 583-639.

See Also

`rjags`, `coda.samples`, `dic.samples`, `summary.mcmc`, `parJagsModel`, `parCodaSamples`, `repro.data` and `raftery.diag`

Examples

```
# (1) Load the data
data(cadmium1)

# (2) Create the transformed dataset
dat <- repro.data(cadmium1)
class(dat)

## Not run:
# (3) Run the fit
out <- repro.fit(dat, quiet = TRUE)
```

```
# (4) Summary
summary(out)

# (5) Plot the fitted curve
plot(out, log.scale = TRUE, c.i. = TRUE)

# (6) Add a specific legend with generic type
plot(out, addlegend = FALSE)
legend("left", legend = c("Without mortality", "With mortality"),
pch = c(19,1))

## End(Not run)
```

`repro.fullsurvplot` *Plot of survival data*

Description

The `repro.fullsurvplot` function plots the number of survivors as a function of time for each concentration and each replicate. This function is also used by `repro.check.data`.

Usage

```
repro.fullsurvplot(data, xlab, ylab, type = "generic", addlegend = TRUE)
```

Arguments

`data`	Raw dataframe with five columns: `replicate`, `conc`, `time`, `Nsurv`, `Nrepro`. See `repro.data` for details.
`xlab`	A label for the X-axis, by default `Time`.
`ylab`	A label for the X-axis, by default `Number of survivors`.
`type`	Graphical method: `generic`, `lattice` or `ggplot`.
`addlegend`	If `TRUE`, a legend is added to the plot.

Note

When `type = "ggplot"`, the function calls package `ggplot2` and returns an object of class `ggplot`. When `type = "lattice"`, the function returns an object of class `trellis`.

Author(s)

Marie Laure Delignette-Muller <marielaure.delignettemuller@vetagro-sup.fr>, Philippe Ruiz <philippe.ruiz@univ-lyon1.fr>

See Also

`ggplot`, `xyplot`, `repro.check.data`

Examples

```
# (1) Load the data
data(zinc)

# (2) Plot the survival data
repro.fullsurvplot(zinc, type = "generic", addlegend = TRUE)

# (3) Plot the survival data with a lattice type
repro.fullsurvplot(zinc, type = "lattice", addlegend = TRUE)

# (4) Plot the survival data with a ggplot type
repro.fullsurvplot(zinc, type = "ggplot", addlegend = FALSE)

# (5) To build a specific legend with a ggplot type
fu <- repro.fullsurvplot(zinc, type = "ggplot", addlegend = FALSE)
fu + theme(legend.position = "left") + scale_colour_hue("Replicate")
```

`repro.survplot` *Plot of survival data*

Description

The `repro.survplot` plots the number of initially present animals still alive at the end of the assay for a given concentration of contaminant. There is a single point by concentration and replicate. The size of points is proportional to the number of overploted replicates.

Usage

```
repro.survplot(data, xlab, ylab, pch, type = "generic",
log.scale = FALSE, addlegend = TRUE, ...)
```

Arguments

`data`	Raw dataframe with five columns: `replicate`, `conc`, `time`, `Nsurv`, `Nrepro`. See `repro.data` for details.
`xlab`	A label for the X-axis, by default `Concentrations`.
`ylab`	A label for the Y-axis, by default `Number of survivors`.
`pch`	Argument to choose the symbol of replicates.
`type`	Graphical method: `generic` or `ggplot`.
`log.scale`	If `TRUE`, a log-scale is used on the X-axis.
`addlegend`	If `TRUE`, a default legend is added to the plot.
`...`	Further arguments pass to the generic plot function.

Note

The argument pch is only used with `type = "generic"`. When `type = "ggplot"`, the function calls package ggplot2 and returns an object of class ggplot.

Author(s)

Marie Laure Delignette-Muller <marielaure.delignettemuller@vetagro-sup.fr>, Philippe Ruiz <philippe.ruiz@univ-lyon1.fr>

See Also

ggplot

Examples

```
# (1) Load the data
data(zinc)

# (2) plot the number of survivors depending on the concentration
repro.survplot(zinc, log.scale = TRUE)

# (3) To build a specific legend
repro.survplot(zinc, type = "generic", addlegend = FALSE)
legend("left", legend = "Replicate", pch = 4)

# (4) To build a specific legend with ggplot type
surv <- repro.survplot(zinc, type = "ggplot", addlegend = FALSE)
surv + theme(legend.position = "left")
```

---

zinc — *Reproduction and survival datasets for* Daphnia magna *exposed to zinc during 21 days*

Description

Reproduction and survival datasets of chronic laboratory bioassays with *Daphnia magna* freshwater invertebrate exposed to four concentrations of one metal contaminant (zinc) during 21 days. Four concentrations were tested with three replicates per concentration. Each replicate contained 20 organisms. Reproduction and survival were monitored at 15 time points.

Usage

```
data(zinc)
```

Format

A data frame with 180 observations on the following five variables:

`replicate` A vector of class `factor` with the replicate code (`A` to `C`).

`conc` A vector of class `numeric` with zinc concentrations in $mg.L^{-1}$.

`time` A vector of class `integer` with the time points (in days from the beginning of the experiment $t = 0$).

`Nsurv` A vector of class `integer` with the number of alive individuals at each time point for each concentration and each replicate.

`Nrepro` A vector of class `integer` with the number of offspring at each time point for each concentration and each replicate.

References

Billoir, E.,Delignette-Muller, M.L., Péry, A.R.R. and Charles S. (2008) A Bayesian Approach to Analyzing Ecotoxicological Data, *Environmental Science & Technology*, 42 (23), 8978-8984.

Examples

```
# (1) Load the data
data(zinc)

# (2) Plot the number of survivors as a function of time for each concentration
repro.fullsurvplot(zinc)

# (3) Plot the number of survivors as a function of concentration
# at the end of the bioassay
repro.survplot(zinc, log.scale = TRUE)

# (4) Plot the cumulated number of offspring as a function of concentration
repro.cumulplot(zinc, log.scale = TRUE)
```

Index