

Package ‘phenology’

August 24, 2026

Type Package

Title Tools to Manage a Parametric Function that Describes Phenology and More

Version 2026.8.24

Date 2026-08-24

Depends numDeriv, parallel, optimx, HelpersMG (>= 6.6), R (>= 4.1)

Suggests shiny, fields, progress, car, MultiRNG, nlWaldTest, pbapply, cranlogs

Description Functions used to fit and test the phenology of species based on counts. Based on Girondot, M. (2010) <[doi:10.3354/esr00292](https://doi.org/10.3354/esr00292)> for the phenology function, Girondot, M. (2017) <[doi:10.1016/j.ecolind.2017.05.063](https://doi.org/10.1016/j.ecolind.2017.05.063)> for the convolution of negative binomial, Girondot, M. and Rizzo, A. (2015) <[doi:10.2993/etbi-35-02-337-353.1](https://doi.org/10.2993/etbi-35-02-337-353.1)> for Bayesian estimate, Pfaller JB, ..., Girondot M (2019) <[doi:10.1007/s00227-019-3545-x](https://doi.org/10.1007/s00227-019-3545-x)> for tag-loss estimate, Hancock J, ..., Girondot M (2019) <[doi:10.1016/j.ecolmodel.2019.04.013](https://doi.org/10.1016/j.ecolmodel.2019.04.013)> for nesting history, Laloe J-O, ..., Girondot M, Hays GC (2020) <[doi:10.1007/s00227-020-03686-x](https://doi.org/10.1007/s00227-020-03686-x)> for aggregating several seasons.

License GPL-2

LazyData yes

LazyLoad yes

Encoding UTF-8

NeedsCompilation no

Config/roxygen2/version 8.1.0

Author Marc Girondot [aut, cre] (ORCID:
<<https://orcid.org/0000-0001-6645-8530>>)

Maintainer Marc Girondot <marc.girondot@gmail.com>

Repository CRAN

Date/Publication 2026-08-24 13:40:02 UTC

Contents

phenology-package	4
adapt_parameters	6
add_phenology	7
add_SE	11
AutoFitPhenology	13
Bayesian.remigration	14
BE_to_LBLE	16
CI.RMU	17
ECFOCF_f	22
ECFOCF_full	23
ExponentialRegression	25
extract_result	28
fitCF	29
fitCF_MHmcmc	35
fitCF_MHmcmc_p	38
fitRMU	39
fitRMU_MHmcmc	44
fitRMU_MHmcmc_p	46
fit_phenology	47
fixed.parameters0	54
generateCF	55
Gratiot	57
IPFit	58
IPModel	65
IPPredict	67
LBLE_to_BE	68
LBLE_to_L	69
likelihood_phenology	70
lnLCF	72
LnRI_norm	73
logLik.ECFOCF	75
logLik.fitRMU	76
logLik.phenology	77
logLik.Tagloss	78
L_to_LBLE	79
map_Gratiot	80
map_phenology	81
MarineTurtles_2002	84
MinBMinE_to_Min	85
outLR	86
o_4p_p1p2	87
Parameter_Global_Year	88
par_init	89
phenology	91
phenology2fitRMU	92
phenology_MHmcmc	94

phenology_MHmcmc_p	97
plot.ECFOCF	98
plot.fitRMU	101
plot.IP	104
plot.phenology	107
plot.phenologydata	110
plot.phenologymap	112
plot.Remigration	113
plot.TableECFOCF	115
plot.Tagloss	117
plot.TaglossData	121
plot_delta	122
plot_phi	123
print.phenology	125
print.phenologymap	126
print.phenologyout	128
remove_site	129
result_Gratiot	130
result_Gratiot1	131
result_Gratiot2	132
result_Gratiot_Flat	133
RI	134
RI2BP	136
Sea_Turtles_Bayesian	138
shift_sinusoid	152
summary.IP	154
summary.phenology	155
summary.phenologymap	157
summary.phenologyout	158
TableECFOCF	160
Tagloss_cumul	161
Tagloss_daymax	165
Tagloss_fit	166
Tagloss_format	169
Tagloss_L	171
Tagloss_LengthObs	176
Tagloss_mcmc	177
Tagloss_mcmc_p	179
Tagloss_model	180
Tagloss_simulate	183
toggle_Min_PMin	184

phenology-package	<i>Tools to Manage a Parametric Function that Describes Phenology and More</i>
-------------------	--

Description

Functions used to fit and test the phenology of species based on counts.

Note that only the most significant changes are reported in the NEWS.

The latest version of this package can always been installed using:

```
install.packages("http://marc.girondot.free.fr/CRAN/HelpersMG.tar.gz", repos=NULL, type="source")
```

```
install.packages("http://marc.girondot.free.fr/CRAN/phenology.tar.gz", repos=NULL, type="source")
```



Details

Fit a parametric function that describes phenology

Package: phenology

Type: Package
 Version: 2026.8.24 build 1737
 Date: 2026-08-24
 License: GPL (>= 2)
 LazyLoad: yes

Author(s)

Marc Girondot <marc.girondot@gmail.com>

References

- Girondot, M. 2010. Estimating density of animals during migratory waves: application to marine turtles at nesting site. *Endangered Species Research*, 12, 85-105.
- Girondot M. and Rizzo A. 2015. Bayesian framework to integrate traditional ecological knowledge into ecological modeling: A case study. *Journal of Ethnobiology*, 35, 339-355. doi:10.2993/etbi-35-02-337-353.1
- Girondot, M. 2010. Editorial: The zero counts. *Marine Turtle Newsletter*, 129, 5-6.
- Girondot, M., 2017. Optimizing sampling design to infer marine turtles seasonal nest number for low-and high-density nesting beach using convolution of negative binomial distribution. *Ecological Indicators* 81, 83–89.
- Rivalan, P., Godfrey, M.H., Prévot-Julliard, A.-C., Girondot, M., 2005. Maximum likelihood estimates of tag loss in leatherback sea turtles. *Journal of Wildlife Management* 69, 540-548.

See Also

- Girondot, M., Rivalan, P., Wongsopawiro, R., Briane, J.-P., Hulin, V., Caut, S., Guirlet, E. & Godfrey, M. H. 2006. Phenology of marine turtle nesting revealed by a statistical model of the nesting season. *BMC Ecology*, 6, 11.
- Delcroix, E., Bédél, S., Santelli, G., Girondot, M., 2013. Monitoring design for quantification of marine turtle nesting with limited human effort: a test case in the Guadeloupe Archipelago. *Oryx* 48, 95-105.
- Briane J-P, Rivalan P, Girondot M (2007) The inverse problem applied to the Observed Clutch Frequency of Leatherbacks from Yalimapo beach, French Guiana. *Chelonian Conservation and Biology* 6:63-69
- Fossette S, Kelle L, Girondot M, Goverse E, Hilterman ML, Verhage B, Thoisy B, de, Georges J-Y (2008) The world's largest leatherback rookeries: A review of conservation-oriented research in French Guiana/Suriname and Gabon. *Journal of Experimental Marine Biology and Ecology* 356:69-82

Examples

```
## Not run:
library(phenology)
# Read a file with data
data(Gratitot)
```

```

# Generate a formatted list named data_Gratiot
data_Gratiot <- add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg <- par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
result_Gratiot <- fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=NULL)
data(result_Gratiot)
# Plot the phenology and get some stats
output <- plot(result_Gratiot)

# How many times this package has been download
library(cranlogs)
phenology <- cran_downloads("phenology", from = "2012-10-06",
  to = Sys.Date() - 1)

sum(phenology$count)
plot(phenology$date, phenology$count, type="l", bty="n")

## End(Not run)

```

adapt_parameters	<i>Extract the parameters from a set of parameters to be used with another dataset.</i>
------------------	---

Description

The function "adapt_parameters" extracts the set of parameters to be used with a subset of data. All the unnecessary parameters are removed. It can be used when a set of beaches are fitted first and after only one of these beaches is fitted again.

Usage

```

adapt_parameters(
  data = stop("Datasets is mandatory for this function"),
  parameters = stop("Set of parameters is mandatory for this function")
)

```

Arguments

data	A dataset of counts
parameters	A set of parameters

Details

adapt_parameters get the fitted parameters from a result object.

Value

Return the set of parameters

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Phenology model: `AutoFitPhenology()`, `BE_to_LBLE()`, `Gratiot`, `LBLE_to_BE()`, `LBLE_to_L()`, `L_to_LBLE()`, `MarineTurtles_2002`, `MinBMinE_to_Min()`, `add_SE()`, `add_phenology()`, `extract_result()`, `fit_phenology()`, `likelihood_phenology()`, `logLik.phenology()`, `map_Gratiot`, `map_phenology()`, `par_init()`, `phenology()`, `phenology2fitRMU()`, `phenology_MHmcmc()`, `phenology_MHmcmc_p()`, `plot.phenology()`, `plot.phenologydata()`, `plot.phenologymap()`, `plot_delta()`, `plot_phi()`, `print.phenology()`, `print.phenologymap()`, `print.phenologyout()`, `remove_site()`, `result_Gratiot`, `result_Gratiot1`, `result_Gratiot2`, `result_Gratiot_Flat`, `summary.phenology()`, `summary.phenologymap()`, `summary.phenologyout()`

Examples

```
library(phenology)
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
refdate <- as.Date("2001-01-01")
data_Gratiot<-add_phenology(Gratiot, name="Complete",
  reference=refdate, format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Add unnecessary parameters to parg
parg <- c(parg, Max_dummybeach=2, Peak_dummybeach=123)
# Extract the fitted parameters
parg1<-adapt_parameters(data=data_Gratiot, parameters=parg)
```

add_phenology

Create a new dataset or add a timeserie to a previous dataset.

Description

To create a new dataset, the syntaxe is :

```
data <- add_phenology(add=newdata, name="Site", reference=as.Date('2001-12-31'), format='\
```

To add a dataset to a previous one, the syntax is :

```
data <- add_phenology(previous=previousdata, add=newdata, name='Site',
  reference=as.Date('2001-01-01'), format="\ The dataset to be added must include 2 or 3 columns.
```

The colname.Date included the dates in the format specified by the parameter format. If the number of nests is known for an exact date, then only one date must be indicated.

If the number of nests is known for a range of date, the first and last dates must be separated by a sep.dates character.

For example: 1/2/2000-10/2/2000

Note that date in the colname.Date column can be already formatted and in this case the parameter format is ignored.

The colname.Number includes the number of nests observed for this date or this range of dates.
The colname.Rookery is optional and includes the name of the rookeries.

If only two columns are indicated, the name can be indicated as a parameter of the function with the parameter name. If no name is indicated, the default name Site will be used, but take care, only one rookery of this name can be used.

Several rookeries can be included in the same file but in this case the rookery name is obligatory at the colname.Rookery column.

The model cannot be fitted if a timeseries has no observation because the trivial solution is of course with max=0. The solution is to include a fake false observation at the closest position of the peak, and then the estimated number of nests/tracks will be the estimated number - 1.

If include0 is TRUE, then the series with no observation are included and one observation is added at the monitored date the closest of datepeakfor0.

The normal way to manage such a situation is as followed:

- 1- Format data with include0 being FALSE
- 2- Fit parameters using `fdf <- fit_phenology()`
- 3- Format data with include0 being TRUE and `datepeakfor0=fdf$par["Peak"]`
- 4- Fix previously fitted parameters using `pfixed <- fdf$par`
- 5- Generate new set of parameters with `par_init(data, fixed.parameters=pfixed)`
- 6- Run again `fit_phenology()`

Some problems that can occur:

If a name is defined as a third column of a data.frame and a name is defined also with name parameter, the third column has priority.

Two different timeseries MUST have different name and characters _ and space are forbidden in timeseries names. They are automatically changed if they are present.

Usage

```
add_phenology(
  add = stop("New data must be given."),
  name = "Site",
  reference = NULL,
  month_ref = NULL,
  sep.dates = "-",
  end.season.date = NULL,
  colname.Date = 1,
  colname.Number = 2,
  colname.Rookery = "Site",
  colname.CountTypes = NULL,
  CountTypes.default = "exact",
  colname.A = NULL,
  A.default = NA,
  colname.S = NULL,
  S.default = NA,
  colname.ZeroCounts = NULL,
  ZeroCounts.default = TRUE,
```

```

    format = "%d/%m/%Y",
    previous = NULL,
    include0 = FALSE,
    datepeakfor0 = NULL,
    expandRange0observation = TRUE,
    check.overlapping.dates = TRUE,
    silent = FALSE
)

```

Arguments

add	The data to be added. It can be a set of several entities that uses the same reference and date format.
name	The name of the monitored site.
reference	as.Date('2001-12-31') The date used as day 0 in the ordinal date model.
month_ref	If no reference date is given, use this month as a reference.
sep.dates	Separator used to separate dates when incertitude is included.
end.season.date	The date corresponding to the end of nesting season.
colname.Date	Name or number of column with dates.
colname.Number	Name or number of column with numbers.
colname.Rookery	Name or number of column with rookery names.
colname.CountTypes	Model of count type. It can be "exact" (default), "minimum" or a number to indicate the maximum possible.
CountTypes.default	The default of CountTypes if colname.CountTypes is not provided.
colname.A	The A parameter of the detection model
A.default	Default A value.
colname.S	The S parameter of the detection model
S.default	Default S value.
colname.ZeroCounts	The name of the column to indicate whether zero counts are included (TRUE is default).
ZeroCounts.default	The default for ZeroCounts.
format	The format of the dates.
previous	Previous data formatted with add_phenology or NULL [default] if no previous data exist.
include0	Does timeseries with only 0 should be included?
datepeakfor0	If series with no observation are included, where add a 1 value in ordinal date (see description)

expandRange0Observation
 If TRUE, the range of date with 0 observations are expanded into individual dates
 check.overlapping.dates
 If TRUE, will check for date overlapping
 silent
 Does information about added timeseries is shown

Details

add_phenology creates a new dataset.

Value

Return a list of formatted data that can be used with fit_phenology()

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
## Not run:
library(phenology)
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
refdate <- as.Date("2001-01-01")
data_Gratiot <- add_phenology(Gratiot, name="Complete",
  reference=refdate, format="%d/%m/%Y")

# Generate initial points for the optimisation
parg <- par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
result_Gratiot <- fit_phenology(data=data_Gratiot, fitted.parameters=parg,
  fixed.parameters=NULL)
data(result_Gratiot)
# Plot the phenology and get some stats
output <- plot(result_Gratiot)

#####
```

```

# Example of use of include0 and datepeakfor0
#####
# Let create a times series with only 0
data0 <- data.frame(Date=c("11/3/2015", "12/3/2015", "13/3/2015-18/3/2015", "25/3/2015"),
                    Number=c(0, 0, 0, 0),
                    Beach=rep("Site", 4), stringsAsFactors=FALSE)
# Here I don't include beach with no observation: error message
try1 <- add_phenology(data0, format="%d/%m/%Y", month_ref=1, include0=FALSE)
# Here I include timeseries with no observation
try1 <- add_phenology(data0, format="%d/%m/%Y", month_ref=1, include0=TRUE, datepeakfor0=100)
try1 <- add_phenology(data0, format="%d/%m/%Y", month_ref=1, include0=TRUE, datepeakfor0=73)
try1 <- add_phenology(data0, format="%d/%m/%Y", month_ref=1, include0=TRUE, datepeakfor0=70)
# It can be done in two steps
try1 <- add_phenology(data0, format="%d/%m/%Y", month_ref=1, include0=TRUE)
try2 <- add_phenology(previous=try1, include0=TRUE, datepeakfor0=100)
# Here I include the series without observation
try1 <- add_phenology(add=data0, format="%d/%m/%Y", month_ref=1,
                    include0=TRUE, expandRange0Observation=TRUE)

#####
# Example of A and S parameters to say that only half of a beach was monitored
#####

refdate <- as.Date("2001-01-01")
data_Gratiot <- add_phenology(Gratiot, name="Complete1",
                             reference=refdate, format="%d/%m/%Y")

S.default = 10
# Let Complete1 be of length 2 and Complete0.5 be of length 1
length.Complete1 <- 2
length.Complete0.5 <- 1
A.default = log(1/(length.Complete0.5/length.Complete1)-1)/(-S.default*4)
# For day 0, the detection probability is
1/(1+exp(-(4*S.default)*(A.default-0)))
data_Gratiot <- add_phenology(previous=data_Gratiot,
                             add=Gratiot,
                             name="Complete0.5",
                             A.default = A.default,
                             S.default = S.default,
                             reference=refdate,
                             format="%d/%m/%Y")
parg <- c(par_init(data_Gratiot, fixed.parameters=c(Min=0, Flat=0)), PMin=0.1)
result_Gratiot <- fit_phenology(data=data_Gratiot, fitted.parameters=parg,
                               fixed.parameters=c(Flat=0))

result_Gratiot$par
# it shows that Max_Complete1 is half of Max_Complete0.5; all is ok
summary(result_Gratiot)$synthesis

## End(Not run)

```

Description

This function is used to add standard error for a fixed parameter.

Usage

```
add_SE(fixed.parameters = NULL, parameters = NULL, SE = NULL)
```

Arguments

fixed.parameters	Set of fixed parameters
parameters	Set of current parameters
SE	Standard error value to be added

Details

add_SE adds standard error for a fixed parameter.

Value

The parameters set with the new SE value

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
library(phenology)
# Generate a set of fixed parameter: Flat and Min
pfixed<-c(Flat=0, Min=0)
# Add SE for the Flat parameter
pfixed<-add_SE(fixed.parameters=pfixed, parameters="Flat", SE=5)
```

Description

This function is used to test several combinations of fit at a time.

Usage

```
AutoFitPhenology(
  data = stop("A dataset must be provided"),
  progressbar = TRUE,
  ...
)
```

Arguments

data	Dataset generated with <code>add_phenology()</code>
progressbar	If FALSE, do not show the progress bar
...	Parameters for <code>fit_phenology()</code>

Details

AutoFitPhenology runs fit for phenology and tests several combinations

Value

A list with 12 elements corresponding to the 12 tested models

Author(s)

Marc Girondot

See Also

Other Phenology model: `BE_to_LBLE()`, `Gratiot`, `LBLE_to_BE()`, `LBLE_to_L()`, `L_to_LBLE()`, `MarineTurtles_2002`, `MinBMinE_to_Min()`, `adapt_parameters()`, `add_SE()`, `add_phenology()`, `extract_result()`, `fit_phenology()`, `likelihood_phenology()`, `logLik.phenology()`, `map_Gratiot`, `map_phenology()`, `par_init()`, `phenology()`, `phenology2fitRMU()`, `phenology_MHmcmc()`, `phenology_MHmcmc_p()`, `plot.phenology()`, `plot.phenologydata()`, `plot.phenologymap()`, `plot_delta()`, `plot_phi()`, `print.phenology()`, `print.phenologymap()`, `print.phenologyout()`, `remove_site()`, `result_Gratiot`, `result_Gratiot1`, `result_Gratiot2`, `result_Gratiot_Flat`, `summary.phenology()`, `summary.phenologymap()`, `summary.phenologyout()`

Examples

```
## Not run:
library(phenology)
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot <- add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Run the optimisation
result_Gratiot_Auto <- AutoFitPhenology(data=data_Gratiot)
result_Gratiot_Auto <- AutoFitPhenology(data=data_Gratiot,
  control=list(trace=0, REPORT=100, maxit=500))

## End(Not run)
```

Bayesian.remigration	<i>Return a posterior remigration interval.</i>
----------------------	---

Description

Model of remigration interval

Usage

```
Bayesian.remigration(
  parameters = stop("Priors must be supplied"),
  data = stop("data must be supplied"),
  kl = NULL,
  n.iter = 1e+05,
  n.chains = 1,
  n.adapt = 10000,
  thin = 1,
  trace = 10,
  adaptive = TRUE,
  adaptive.lag = 500,
  adaptive.fun = function(x) {
    ifelse(x > 0.234, 1.3, 0.7)
  },
  intermediate = NULL,
  filename = "intermediate.Rdata",
  previous = NULL
)
```

Arguments

parameters	Priors for Bayesian MCMC
------------	--------------------------

<code>data</code>	Data to be fitted
<code>kl</code>	Maximum number of years for remigration intervals.
<code>n.iter</code>	Number of iterations for MCMC
<code>n.chains</code>	Number of replicates
<code>n.adapt</code>	Number of iterations before to store outputs
<code>thin</code>	Number of iterations between each stored output
<code>trace</code>	Or FALSE or period to show progress
<code>adaptive</code>	Should an adaptive process for SDProp be used
<code>adaptive.lag</code>	Lag to analyze the SDProp value in an adaptive context
<code>adaptive.fun</code>	Function used to change the SDProp
<code>intermediate</code>	Period for saving intermediate result, NULL for no save
<code>filename</code>	If intermediate is not NULL, save intermediate result in this file
<code>previous</code>	Previous result to be continued. Can be the filename in which intermediate results are saved.

Details

Bayesian.remigration fits a remigration interval using Bayesian MCMC

Value

Return a posterior remigration interval.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Model of Remigration Interval: [LnRI_norm\(\)](#), [RI\(\)](#), [RI2BP\(\)](#), [plot.Remigration\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example

# Each year a fraction of 0.9 is surviving
s <- c(s=0.9)
# Probability of tag retention; 0.8
t <- c(t=0.8)
# Time-conditional return probability - This is the true remigration rate
r <- c(r1=0.1, r2=0.8, r3=0.7, r4=0.7, r5=1)
# Capture probability
p <- c(p1=0.6, p2=0.6, p3=0.6, p4=0.6, p5=0.6)
# Number of observations for 400 tagged females after 1, 2, 3, 4, and 5 years
```

```

OBS <- c(400, 10, 120, 40, 20, 10)

kl_s <- length(s)
kl_t <- length(t)
kl_r <- length(r)
kl_p <- length(p)

pMCMC <- data.frame(Density=c("newdbeta", "newdbeta", rep("dunif", kl_r),
                             rep("newdbeta", kl_p), "dunif"),
                    Prior1=c(s, t, rep(0, kl_r), rep(0.2, kl_p), 0),
                    Prior2=c(0.02, 0.02, rep(1, kl_r), rep(0.08, kl_p), 10),
                    SDProp=c(0.05, 0.05, rep(0.05, kl_r), rep(0.05, kl_p), 0.05),
                    Min=c(0, 0, rep(0, kl_r), rep(0, kl_p), 0),
                    Max=c(1, 1, rep(1, kl_r), rep(1, kl_p), 10),
                    Init=c(s, t, r, p, 1), stringsAsFactors = FALSE,
                    row.names=c("s",
                                "t",
                                names(r),
                                names(p), "sd")
                    )
rMCMC <- Bayesian.remigration(parameters = pMCMC,
n.iter = 1000000,
n.adapt = 300000,
trace=10000,
data=OBS)

plot(rMCMC)

## End(Not run)

```

BE_to_LBLE

Transform a set of parameters from Begin End to LengthB LengthE.

Description

This function is used to transform a set of parameters that uses Begin, Peak and End to a set of parameters that uses LengthB, Peak and LengthE.

Usage

```
BE_to_LBLE(parameters = NULL, help = FALSE)
```

Arguments

parameters	Set of current parameters
help	If TRUE, an help is displayed

Details

BE_to_LBLE transforms a set of parameters from Begin End format to LengthB LengthE.

Value

Return the set of modified parameters

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
## Not run:
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
refdate <- as.Date("2001-01-01")
data_Gratiot<-add_phenology(Gratiot, name="Complete",
  reference=refdate, format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Change the parameters to Begin End format
parg1<-LBLE_to_BE(parameters=parg)
# And change back to LengthB LengthE.
parg2<-BE_to_LBLE(parameters=parg1)

## End(Not run)
```

Description

The data must be a data.frame with the first column being years and two columns for each beach: the average and the se for the estimate.

The correspondence between mean, se and density for each rookery are given in the RMU.names data.frame.

This data.frame must have a column named mean, another named se and a third named density. If no sd column exists, no sd will be considered for the series and if no density column exists, it will be considered as being "dnorm".

In the result list, the mean proportions for each rookeries are in \$proportions.

The names of beach columns must not begin by T_, SD_, a0_, a1_ or a2_ and cannot be r.

A RMU is the acronym for Regional Management Unit. See:

Wallace, B.P., DiMatteo, A.D., Hurley, B.J., Finkbeiner, E.M., Bolten, A.B., Chaloupka, M.Y., Hutchinson, B.J., Abreu-Grobois, F.A., Amoroch, D., Bjorndal, K.A., Bourjea, J., Bowen, B.W., Dueñas, R.B., Casale, P., Choudhury, B.C., Costa, A., Dutton, P.H., Fallabrino, A., Girard, A., Girondot, M., Godfrey, M.H., Hamann, M., López-Mendilaharsu, M., Marcovaldi, M.A., Mortimer, J.A., Musick, J.A., Nel, R., Seminoff, J.A., Troëng, S., Witherington, B., Mast, R.B., 2010. Regional management units for marine turtles: a novel framework for prioritizing conservation and research across multiple scales. PLoS One 5, e15465.

Variance for each value is additive based on both the observed SE (in the RMU.data object) and a constant value dependent on the rookery when model.SD is equal to "Rookery-constant". The value is a global constant when model.SD is "global-constant". The value is proportional to the observed number of nests when model.SD is "global-proportional" with aSD_*observed+SD_ with aSD_ and SD_ being fitted values. This value is fixed to zero when model.SD is "Zero".

If method is NULL, replicate.CI is set to 0.

If method is hessian, it will generate replicate.CI random numbers using Hessian matrix with covariance.

If method is se, it will generate replicate.CI random numbers using SE values then without covariance.

If method is mcmc, it will generate replicate.CI random numbers using random samples of the MCMC or the regularThin number.

If replicate.CI is 0, no CI is estimated, and only point estimation is returned.

Usage

```
CI.RMU(
  result = stop("A result obtained from fitRMU is necessary"),
  resultMCMC = NULL,
  method = NULL,
  chain = 1,
  replicate.CI = 10000,
  regularThin = TRUE,
  probs = c(0.025, 0.5, 0.975),
  silent = FALSE
)
```

Arguments

result	A result of fitRMu()
resultMCMC	A results of fitRMU_MHmcmc()
method	Can be NULL, "SE", "Hessian", "MCMC", or "PseudoHessianFromMCMC"
chain	Number of MCMC chain to be used
replicate.CI	Number of replicates
regularThin	If TRUE, use regular thin for MCMC. If a number, take this replicate.
probs	The probabilities to return for quantiles
silent	If TRUE does not display anything

Details

CI.RMU calculates the confidence interval of the results of fitRMU()

Value

Return a list with Total, Proportions, and Numbers

Author(s)

Marc Girondot

See Also

Other Fill gaps in RMU: [fitRMU\(\)](#), [fitRMU_MHmcmc\(\)](#), [fitRMU_MHmcmc_p\(\)](#), [logLik.fitRMU\(\)](#), [plot.fitRMU\(\)](#)

Examples

```
## Not run:
library("phenology")
RMU.names.AtlanticW <- data.frame(mean=c("Yalimapo.French.Guiana",
                                         "Galibi.Suriname",
                                         "Irakumpapy.French.Guiana"),
                                   se=c("se_Yalimapo.French.Guiana",
                                         "se_Galibi.Suriname",
                                         "se_Irakumpapy.French.Guiana"),
                                   density=c("density_Yalimapo.French.Guiana",
                                             "density_Galibi.Suriname",
                                             "density_Irakumpapy.French.Guiana"),
                                   stringsAsFactors = FALSE)

data.AtlanticW <- data.frame(Year=c(1990:2000),
                               Yalimapo.French.Guiana=c(2076, 2765, 2890, 2678, NA,
                                                         6542, 5678, 1243, NA, 1566, 1566),
                               se_Yalimapo.French.Guiana=c(123.2, 27.7, 62.5, 126, NA,
                                                           230, 129, 167, NA, 145, 20),
                               density_Yalimapo.French.Guiana=rep("dnorm", 11),
                               Galibi.Suriname=c(276, 275, 290, NA, 267,
```

```

        542, 678, NA, 243, 156, 123),
se_Galibi.Suriname=c(22.3, 34.2, 23.2, NA, 23.2,
        4.3, 2.3, NA, 10.3, 10.1, 8.9),
density_Galibi.Suriname=rep("dnorm", 11),
Irakumpapy.French.Guiana=c(1076, 1765, 1390, 1678, NA,
        3542, 2678, 243, NA, 566, 566),
se_Irakumpapy.French.Guiana=c(23.2, 29.7, 22.5, 226, NA,
        130, 29, 67, NA, 15, 20),
density_Irakumpapy.French.Guiana=rep("dnorm", 11), stringsAsFactors = FALSE
)

cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
        colname.year="Year", model.trend="Constant",
        model.SD="Zero")

out.CI.Cst <- CI.RMU(result=cst)

cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
        colname.year="Year", model.trend="Constant",
        model.SD="Zero",
        control=list(trace=1, REPORT=100, maxit=500, parscale = c(3000, -0.2, 0.6)))

cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
        colname.year="Year", model.trend="Constant",
        model.SD="Zero", method=c("Nelder-Mead","BFGS"),
        control = list(trace = 0, REPORT = 100, maxit = 500,
        parscale = c(3000, -0.2, 0.6)))
expo <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
        colname.year="Year", model.trend="Exponential",
        model.SD="Zero", method=c("Nelder-Mead","BFGS"),
        control = list(trace = 0, REPORT = 100, maxit = 500,
        parscale = c(6000, -0.05, -0.25, 0.6)))
YS <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
        colname.year="Year", model.trend="Year-specific", method=c("Nelder-Mead","BFGS"),
        model.SD="Zero")
YS1 <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
        colname.year="Year", model.trend="Year-specific", method=c("Nelder-Mead","BFGS"),
        model.SD="Zero", model.rookeries="First-order")
YS1_cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
        colname.year="Year", model.trend="Year-specific",
        model.SD="Constant", model.rookeries="First-order",
        parameters=YS1$par, method=c("Nelder-Mead","BFGS"))
YS2 <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
        colname.year="Year", model.trend="Year-specific",
        model.SD="Zero", model.rookeries="Second-order",
        parameters=YS1$par, method=c("Nelder-Mead","BFGS"))
YS2_cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
        colname.year="Year", model.trend="Year-specific",
        model.SD="Constant", model.rookeries="Second-order",
        parameters=YS1_cst$par, method=c("Nelder-Mead","BFGS"))

```

```

compare_AIC(Constant=cst,
             Exponential=expo,
             YearSpecific=YS)

compare_AIC(YearSpecific_ProportionsFirstOrder_Zero=YS1,
            YearSpecific_ProportionsFirstOrder_Constant=YS1_cst)

compare_AIC(YearSpecific_ProportionsConstant=YS,
            YearSpecific_ProportionsFirstOrder=YS1,
            YearSpecific_ProportionsSecondOrder=YS2)

compare_AIC(YearSpecific_ProportionsFirstOrder=YS1_cst,
            YearSpecific_ProportionsSecondOrder=YS2_cst)

plot(cst, main="Use of different beaches along the time", what="total")
plot(expo, main="Use of different beaches along the time", what="total")
plot(YS2_cst, main="Use of different beaches along the time", what="total")

plot(YS1, main="Use of different beaches along the time")
plot(YS1_cst, main="Use of different beaches along the time")
plot(YS1_cst, main="Use of different beaches along the time", what="numbers")

# Gamma distribution should be used for MCMC outputs

RMU.names.AtlanticW <- data.frame(mean=c("Yalimapo.French.Guiana",
                                         "Galibi.Suriname",
                                         "Irakumpapy.French.Guiana"),
                                se=c("se_Yalimapo.French.Guiana",
                                       "se_Galibi.Suriname",
                                       "se_Irakumpapy.French.Guiana"),
                                density=c("density_Yalimapo.French.Guiana",
                                           "density_Galibi.Suriname",
                                           "density_Irakumpapy.French.Guiana"))

data.AtlanticW <- data.frame(Year=c(1990:2000),
                             Yalimapo.French.Guiana=c(2076, 2765, 2890, 2678, NA,
                                                         6542, 5678, 1243, NA, 1566, 1566),
                             se_Yalimapo.French.Guiana=c(123.2, 27.7, 62.5, 126, NA,
                                                         230, 129, 167, NA, 145, 20),
                             density_Yalimapo.French.Guiana=rep("dgamma", 11),
                             Galibi.Suriname=c(276, 275, 290, NA, 267,
                                                  542, 678, NA, 243, 156, 123),
                             se_Galibi.Suriname=c(22.3, 34.2, 23.2, NA, 23.2,
                                                  4.3, 2.3, NA, 10.3, 10.1, 8.9),
                             density_Galibi.Suriname=rep("dgamma", 11),
                             Irakumpapy.French.Guiana=c(1076, 1765, 1390, 1678, NA,
                                                         3542, 2678, 243, NA, 566, 566),
                             se_Irakumpapy.French.Guiana=c(23.2, 29.7, 22.5, 226, NA,
                                                         130, 29, 67, NA, 15, 20),
                             density_Irakumpapy.French.Guiana=rep("dgamma", 11)
                             )
cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
             colname.year="Year", model.trend="Constant",

```

```

model.SD="Zero")

## End(Not run)

```

ECFOCF_f

Calculate a table of probabilities of ECF and OCF.

Description

This function calculates a table of probabilities of ECF and OCF.
If p is lower or higher than $1E-100$ or $1-1E-100$, it is changed to $1E-100$ and $1-(1E-100)$ respectively.
Names for p vector elements should be p , or px (with $x=1:categories$), or $px.period$.
If mu_season and sd_season are equal to NA , the model is not temporalized.
If mu_season and sd_season are not NA , the model returns a 3D-table OCFECF.

Usage

```

ECFOCF_f(
  mu,
  sd = NA,
  p,
  MaxNests = 15,
  mu_season = NA,
  sd_season = NA,
  MeanDaysBetween2Nests = 9.8,
  length_season = floor(365/MeanDaysBetween2Nests) + 1,
  parallel = TRUE
)

```

Arguments

<code>mu</code>	The average of lognormal for clutch frequency.
<code>sd</code>	The sd parameter of lognormal for clutch frequency.
<code>p</code>	The capture probability for an individual nesting event. As a probability.
<code>MaxNests</code>	Maximum number of nests by a female.
<code>mu_season</code>	The average of ordinal day for beginning of nesting season.
<code>sd_season</code>	The sd parameter of lognormal for ordinal day for beginning of nesting season.
<code>MeanDaysBetween2Nests</code>	Average number of days between two nests.
<code>length_season</code>	The total length of season based on groups of interclutch intervals.
<code>parallel</code>	If TRUE parallel computing is used.

Details

ECFOCF_f calculate a table of probabilities of ECF and OCF.

Value

Return a matrix of class TableECFOCF.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Model of Clutch Frequency: [ECFOCF_full\(\)](#), [TableECFOCF\(\)](#), [fitCF\(\)](#), [fitCF_MHmcmc\(\)](#), [fitCF_MHmcmc_p\(\)](#), [generateCF\(\)](#), [lnLCF\(\)](#), [logLik.ECFOCF\(\)](#), [plot.ECFOCF\(\)](#), [plot.TableECFOCF\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
modelECFOCF <- ECFOCF_f(mu=5.58013243236187,
                        sd=1.225581130238,
                        p=invlogit(1.3578137414575),
                        MaxNests=15)

plot(modelECFOCF)
modelECFOCF <- ECFOCF_f(mu=5.58013243236187,
                        sd=1.225581130238,
                        mu_season=12,
                        sd_season=2,
                        p=c(p1=invlogit(1.3578137414575)),
                        MaxNests=15,
                        MeanDaysBetween2Nests=9.8,
                        length_season=floor(365/9.8)+1)

plot(modelECFOCF, period=2)

## End(Not run)
```

ECFOCF_full

Calculate a table of probabilities of ECF and OCF.

Description

This function calculates a table of probabilities of ECF and OCF.

If p is lower or higher than 1E-100 or 1-1E-100, it is changed to 1E-100 and 1-(1E-100) respectively.

Names for p vector elements should be p, px, or px.period (with x=1:categories).

Take care, p are logit, a and OTN are probabilities.

If mu_season and sd_season are equal to NA, the model is not temporalized.
 If mu_season and sd_season are not NA, the model returns a 3D-table OCFECF.

Usage

```
ECFOCF_full(
  mu,
  sd = NA,
  p,
  a = NULL,
  MaxNests = 15,
  mu_season = NA,
  sd_season = NA,
  OTN = c(OTN1 = 1),
  MeanDaysBetween2Nests = 9.8,
  length_season = NA,
  parallel = TRUE
)
```

Arguments

mu	The average of lognormal for clutch frequency.
sd	The sd parameter of lognormal for clutch frequency.
p	The capture probability for an individual nesting event. As a logit.
a	The common capture probability. As a probability
MaxNests	Maximum number of nests by a female.
mu_season	The average of ordinal day for beginning of nesting season.
sd_season	The sd parameter of lognormal for ordinal day for beginning of nesting season.
OTN	The relative probability of categories.
MeanDaysBetween2Nests	Average number of days between two nests.
length_season	The total length of season based on groups of interclutch intervals.
parallel	If TRUE parallel computing is used.

Details

ECFOCF_full calculate a table of probabilities of ECF and OCF.

Value

Return a matrix of class TableECFOCF.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Model of Clutch Frequency: [ECFOCF_f\(\)](#), [TableECFOCF\(\)](#), [fitCF\(\)](#), [fitCF_MHmcmc\(\)](#), [fitCF_MHmcmc_p\(\)](#), [generateCF\(\)](#), [lnLCF\(\)](#), [logLik.ECFOCF\(\)](#), [plot.ECFOCF\(\)](#), [plot.TableECFOCF\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example

modelECFOCF <- ECFOCF_full(mu=c(mu1=5.58013243236187),
                           sd=c(sd1=1.225581130238),
                           mu_season=c(mu_season1=12),
                           sd_season=c(sd_season1=2),
                           p=c(p1=logit(0.7954041)),
                           a=c(a1=1),
                           MaxNests=15,
                           MeanDaysBetween2Nests=9.8,
                           length_season=floor(365/9.8)+1)
plot(modelECFOCF, period=12, max.scale=0.02)
modelECFOCF <- ECFOCF_full(mu=c(mu1=5.58013243236187),
                           sd=c(sd1=1.225581130238),
                           a=c(a1=1),
                           p=c(p1=1.3578137414575),
                           MaxNests=15)

plot(modelECFOCF)

## End(Not run)
```

ExponentialRegression *Non-biased exponential regression*

Description

The idea of this function is to fit a regression exponential model using MCMC because regression model using glm can produce biased outputs.

prior.default.distribution can be "dnorm", "dunif", or "dgamma". Note that if you propose dgamma prior, it will use uniform prior for r because r can be negative.

The SD model is $asd * Nt + bsd$. The asd parameter represents multiplicative model and the bsd parameter represents additive model. Both can be used simultaneously.

density can be dnorm or dnbinom_new (from HelpersMG package). dnbinom_new() is a negative binomial with mean and sd parametrization.

Usage

```
ExponentialRegression(
  data = stop("A data.frame with values"),
  colname.time = "time",
```

```

  colname.number = "numbers",
  weights = NULL,
  fitted.parameters = c(N0 = NA, r = NA, asd = NA),
  fixed.parameters = NULL,
  n.iter = c(1e+05, 1e+05),
  prior.default.distribution = "dnorm",
  density = dnorm
)

```

Arguments

<code>data</code>	A data.frame with a column time and a column number.
<code>colname.time</code>	Name of the column to be used as time index.
<code>colname.number</code>	Name of the column to be used as number index.
<code>weights</code>	an optional vector of ‘prior weights’ to be used in the fitting process. Should be NULL or a numeric vector.
<code>fitted.parameters</code>	A named vector with the parameters to be fitted
<code>fixed.parameters</code>	A named vector with the parameters to be fixed
<code>n.iter</code>	The number of MCMC iterations.
<code>prior.default.distribution</code>	The default prior distribution; see description.
<code>density</code>	by default is dnorm but can be dnbinom_new

Details

ExponentialRegression is used to fit additive, multiplicative or mixte exponential regression

Value

Return a list with the results of exponential regression

Author(s)

Marc Girondot <marc.girondot@gmail.com>

Examples

```

## Not run:
library("phenology")
t <- 1:100
N0 <- 100
r <- 0.05
y <- N0*exp(t*r)

# Multiplicative model
Nt <- rnorm(100, mean=y, sd=0.2*y)

```

```

df <- data.frame(time=t, numbers=Nt)
g <- ExponentialRegression(data=df, fitted.parameters=c(N0=NA, r=NA, asd=NA))
plot(g, parameters="r")
as.parameters(g, index="median")
# Note that if you propose gamma prior, it will use uniform prior for r
# because r can be negative
g <- ExponentialRegression(data=df,
                           fitted.parameters=c(N0=NA, r=NA, asd=NA),
                           prior.default.distribution="dgamma")
plot(g, parameters="r")
as.parameters(g, index="median")

# Additive model
Nt <- rnorm(100, mean=y, sd=5)
df <- data.frame(time=t, numbers=Nt)
g <- ExponentialRegression(data=df, fitted.parameters=c(N0=NA, r=NA, bsd=NA))
plot(g, parameters="r")
as.parameters(g, index="median")

# Mixt model
Nt <- rnorm(100, mean=y, sd=0.2*y+5)
df <- data.frame(time=t, numbers=Nt)
g <- ExponentialRegression(data=df, fitted.parameters=c(N0=NA, r=NA, asd=NA, bsd=NA))
plot(g, parameters="r")
as.parameters(g, index="median")

# Example with 3 common ways to perform the regression
t <- 1:100
N0 <- 100
r <- 0.05
y <- N0*exp(t*r)
out_glm <- NULL
out_mcmc <- NULL
out_nls <- NULL
for (i in 1:500) {
  print(i)
  set.seed(i)
  Nt <- rnorm(100, mean=y, sd=0.2*y)
  df <- data.frame(time=t, numbers=Nt)
  g0 <- glm(log(numbers) ~ time, data = df)
  out_glm <- c(out_glm, c(exp(coef(g0)[1]), coef(g0)[2]))
  g1 <- ExponentialRegression(data=df, n.iter=c(10000, 20000))
  out_mcmc <- c(out_mcmc, as.parameters(g1, index="median")[1:2])
  g2 <- nls(numbers ~ N0*exp(r*time), start = list(N0 = 100, r = 0.05), data = df)
  out_nls <- c(out_nls, coef(g2))
}

# In conclusion the method proposed here has no biais as compare to glm and nls fits
out_glm <- matrix(out_glm, ncol=2, byrow=TRUE)
out_mcmc <- matrix(out_mcmc, ncol=2, byrow=TRUE)
out_nls <- matrix(out_nls, ncol=2, byrow=TRUE)
mean(out_glm[, 1]); mean(out_mcmc[, 1]); mean(out_nls[, 1])
sd(out_glm[, 1])/sqrt(nrow(out_glm)); sd(out_mcmc[, 1])/sqrt(nrow(out_mcmc));
sd(out_nls[, 1])/sqrt(nrow(out_nls))

```

```

mean(out_glm[, 2]); mean(out_mcmc[, 2]); mean(out_nls[, 2])
sd(out_glm[, 2])/sqrt(nrow(out_glm)); sd(out_mcmc[, 2])/sqrt(nrow(out_mcmc));
sd(out_nls[, 2])/sqrt(nrow(out_nls))

## End(Not run)

```

extract_result	<i>Extract the set of parameters from a result object.</i>
----------------	--

Description

The function "extract_result" permits to extract the set of parameters from a result object obtained after fit_phenology.

Usage

```
extract_result(result = NULL)
```

Arguments

result	A result file
--------	---------------

Details

extract_result get the fitted parameters from a result object.

Value

Return the set of fitted parameters

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
library(phenology)
## Not run:
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot<-add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
# result_Gratiot<-fit_phenology(data=data_Gratiot, fitted.parameters=parg,
  fixed.parameters=NULL)
data(result_Gratiot)
# Extract the fitted parameters
parg1<-extract_result(result_Gratiot)

## End(Not run)
```

fitCF

Fit a model of Clutch Frequency for marine turtles.

Description

This function fits a model of clutch frequency.

This model is an enhanced version of the one published by Briane et al. (2007).

Parameters are μ and σ being the parameters of a lognormal distribution used to model the clutch frequency.

This distribution is used only as a guide but has not statistical meaning.

The parameter p is the -logit probability that a female is seen on the beach for a particular nesting event. It includes both the probability that it is captured but also the probability that it uses that specific beach.

Several categories of females can be included in the model using index after the name of the parameter, for example μ_1 , σ_1 and μ_2 , σ_2 indicates that two categories of females with different clutch frequencies distribution are present. Similarly p_1 and p_2 indicates that two categories of females with different capture probabilities are present.

If more than one category is used, then it is necessary to include the parameter OTN to indicate the relative frequencies of each category. If two categories are used, one OTN parameter named OTN1 must be included. The OTN2 is forced to be 1. Then the relative frequency for category 1 is $OTN1/(OTN1+1)$ and for category 2 is $1/(OTN1+1)$. Same logic must be applied for 3 and more categories with always the last one being fixed to 1.

if p or a (-logit of the capture probability) are equal to $-\infty$, the probability of capture is 0 and if they are equal to $+\infty$, the probability is 1.

The value of p out of the period of nesting must be set to $+\infty$ (capture probability=1) to indicate that no turtle is nesting in this period.

p must be set to -Inf (capture probability=0) to indicate that no monitoring has been done during a specific period of the nesting season.

The best way to indicate capture probability for 3D model (OCF, ECF, Period) is to indicate p.period common for all categories and a1, a2, etc for each category. The capture probability for category 1 will be p.period * a1, and for category 2 will be p.period * a2, etc.

In this case, the parameters p.period should be indicated in fitted parameters as well as a1, but a2 must be fixed to +Inf in fixed.parameters. Then the capture probability for category 2 will be p.period and for category 1 a1 * p.period.

If itnmax is equal to 0, it will return the model using the parameters without fitting them.

Usage

```
fitCF(
  x = c(mu = 4, sd = 100, p = 0),
  fixed.parameters = NULL,
  data = stop("Data formatted with TableECFOCF() must be provided"),
  method = c("Nelder-Mead", "BFGS"),
  control = list(trace = 1, REPORT = 100, maxit = 500),
  itnmax = c(500, 100),
  hessian = TRUE,
  parallel = TRUE,
  verbose = FALSE
)
```

Arguments

x	Initial parameters to be fitted
fixed.parameters	Parameters that are fixed.
data	CMR data formatted using TableECFOCF()
method	Method to be used by optimx()
control	List of controls for optimx()
itnmax	A vector with maximum iterations for each method.
hessian	Logical to estimate SE of parameters
parallel	If TRUE, will use parallel computing for ECFOCF_f()
verbose	If TRUE, print the parameters at each step

Details

fitCF fit a model of Clutch Frequency for marine turtles.

Value

Return a list of class ECFOCF with the fit information.

The list has the following items:

- data: The observations to be fitted
- par: The fitted parameters
- SE: The standard error of parameters if hessian is TRUE
- value: The -log likelihood of observations within the fitted model
- AIC: The AIC of fitted model
- mu: The vector of fitted mu values
- sd: The vector of fitted sd values
- prob: The vector of fitted capture probabilities
- a: The vector of fitted capture probabilities multiplier
- OTN: The vector of fitted relative probabilities of contribution
- period_categories: A list with the different period probabilities as named vectors for each category
- period: The combined period probabilities using OTN as named vector
- CF_categories: A list with the different CF probabilities as named vectors for each category
- CF: The combined CF probabilities using OTN as named vector
- ECFOCF_categories: A list with the different probability ECFOCF tables for each category
- ECFOCF: The combined table of ECFOCF using OTN probabilities tables
- ECFOCF_0: The combined table of ECFOCF probabilities tables using OTN without the OCF=0
- SE_df: A data.frame with SE and 95\

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Briane J-P, Rivalan P, Girondot M (2007) The inverse problem applied to the Observed Clutch Frequency of Leatherbacks from Yalimapo beach, French Guiana. *Chelonian Conservation and Biology* 6:63-69

Fossette S, Kelle L, Girondot M, Goverse E, Hilterman ML, Verhage B, Thoisy B, de, Georges J-Y (2008) The world's largest leatherback rookeries: A review of conservation-oriented research in French Guiana/Suriname and Gabon. *Journal of Experimental Marine Biology and Ecology* 356:69-82

Other Model of Clutch Frequency: [ECFOCF_f\(\)](#), [ECFOCF_full\(\)](#), [TableECFOCF\(\)](#), [fitCF_MHmcmc\(\)](#), [fitCF_MHmcmc_p\(\)](#), [generateCF\(\)](#), [lnLCF\(\)](#), [logLik.ECFOCF\(\)](#), [plot.ECFOCF\(\)](#), [plot.TableECFOCF\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
data(MarineTurtles_2002)
ECFOCF_2002 <- TableECFOCF(MarineTurtles_2002)

# Parametric model for clutch frequency
o_mu1p1_Cfp <- fitCF(x = c(mu = 2.1653229641404539,
  sd = 1.1465246643327098,
  p = 0.25785366120357966),
  fixed.parameters=NULL,
  data=ECFOCF_2002, hessian = TRUE)

# Non parametric model for clutch frequency
o_mu1p1_CFn <- fitCF(x = c(mu.1 = 18.246619595610383,
  mu.2 = 4.2702163522832892,
  mu.3 = 2.6289986859556458,
  mu.4 = 3.2496360919228611,
  mu.5 = 2.1602522716550943,
  mu.6 = 0.68617023351032846,
  mu.7 = 4.2623607001877026,
  mu.8 = 1.1805600042630455,
  mu.9 = 2.2786176350939731,
  mu.10 = 0.47676265496204945,
  mu.11 = 5.8988238539197062e-08,
  mu.12 = 1.4003187851424953e-07,
  mu.13 = 2.4128444894899776e-07,
  mu.14 = 2.4223748020049825e-07,
  p = 0.32094401970037578),
  fixed.parameters=c(mu.15 = 1E-10),
  data=ECFOCF_2002, hessian = TRUE)

o_mu2p1 <- fitCF(x = c(mu1 = 1.2190766766978423,
  sd1 = 0.80646454821956925,
  mu2 = 7.1886819592223246,
  sd2 = 0.18152887523015518,
  p = 0.29347220802963259,
  OTN = 2.9137627675219533),
  fixed.parameters=NULL,
  data=ECFOCF_2002, hessian = TRUE)

o_mu1p2 <- fitCF(x = c(mu = 5.3628701816871462,
  sd = 0.39390555498088764,
  p1 = 0.61159637544418755,
  p2 = -2.4212753004659189,
  OTN = 0.31898004668901009),
  data=ECFOCF_2002, hessian = TRUE)

o_mu2p2 <- fitCF(x = c(mu1 = 0.043692606004492131,
  sd1 = 1.9446036983033428,
  mu2 = 7.3007868915644751,
```

```

sd2 = 0.16109296152913491,
p1 = 1.6860260469536992,
p2 = -0.096816113083788985,
OTN = 2.2604431232973501),
data=ECFOCF_2002, hessian = TRUE)

compare_AIC(mu1p1=o_mu1p1_CfP,
mu2p1=o_mu2p1,
mu1p2=o_mu1p2,
mu2p2=o_mu2p2)

o_mu3p3 <- fitCF(x = c(mu1 = 0.24286312214288761,
sd1 = 0.34542255091729313,
mu2 = 5.0817174343025551,
sd2 = 1.87435099405695,
mu3 = 5.2009265101740683,
sd3 = 1.79700447678357,
p1 = 8.8961708614726156,
p2 = 0.94790116453886453,
p3 = -0.76572930634505421,
OTN1 = 1.2936848663276974,
OTN2 = 0.81164278235645926),
data=ECFOCF_2002, hessian = TRUE)

o_mu3p1 <- fitCF(x = structure(c(0.24387978183477,
1.2639261745506,
4.94288464711349,
1.945082889758,
4.9431672350811,
1.287663104591,
0.323636536050397,
1.37072039291397,
9.28055412564559e-06),
names = c("mu1", "sd1", "mu2",
"sd2", "mu3", "sd3",
"p", "OTN1", "OTN2")),
data=ECFOCF_2002, hessian = TRUE)

o_mu1p3 <- fitCF(x = structure(c(4.65792402108387,
1.58445909785,
-2.35414198317177,
0.623757854800649,
-3.62623634029326,
11.6950204755787,
4.05273728846523),
names = c("mu", "sd",
"p1", "p2", "p3",
"OTN1", "OTN2")),
data=ECFOCF_2002, hessian = TRUE)

compare_AIC(mu1p1=o_mu1p1,

```

```

mu2p1=o_mu2p1,
mu1p2=o_mu1p2,
mu2p2=o_mu2p2,
mu3p3=o_mu3p3,
mu1p3=o_mu1p3,
mu3p1=o_mu3p1)

# 3D model for (ECF, OCF, period)

ECFOCF_2002 <- TableECFOCF(MarineTurtles_2002,
                           date0=as.Date("2002-01-01"))

fp <- rep(0, dim(ECFOCF_2002)[3])
names(fp) <- paste0("p.", formatC(1:(dim(ECFOCF_2002)[3]), width=2, flag="0"))
par <- c(mu = 2.6404831115214353,
        sd = 0.69362774786433479,
        mu_season = 12.6404831115214353,
        sd_season = 1.69362774786433479)
par <- c(par, fp[attributes(ECFOCF_2002)$table["begin"]:
                  attributes(ECFOCF_2002)$table["final"]])

# The value of p (logit of the capture probability) out of the period
# of nesting must be set to +Inf (capture probability=1)
# to indicate that no turtle is nesting in this period

# p must be set to -Inf (capture probability=0) to indicate that no
# monitoring has been done during a specific period of the nesting season.

fixed.parameters <- c(p=+Inf)
# The fitted values are:
par <- c(mu = 2.4911638591178051,
        sd = 0.96855483039640977,
        mu_season = 13.836059118657793,
        sd_season = 0.17440085345943984,
        p.10 = 1.3348233607728222,
        p.11 = 1.1960387774393837,
        p.12 = 0.63025680979544774,
        p.13 = 0.38648155002707452,
        p.14 = 0.31547864054366048,
        p.15 = 0.19720001827017075,
        p.16 = 0.083199496372073328,
        p.17 = 0.32969130595897905,
        p.18 = 0.36582777525265819,
        p.19 = 0.30301248314170637,
        p.20 = 0.69993987591518514,
        p.21 = 0.13642423871641118,
        p.22 = -1.3949268190534629)

o_mu1p1season1 <- fitCF(x=par, data=ECFOCF_2002,
                        fixed.parameters=fixed.parameters)

# Same model but with two different models of capture probabilities

```

```

fp <- rep(0, dim(ECFOCF_2002)[3])
names(fp) <- paste0("p1.", formatC(1:(dim(ECFOCF_2002)[3]), width=2, flag="0"))
par <- c(mu = 2.6404831115214353,
        sd = 0.69362774786433479,
        mu_season = 12.6404831115214353,
        sd_season = 1.69362774786433479)
par <- c(par, fp[attributes(ECFOCF_2002)$table["begin"]]:
        attributes(ECFOCF_2002)$table["final"]])
names(fp) <- paste0("p2.", formatC(1:(dim(ECFOCF_2002)[3]), width=2, flag="0"))
par <- c(par, fp[attributes(ECFOCF_2002)$table["begin"]]:
        attributes(ECFOCF_2002)$table["final"]])
fixed.parameters <- c(p1=+Inf, p2=+Inf)

o_mulp2season1 <- fitCF(x=par, data=ECFOCF_2002,
                      fixed.parameters=fixed.parameters)

# Here the two different capture probabilities are different
# by a constant:
# p1=invlogit(-p)      [Note that invlogit(-a1) = 1]
# p2=invlogit(-p)*invlogit(-a2)

fp <- rep(0, dim(ECFOCF_2002)[3])
names(fp) <- paste0("p.", formatC(1:(dim(ECFOCF_2002)[3]), width=2, flag="0"))
par <- c(mu = 2.6404831115214353,
        sd = 0.69362774786433479,
        mu_season = 12.6404831115214353,
        sd_season = 1.69362774786433479,
        a2=0)
par <- c(par, fp[attributes(ECFOCF_2002)$table["begin"]]:
        attributes(ECFOCF_2002)$table["final"]])
fixed.parameters <- c(a1=+Inf, p=+Inf)

o_mulp1aseason1 <- fitCF(x=par, data=ECFOCF_2002,
                      fixed.parameters=fixed.parameters)
                      data=ECFOCF_2002)

## End(Not run)

```

fitCF_MHmcmc

Run the Metropolis-Hastings algorithm for ECFOCF data

Description

Run the Metropolis-Hastings algorithm for ECFOCF data.

The number of iterations is $n.iter + n.adapt + 1$ because the initial likelihood is also displayed.

I recommend $thin=1$ because the method to estimate SE uses resampling.

As initial point is maximum likelihood, $n.adapt = 0$ is a good solution.

The parameters intermediate and filename are used to save intermediate results every 'intermediate' iterations (for example 1000). Results are saved in a file of name filename.

The parameter previous is used to indicate the list that has been save using the parameters intermediate and filename. It permits to continue a mcmc search.

These options are used to prevent the consequences of computer crash or if the run is very very long and computer processes at time limited.

Usage

```
fitCF_MHmcmc(
  result = stop("An output from fitCF() must be provided"),
  n.iter = 10000,
  parametersMCMC = stop("A parameter set from fitCF_MHmcmc_p() must be provided"),
  n.chains = 1,
  n.adapt = 0,
  thin = 1,
  WAIC = FALSE,
  adaptive = FALSE,
  adaptive.lag = 500,
  adaptive.fun = function(x) {
    ifelse(x > 0.234, 1.3, 0.7)
  },
  trace = FALSE,
  traceML = FALSE,
  intermediate = NULL,
  filename = "intermediate.Rdata",
  previous = NULL
)
```

Arguments

result	An object obtained after a SearchR fit
n.iter	Number of iterations for each step
parametersMCMC	A set of parameters used as initial point for searching with information on priors
n.chains	Number of replicates
n.adapt	Number of iterations before to store outputs
thin	Number of iterations between each stored output
WAIC	Should WAIC information be saved?
adaptive	Should an adaptive process for SDProp be used
adaptive.lag	Lag to analyze the SDProp value in an adaptive content
adaptive.fun	Function used to change the SDProp
trace	TRUE or FALSE or period, shows progress
traceML	TRUE or FALSE to show ML
intermediate	Period for saving intermediate result, NULL for no save
filename	If intermediate is not NULL, save intermediate result in this file
previous	Previous result to be continued. Can be the filename in which intermediate results are saved.

Details

fitCF_MHmcmc runs the Metropolis-Hastings algorithm for ECFOCF (Bayesian MCMC)

Value

A list with resultMCMC being mcmc.list object, resultLnL being likelihoods and parametersMCMC being the parameters used

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Model of Clutch Frequency: [ECFOCF_f\(\)](#), [ECFOCF_full\(\)](#), [TableECFOCF\(\)](#), [fitCF\(\)](#), [fitCF_MHmcmc_p\(\)](#), [generateCF\(\)](#), [lnLCF\(\)](#), [logLik.ECFOCF\(\)](#), [plot.ECFOCF\(\)](#), [plot.TableECFOCF\(\)](#)

Examples

```
## Not run:
library("phenology")
data(MarineTurtles_2002)
ECFOCF_2002 <- TableECFOCF(MarineTurtles_2002)

# Paraetric model for clutch frequency
o_mu1p1_CFp <- fitCF(x = c(mu = 2.1653229641404539,
  sd = 1.1465246643327098,
  p = 0.25785366120357966),
  fixed.parameters=NULL,
  data=ECFOCF_2002, hessian = TRUE)

pMCMC <- fitCF_MHmcmc_p(result=o_mu1p1_CFp, accept=TRUE)
fitCF_MCMC <- fitCF_MHmcmc(result = o_mu1p1_CFp, n.iter = 1000,
  parametersMCMC = pMCMC, n.chains = 1, n.adapt = 0,
  adaptive=TRUE, WAIC = TRUE,
  thin = 1, trace = TRUE)

plot(fitCF_MCMC, parameters="mu")
plot(fitCF_MCMC, parameters="sd")
plot(fitCF_MCMC, parameters="p", xlim=c(0, 0.5), breaks=seq(from=0, to=0.5, by=0.05))
plot(fitCF_MCMC, parameters="p", transform = invlogit, xlim=c(0, 1),
  breaks=c(seq(from=0, to=1, by=0.05)))

## End(Not run)
```

fitCF_MHmcmc_p	<i>Generate set of parameters to be used with fitCF_MHmcmc()</i>
----------------	--

Description

Interactive script used to generate set of parameters to be used with fitCF_MHmcmc().

Usage

```
fitCF_MHmcmc_p(
  result = stop("An output from fitCF() must be provided"),
  density = "dunif",
  accept = FALSE
)
```

Arguments

result	An object obtained after a fitCF() fit
density	Preset of density; can be dnorm or dunif
accept	If TRUE, does not wait for user interaction

Details

fitCF_MHmcmc_p generates set of parameters to be used with fitCF_MHmcmc()

Value

A matrix with the parameters

Author(s)

Marc Girondot

See Also

Other Model of Clutch Frequency: [ECFOCF_f\(\)](#), [ECFOCF_full\(\)](#), [TableECFOCF\(\)](#), [fitCF\(\)](#), [fitCF_MHmcmc\(\)](#), [generateCF\(\)](#), [lnLCF\(\)](#), [logLik.ECFOCF\(\)](#), [plot.ECFOCF\(\)](#), [plot.TableECFOCF\(\)](#)

Examples

```
## Not run:
library("phenology")
data(MarineTurtles_2002)
ECFOCF_2002 <- TableECFOCF(MarineTurtles_2002)

# Paraetric model for clutch frequency
o_mulp1_CFp <- fitCF(x = c(mu = 2.1653229641404539,
```

```

sd = 1.1465246643327098,
p = 0.25785366120357966),
fixed.parameters=NULL,
data=ECFOCF_2002, hessian = TRUE)

pMCMC <- fitCF_MHmcmc_p(result=o_mu1p1_CFp, accept=TRUE)
fitCF_MCMC <- fitCF_MHmcmc(result = o_mu1p1_CFp, n.iter = 10000,
                           parametersMCMC = pMCMC, n.chains = 1, n.adapt = 0,
                           thin = 1, trace = FALSE)
plot(fitCF_MCMC, parameters="mu")
plot(fitCF_MCMC, parameters="sd")
plot(fitCF_MCMC, parameters="p", xlim=c(0, 0.5), breaks=seq(from=0, to=0.5, by=0.05))
plot(fitCF_MCMC, parameters="p", transform = invlogit, xlim=c(0, 1),
     breaks=c(seq(from=0, to=1, by=0.05)))

## End(Not run)

```

fitRMU

Adjust incomplete timeseries with various constraints.

Description

The data must be a data.frame with the first column being years and two columns for each beach: the average and the se for the estimate.

The correspondence between mean, se and density for each rookery are given in the RMU.names data.frame.

This data.frame must have a column named mean, another named se and a third named density. If no sd column exists, no sd will be considered for the series and if no density column exists, it will be considered as being "dnorm" (Gaussian distribution).

The aggregated number of nests and its confidence interval can be obtained using CI.RMU().

The names of beach columns must not begin by T_, SD_, a0_, a1_ or a2_ and cannot be r.

A RMU is the acronym for Regional Management Unit. See:

Wallace, B.P., DiMatteo, A.D., Hurley, B.J., Finkbeiner, E.M., Bolten, A.B., Chaloupka, M.Y., Hutchinson, B.J., Abreu-Grobois, F.A., Amorocho, D., Bjørndal, K.A., Bourjé, J., Bowen, B.W., Dueñas, R.B., Casale, P., Choudhury, B.C., Costa, A., Dutton, P.H., Fallabrino, A., Girard, A., Girondot, M., Godfrey, M.H., Hamann, M., López-Mendilaharsu, M., Marcovaldi, M.A., Mortimer, J.A., Musick, J.A., Nel, R., Seminoff, J.A., Tröng, S., Witherington, B., Mast, R.B., 2010. Regional management units for marine turtles: a novel framework for prioritizing conservation and research across multiple scales. PLoS One 5, e15465.

Variance for each value is additive based on both the observed SE (in the RMU.data object) and a value.

The value is a global constant when model.SD is "global-constant".

The value is proportional to the observed number of nests when model.SD is "global-proportional" with $aSD_ * observed + SD_$ with $aSD_$ and $SD_$ being fitted values. This value is fixed to zero when model.SD is "Zero".

The value is dependent on the rookery when model.SD is equal to "Rookery-constant" or "Rookery-proportional" with a similar formula as previously described for "global".

if method is NULL, it will simply return the names of required parameters.

Usage

```

fitRMU(
  RMU.data = stop("data parameter must be provided"),
  years.byrow = TRUE,
  RMU.names = NULL,
  model.trend = "Constant",
  model.rookeries = "Constant",
  model.SD = "Global-constant",
  parameters = NULL,
  fixed.parameters = NULL,
  SE = NULL,
  method = c("Nelder-Mead", "BFGS"),
  control = list(trace = 1),
  itnmax = c(1500, 1500),
  cptmax.optim = 100,
  limit.cpt.optim = 1e-05,
  hessian = TRUE,
  replicate.CI = 1000,
  colname.year = "Year",
  maxL = 1e+09
)

```

Arguments

RMU.data	A data.frame with a column Year (the name is defined in colname.year) and one to three columns per rookery defined in RMU.names
years.byrow	If TRUE, the RMU.data data.frame is organized with years in rows
RMU.names	A dataframe with one to three columns indicating name of columns for mean, standard deviation, and distribution for roockeris
model.trend	Can be Constant, Exponential or Year-specific
model.rookeries	Description temporal change in rookeries proportion. It be Constant, First-order or Second-order
model.SD	Can be Zero, Global-constant, Global-proportional or Rookery-constant. See description.
parameters	Parameters to fit
fixed.parameters	Parameters that are fixed
SE	Parameters SE for example from fitRMU_MHmcmc()
method	Methods to be used by optim()
control	List of controls for optim()
itnmax	A vector with maximum iterations for each method.
cptmax.optim	How many times optim can be ran when likelihood is better.
limit.cpt.optim	Limit to consider that likelihood is better.


```

se_Irakumpapy.French.Guiana=c(23.2, 29.7, 22.5, 226, NA,
                             130, 29, 67, NA, 15, 20),
density_Irakumpapy.French.Guiana=rep("dnorm", 11)
)

cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
             colname.year="Year", model.trend="Constant",
             model.SD="Zero")
cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
             colname.year="Year", model.trend="Constant",
             model.SD="Zero",
             control=list(trace=1, REPORT=100, maxit=500, parscale = c(3000, -0.2, 0.6)))

cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
             colname.year="Year", model.trend="Constant",
             model.SD="Zero", method=c("Nelder-Mead","BFGS"),
             control = list(trace = 0, REPORT = 100, maxit = 500,
                             parscale = c(3000, -0.2, 0.6)))
expo <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
             colname.year="Year", model.trend="Exponential",
             model.SD="Zero", method=c("Nelder-Mead","BFGS"),
             control = list(trace = 0, REPORT = 100, maxit = 500,
                             parscale = c(6000, -0.05, -0.25, 0.6)))
YS <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
             colname.year="Year", model.trend="Year-specific", method=c("Nelder-Mead","BFGS"),
             model.SD="Zero")
YS1 <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
             colname.year="Year", model.trend="Year-specific", method=c("Nelder-Mead","BFGS"),
             model.SD="Zero", model.rookeries="First-order")
YS1_cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
             colname.year="Year", model.trend="Year-specific",
             model.SD="Constant", model.rookeries="First-order",
             parameters=YS1$par, method=c("Nelder-Mead","BFGS"))
YS2 <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
             colname.year="Year", model.trend="Year-specific",
             model.SD="Zero", model.rookeries="Second-order",
             parameters=YS1$par, method=c("Nelder-Mead","BFGS"))
YS2_cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
             colname.year="Year", model.trend="Year-specific",
             model.SD="Constant", model.rookeries="Second-order",
             parameters=YS1_cst$par, method=c("Nelder-Mead","BFGS"))

compare_AIC(Constant=cst, Exponential=expo,
YearSpecific=YS)

compare_AIC(YearSpecific_ProportionsFirstOrder_Zero=YS1,
YearSpecific_ProportionsFirstOrder_Constant=YS1_cst)

compare_AIC(YearSpecific_ProportionsConstant=YS,
             YearSpecific_ProportionsFirstOrder=YS1,
             YearSpecific_ProportionsSecondOrder=YS2)

compare_AIC(YearSpecific_ProportionsFirstOrder=YS1_cst,

```

```

YearSpecific_ProportionsSecondOrder=YS2_cst)

# Example of different types of plots
plot(cst, main="Use of different beaches along the time", what="total",
      ylim=c(0, 4000))
plot(cst, main="Use of different beaches along the time", what = "proportions",
      replicate.CI=0)
plot(cst, main="Use of different beaches along the time", what = "numbers",
      aggregate="model", ylim=c(0, 4000), replicate.CI=0)
plot(cst, main="Use of different beaches along the time", what = "numbers",
      aggregate="both", ylim=c(0, 11000), replicate.CI=0)

plot(expo, main="Use of different beaches along the time", what="total")
plot(YS2_cst, main="Use of different beaches along the time", what="total")

plot(YS1, main="Use of different beaches along the time")
plot(YS1_cst, main="Use of different beaches along the time")
plot(YS1_cst, main="Use of different beaches along the time", what="numbers")

# Gamma distribution should be used for MCMC outputs

RMU.names.AtlanticW <- data.frame(mean=c("Yalimapo.French.Guiana",
                                         "Galibi.Suriname",
                                         "Irakumpapy.French.Guiana"),
                                  se=c("se_Yalimapo.French.Guiana",
                                         "se_Galibi.Suriname",
                                         "se_Irakumpapy.French.Guiana"),
                                  density=c("density_Yalimapo.French.Guiana",
                                         "density_Galibi.Suriname",
                                         "density_Irakumpapy.French.Guiana"),
                                  stringsAsFactors = FALSE)

data.AtlanticW <- data.frame(Year=c(1990:2000),
                              Yalimapo.French.Guiana=c(2076, 2765, 2890, 2678, NA,
                                                         6542, 5678, 1243, NA, 1566, 1566),
                              se_Yalimapo.French.Guiana=c(123.2, 27.7, 62.5, 126, NA,
                                                         230, 129, 167, NA, 145, 20),
                              density_Yalimapo.French.Guiana=rep("dgamma", 11),
                              Galibi.Suriname=c(276, 275, 290, NA, 267,
                                                  542, 678, NA, 243, 156, 123),
                              se_Galibi.Suriname=c(22.3, 34.2, 23.2, NA, 23.2,
                                                  4.3, 2.3, NA, 10.3, 10.1, 8.9),
                              density_Galibi.Suriname=rep("dgamma", 11),
                              Irakumpapy.French.Guiana=c(1076, 1765, 1390, 1678, NA,
                                                         3542, 2678, 243, NA, 566, 566),
                              se_Irakumpapy.French.Guiana=c(23.2, 29.7, 22.5, 226, NA,
                                                         130, 29, 67, NA, 15, 20),
                              density_Irakumpapy.French.Guiana=rep("dgamma", 11), stringsAsFactors = FALSE
                              )
cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
              colname.year="Year", model.trend="Constant",
              model.SD="Zero")

```

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

```
fitRMU_MHmcmc
```

```
Run the Metropolis-Hastings algorithm for RMU.data
```

Description

Run the Metropolis-Hastings algorithm for RMU.data.

The number of iterations is `n.iter+n.adapt+1` because the initial likelihood is also displayed.

I recommend `thin=1` because the method to estimate SE uses resampling.

As initial point is maximum likelihood, `n.adapt = 0` is a good solution.

The parameters `intermediate` and `filename` are used to save intermediate results every 'intermediate' iterations (for example 1000). Results are saved in a file of name `filename`.

The parameter `previous` is used to indicate the list that has been save using the parameters `intermediate` and `filename`. It permits to continue a mcmc search.

These options are used to prevent the consequences of computer crash or if the run is very very long and computer processes at time limited.

Usage

```
fitRMU_MHmcmc(
  result = stop("An output from fitRMU() must be provided"),
  n.iter = 10000,
  parametersMCMC = stop("A parameter set from fitRMU_MHmcmc_p() must be provided"),
  n.chains = 1,
  n.adapt = 0,
  thin = 1,
  adaptive = FALSE,
  adaptive.lag = 500,
  adaptive.fun = function(x) {
    ifelse(x > 0.234, 1.3, 0.7)
  },
  trace = FALSE,
  traceML = FALSE,
  intermediate = NULL,
  filename = "intermediate.Rdata",
  previous = NULL
)
```

Arguments

<code>result</code>	An object obtained after a SearchR fit
<code>n.iter</code>	Number of iterations for each step
<code>parametersMCMC</code>	A set of parameters used as initial point for searching with information on priors
<code>n.chains</code>	Number of replicates


```

Irakumpapy.French.Guiana=c(1076, 1765, 1390, 1678, NA,
                           3542, 2678, 243, NA, 566, 566),
se_Irakumpapy.French.Guiana=c(23.2, 29.7, 22.5, 226, NA,
                              130, 29, 67, NA, 15, 20))

cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
             colname.year="Year", model.trend="Constant",
             model.SD="Zero")
pMCMC <- fitRMU_MHmcmc_p(result=cst, accept=TRUE)
fitRMU_MCMC <- fitRMU_MHmcmc(result = cst, n.iter = 10000,
                             parametersMCMC = pMCMC, n.chains = 1, n.adapt = 0, thin = 1, trace = FALSE)

## End(Not run)

```

fitRMU_MHmcmc_p	<i>Generates set of parameters to be used with fitRMU_MHmcmc()</i>
-----------------	--

Description

Interactive or automatic script used to generate set of parameters to be used with fitRMU_MHmcmc(). If density="dgamma" is used, a uniform distribution is used for r, as r can be negative.

Usage

```

fitRMU_MHmcmc_p(
  result = stop("An output from fitRMU() must be provided"),
  density = "dunif",
  accept = FALSE
)

```

Arguments

result	An object obtained after a fitRMU() fit
density	Preset of density; can be dnorm, dunif, or dgamma
accept	If TRUE, does not wait for user interaction

Details

fitRMU_MHmcmc_p generates set of parameters to be used with fitRMU_MHmcmc()

Value

A matrix with the parameters

Author(s)

Marc Girondot

See Also

Other Fill gaps in RMU: [CI.RMU\(\)](#), [fitRMU\(\)](#), [fitRMU_MHmcmc\(\)](#), [logLik.fitRMU\(\)](#), [plot.fitRMU\(\)](#)

Examples

```
## Not run:
library("phenology")
RMU.name.AtlanticW <- data.frame(mean=c("Yalimapo.French.Guiana",
                                       "Galibi.Suriname",
                                       "Irakumpapy.French.Guiana"),
                                se=c("se_Yalimapo.French.Guiana",
                                      "se_Galibi.Suriname",
                                      "se_Irakumpapy.French.Guiana"))

data.AtlanticW <- data.frame(Year=c(1990:2000),
                              Yalimapo.French.Guiana=c(2076, 2765, 2890, 2678, NA,
                                                         6542, 5678, 1243, NA, 1566, 1566),
                              se_Yalimapo.French.Guiana=c(123.2, 27.7, 62.5, 126, NA,
                                                           230, 129, 167, NA, 145, 20),
                              Galibi.Suriname=c(276, 275, 290, NA, 267,
                                                  542, 678, NA, 243, 156, 123),
                              se_Galibi.Suriname=c(22.3, 34.2, 23.2, NA, 23.2,
                                                     4.3, 2.3, NA, 10.3, 10.1, 8.9),
                              Irakumpapy.French.Guiana=c(1076, 1765, 1390, 1678, NA,
                                                         3542, 2678, 243, NA, 566, 566),
                              se_Irakumpapy.French.Guiana=c(23.2, 29.7, 22.5, 226, NA,
                                                            130, 29, 67, NA, 15, 20))

cst <- fitRMU(data=data.AtlanticW, RMU.name=RMU.name.AtlanticW,
              colname.year="Year", model.trend="Constant",
              model.SD="Zero")
pMCMC <- fitRMU_MHmcmc_p(result=cst, accept=TRUE)

## End(Not run)
```

fit_phenology

Fit the phenology parameters to timeseries of counts.

Description

Function of the package phenology to fit parameters to timeseries.

To fit data, the syntax is :

```
Result <- fit_phenology(data=dataset, fitted.parameters=par, fixed.parameters=pfixed, trace=1, hessian=TRUE)
```

or if no parameter is fixed :

```
Result <- fit_phenology(data=dataset, fitted.parameters=par)
```

Add trace=1 [default] to have information on the fit progression or trace=0 to hide information on the fit progression.

hessian = FALSE does not estimate Hessian matrix and SE of parameters.

If the parameter Theta is fixed to +Inf, a Poissonian model of daily nest distribution is implemented.

Special section about cofactors:

cofactors must be a data.frame with a column Date and a column for each cofactor

add.cofactors are the names of the column of parameter cofactors to use as a cofactor;

The model is then: $\text{parameter}[\text{add.cofactors}] * \text{cofactor[, add.cofactors]}$

If the name of the parameter is $\text{paste0}(\text{add.cofactors}, \text{"multi"})$, then the model is:

$\text{parameter}[\text{paste0}(\text{add.cofactors}, \text{"multi"})] * \text{cofactor[, add.cofactors]} * (\text{number of nests without cofactor})$

About parallel computing:

Set options mc.cores and forking to tell what sort of parallel computing

Example:

```
options(mc.cores = detectCores())
```

```
options(forking = FALSE)
```

Usage

```
fit_phenology(
  data = file.choose(),
  fitted.parameters = NULL,
  fixed.parameters = NULL,
  model_before = NULL,
  store.intermediate = FALSE,
  file.intermediate = "Intermediate.rda",
  hessian = FALSE,
  silent = FALSE,
  cofactors = NULL,
  add.cofactors = NULL,
  zero = 1e-09,
  lower = 0,
  upper = Inf,
  stop.fit = FALSE,
  method_Snbinom = "saddlepoint",
  control = list(trace = 1, REPORT = 1, maxit = 1000),
  method = c("Nelder-Mead", "L-BFGS-B")
)
```

Arguments

data	A dataset generated by add_format
fitted.parameters	Set of parameters to be fitted
fixed.parameters	Set of fixed parameters
model_before	The change of parameters before to estimate daily counts.
store.intermediate	TRUE or FALSE to save the intermediates
file.intermediate	Name of the file where to save the intermediates as a list
hessian	If FALSE does not estimate se of parameters

silent	If TRUE does not show any message
cofactors	data.frame with a column Date and a column for each cofactor
add.cofactors	Names of the column of parameter cofactors to use as a cofactor
zero	If the theoretical nest number is under this value, this value will be used
lower	Lower bound for each parameter
upper	Upper bound for each parameter
stop.fit	If TRUE, will stop search for parameters even if not ML
method_Snbinom	Can be Furman, exact, or saddlepoint.
control	List for control parameters for optim
method	Method used by optim. Several can be setup.

Details

fit_phenology fits parameters to timeseries.

Value

Return a list of with data and result

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
## Not run:
library(phenology)
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot <- add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg <- par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
result_Gratiot <- fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=NULL)
```

```

data(result_Gratiot)
# Plot the phenology and get some stats
output <- plot(result_Gratiot)
# or
output <- summary(result_Gratiot)

# With one sinusoid
parg <- c('LengthB' = 95.187648986054285,
          'Peak' = 173.86111755419921,
          'LengthE' = 63.183281230994481,
          'Max_Complete' = 33.052091519419136,
          'MinB_Complete' = 0.21716081973776738,
          'MinE_Complete' = 0.42444245288475702,
          'Theta' = 3.9554976911657187,
          'Alpha' = 0,
          'Beta' = 0.21019683898423902,
          'Delta' = 3.6798422076201724,
          'Phi' = 10)
result_Gratiot1 <- fit_phenology(data=data_Gratiot,
                                fitted.parameters=parg, fixed.parameters=NULL)
plot(result_Gratiot1)

# With two sinusoids
parg <- c('LengthB' = 95.821859173220659,
          'Peak' = 174.89756503758881,
          'LengthE' = 60.954167489825352,
          'Max_Complete' = 33.497846416498774,
          'MinB_Complete' = 0.21331670808871037,
          'MinE_Complete' = 0.43730327416828613,
          'Theta' = 4.2569203480842814,
          'Alpha1' = 0.15305562092653918,
          'Beta1' = 0,
          'Delta1' = 9.435730952200263,
          'Phi1' = 15.7944494669335,
          'Alpha' = 0,
          'Beta' = 0.28212926001402688,
          'Delta' = 18.651087311957518,
          'Phi' = 9.7549929595313056)
result_Gratiot2 <- fit_phenology(data=data_Gratiot,
                                fitted.parameters=parg, fixed.parameters=NULL)
plot(result_Gratiot2)

compare_AICc(no=result_Gratiot,
             one=result_Gratiot1,
             two=result_Gratiot2)

# With parametrization based on Girondot 2010
parg <- c('Peak' = 173.52272236775076,
          'Flat' = 0,
          'LengthB' = 94.433284359804205,
          'LengthE' = 64.288485646867329,
          'Max_Complete' = 32.841568389778033,
          'PMin' = 1.0368261725650889,

```

```

      'Theta' = 3.5534818927979592)
result_Gratiot_par1 <- fit_phenology(data=data_Gratiot,
                                   fitted.parameters=parg, fixed.parameters=NULL)
# With new parametrization based on Omeyer et al. (2022):
# Omeyer, L. C. M., McKinley, T. J., Br  heret, N., Bal, G., Balchin, G. P.,
# Bitsindou, A., Chauvet, E., Collins, T., Curran, B. K., Formia, A., Girard, A.,
# Girondot, M., Godley, B. J., Mavoungou, J.-G., Poli, L., Tilley, D.,
# VanLeeuwe, H. & Metcalfe, K. 2022. Missing data in sea turtle population
# monitoring: a Bayesian statistical framework accounting for incomplete
# sampling Front. Mar. Sci. (IF 3.661), 9, 817014.

parg <- result_Gratiot_par1$par
parg <- c(tp=unname(parg["Peak"]), tf=unname(parg["Flat"]),
          s1=unname(parg["LengthB"])/4.8, s2=unname(parg["LengthE"])/4.8,
          alpha=unname(parg["Max_Complete"]), Theta=0.66)
result_Gratiot_par2 <- fit_phenology(data=data_Gratiot,
                                   fitted.parameters=parg,
                                   fixed.parameters=NULL)
compare_AICc(GirondotModel=result_Gratiot_par1,
             OmeyerModel=result_Gratiot_par2)
plot(result_Gratiot_par1)
plot(result_Gratiot_par2)

# Use fit with co-factor

# First extract tide information for that place
td <- tide.info(year=2001, latitude=4.9167, longitude=-52.3333)
# I keep only High tide level
td2 <- td[td$Phase=="High Tide", ]
# I get the date
td3 <- cbind(td2, Date=as.Date(td2$DateTime.local))
td5 <- aggregate(x=td3[, c("Date", "DateTime.local", "Tide.meter")],
                 by=list(Date=td3[, "Date"]), FUN=max)[, 2:4]
with(td5, plot(DateTime.local, Tide.meter, type="l"))
td6 <- td5[, c("Date", "Tide.meter")]
parg <- par_init(data_Gratiot, fixed.parameters=NULL,
                 add.cofactors="Tide.meter")

likelihood_phenology(data=data_Gratiot, fitted.parameters = parg,
                     cofactors=td6, add.cofactors="Tide.meter")

result_Gratiot_CF <- fit_phenology(data=data_Gratiot,
                                   fitted.parameters=parg, fixed.parameters=NULL, cofactors=td6,
                                   add.cofactors="Tide.meter")

compare_AIC(WithoutCF=result_Gratiot, WithCF=result_Gratiot_CF)
plot(result_Gratiot_CF)

# Example with two series fitted with different peaks but same Length of season

Gratiot2 <- Gratiot
Gratiot2[, 2] <- floor(Gratiot2[, 2]*runif(n=nrow(Gratiot2)))
data_Gratiot <- add_phenology(Gratiot, name="Complete1",

```

```

                                reference=as.Date("2001-01-01"), format="%d/%m/%Y")
data_Gratiot <- add_phenology(Gratiot2, name="Complete2",
                                reference=as.Date("2001-01-01"),
                                format="%d/%m/%Y", previous=data_Gratiot)

pfixed=c(Min=0)
p <- par_init(data_Gratiot, fixed.parameters = pfixed)
p <- c(p, Peak_Complete1=175, Peak_Complete2=175)
p <- p[-4]
p <- c(p, Length=90)
p <- p[-(3:4)]
result_Gratiot <- fit_phenology(data=data_Gratiot, fitted.parameters=p,
                                fixed.parameters=pfixed)

# An example with bimodality

g <- Gratiot
g[30:60, 2] <- sample(10:20, 31, replace = TRUE)
data_g <- add_phenology(g, name="Complete", reference=as.Date("2001-01-01"),
                                format="%d/%m/%Y")
parg <- c('Max.1_Complete' = 5.6344636692341856,
          'MinB.1_Complete' = 0.15488810581002324,
          'MinE.1_Complete' = 0.2,
          'LengthB.1' = 22.366647176407636,
          'Peak.1' = 47.902473939250036,
          'LengthE.1' = 17.828495918533015,
          'Max.2_Complete' = 33.053364083447434,
          'MinE.2_Complete' = 0.42438173496989717,
          'LengthB.2' = 96.651564706802702,
          'Peak.2' = 175.3451874571835,
          'LengthE.2' = 62.481968743789835,
          'Theta' = 3.6423908093342572)
pfixed <- c('MinB.2_Complete' = 0,
          'Flat.1' = 0,
          'Flat.2' = 0)
result_g <- fit_phenology(data=data_g, fitted.parameters=parg, fixed.parameters=pfixed)
plot(result_g)

# Exemple with some minimum counts

nb <- Gratiot[, 2]
nbs <- sample(0:1, length(nb), replace=TRUE)
nb <- ifelse(nb != 0, ifelse(nbs == 1, 1, nb), 0)
nbc <- ifelse(nb != 0, ifelse(nbs == 1, "minimum", "exact"), "exact")

Gratiot_minimal <- cbind(Gratiot, CountTypes=nbc)
Gratiot_minimal[, 2] <- nb

data_Gratiot_minimal <- add_phenology(add=Gratiot_minimal,
                                colname.CountTypes = "CountTypes",
                                month_ref=1)
parg <- par_init(data_Gratiot_minimal, fixed.parameters=NULL)

result_Gratiot_minimal <- fit_phenology(data=data_Gratiot_minimal,

```

```

    fitted.parameters=parg, fixed.parameters=NULL)
plot(result_Gratiot_minimal)

summary(result_Gratiot_minimal)

# Exemple with all zero counts being not recorded

Gratiot_NoZeroCounts <- cbind(Gratiot[Gratiot[, 2] != 0, ], ZeroCounts=FALSE)
data_Gratiot_NoZeroCounts <- add_phenology(add=Gratiot_NoZeroCounts,
                                           colname.ZeroCounts = "ZeroCounts",
                                           ZeroCounts.default=FALSE,
                                           month_ref=1)

# or
data_Gratiot_NoZeroCounts <- add_phenology(add=Gratiot[Gratiot[, 2] != 0, ],
                                           ZeroCounts.default=FALSE,
                                           month_ref=1)

parg <- par_init(data_Gratiot_NoZeroCounts, fixed.parameters=NULL)

result_Gratiot_NoZeroCounts <- fit_phenology(data=data_Gratiot_NoZeroCounts,
                                             fitted.parameters=parg, fixed.parameters=NULL)
plot(result_Gratiot_NoZeroCounts)

summary(result_Gratiot_NoZeroCounts)

# Exemple with data in range of date
Gratiot_rangedate <- Gratiot
Gratiot_rangedate[, 1] <- as.character(Gratiot_rangedate[, 1])
Gratiot_rangedate[148, 1] <- paste0(Gratiot_rangedate[148, 1], "-", Gratiot_rangedate[157, 1])
Gratiot_rangedate[148, 2] <- sum(Gratiot_rangedate[148:157, 2])
Gratiot_rangedate <- Gratiot_rangedate[-(149:157), ]

data_Gratiot_rangedate <- add_phenology(add=Gratiot_rangedate,
                                         month_ref=1)
parg <- par_init(data_Gratiot_rangedate, fixed.parameters=NULL)

likelihood_phenology(data=data_Gratiot_rangedate,
                     fitted.parameters=parg)

result_Gratiot_rangedate <- fit_phenology(data=data_Gratiot_rangedate,
                                           fitted.parameters=parg,
                                           fixed.parameters=NULL)

plot(result_Gratiot_rangedate)

likelihood_phenology(result=result_Gratiot_rangedate)

# Exemple with data in range of date and CountTypes being minimum
Gratiot_rangedate <- Gratiot
Gratiot_rangedate[, 1] <- as.character(Gratiot_rangedate[, 1])
Gratiot_rangedate[148, 1] <- paste0(Gratiot_rangedate[148, 1], "-", Gratiot_rangedate[157, 1])
Gratiot_rangedate[148, 2] <- sum(Gratiot_rangedate[148:157, 2])
Gratiot_rangedate <- Gratiot_rangedate[-(149:157), ]

```

```

Gratiot_rangedate <- cbind(Gratiot_rangedate, CountTypes="exact")
Gratiot_rangedate[148, 2] <- 100
Gratiot_rangedate[148, "CountTypes"] <- "minimum"
Gratiot_rangedate[28, "CountTypes"] <- "minimum"

data_Gratiot_rangedate <- add_phenology(add=Gratiot_rangedate,
                                     colname.CountTypes="CountTypes",
                                     month_ref=1)
parg <- par_init(data_Gratiot_rangedate, fixed.parameters=NULL)

result_Gratiot_rangedate <- fit_phenology(data=data_Gratiot_rangedate,
    fitted.parameters=parg, fixed.parameters=NULL)

likelihood_phenology(result=result_Gratiot_rangedate)

plot(result_Gratiot_rangedate)

## End(Not run)

```

fixed.parameters0	<i>Generate a set of fixed parameters for series with only 0 counts</i>
-------------------	---

Description

This function generates a set of fixed parameters for series with only 0 counts.
The parameter series must be a result from `add_phenology()`.

Usage

```

fixed.parameters0(
  series = stop("A result from add_phenology() must be provided.")
)

```

Arguments

`series` Set of series generated with `add_phenology()`

Details

`fixed.parameters0` generates a set of fixed parameters for series with only 0 counts

Value

Return a set of parameters

Author(s)

Marc Girondot

Examples

```
## Not run:
refdate <- as.Date("2001-01-01")
data_Gratiot <- add_phenology(Gratiot, name="Complete",
  reference=refdate, format="%d/%m/%Y")
pfixed <- fixed.parameters0(data_Gratiot)

## End(Not run)
```

generateCF

Generate a set of data to test Clutch Frequency for marine turtles.

Description

This function generates a dataframe to test `fitCF()`.

This model is an enhanced version of the one published by Briane et al. (2007).

Parameters are `mu` and `sd` being the parameters of a distribution used to model the clutch frequency. This distribution is used only as a guide but has not statistical meaning.

The parameter `p` is the -logit probability that a female is seen on the beach for a particular nesting event. It includes both the probability that it is captured but also the probability that it uses that specific beach.

Several categories of females can be included in the model using index after the name of the parameter, for example `mu1`, `sd1` and `mu2`, `sd2` indicates that two categories of females with different clutch frequencies distribution are present. Similarly `p1` and `p2` indicates that two categories of females with different capture probabilities are present.

If more than one category is used, then it is necessary to include the parameter `OTN` to indicate the relative frequencies of each category. If two categories are used, one `OTN` parameter named `ONT1` must be included. The `ONT2` is forced to be 1. Then the relative frequency for category 1 is $ONT1/(ONT1+1)$ and for category 2 is $1/(ONT1+1)$. Same logic must be applied for 3 and more categories with always the last one being fixed to 1.

if `p` or `a` (logit of the capture probability) are equal to $-\infty$, the probability of capture is 0 and if they are equal to $+\infty$, the probability is 1.

The value of `p` out of the period of nesting must be set to $+\infty$ (capture probability=1) to indicate that no turtle is nesting in this period.

`p` must be set to $-\infty$ (capture probability=0) to indicate that no monitoring has been done during a specific period of the nesting season.

The best way to indicate capture probability for 3D model (OCF, ECF, Period) is to indicate `p.period` common for all categories and `a1`, `a2`, etc for each category. The capture probability for category 1 will be `p.period * a1`, and for category 2 will be `p.period * a2`, etc.

In this case, the parameters `p.period` should be indicated in fitted parameters as well as `a1`, but `a2` must be fixed to $+\infty$ in `fixed.parameters`. Then the capture probability for category 2 will be

p.period and for category 1 a1 * p.period.

Usage

```
generateCF(
  x = c(mu = 4, sd = 1, p = +Inf, mu_season = 13.8360591186578, sd_season =
    0.17440085345944),
  MeanDaysBetween2Nests = 9.8,
  date0 = as.Date("2020-01-01"),
  n = 1,
  verbose = TRUE
)
```

Arguments

x	Initial parameters to be used
MeanDaysBetween2Nests	Number of days in average between two nests
date0	Initial date to generate data
n	Number of individuals to model
verbose	If TRUE, give information about each animal.

Details

generateCF generates set of data to test fitCF.

Value

Return a list with 4 elements: Category, CF, Beginning and Observations being a dataframe of individuals.

Author(s)

Marc Girondot

See Also

Briane J-P, Rivalan P, Girondot M (2007) The inverse problem applied to the Observed Clutch Frequency of Leatherbacks from Yalimapo beach, French Guiana. *Chelonian Conservation and Biology* 6:63-69

Fossette S, Kelle L, Girondot M, Goverse E, Hilterman ML, Verhage B, Thoisy B, de, Georges J-Y (2008) The world's largest leatherback rookeries: A review of conservation-oriented research in French Guiana/Suriname and Gabon. *Journal of Experimental Marine Biology and Ecology* 356:69-82

Other Model of Clutch Frequency: [ECFOCF_f\(\)](#), [ECFOCF_full\(\)](#), [TableECFOCF\(\)](#), [fitCF\(\)](#), [fitCF_MHmcmc\(\)](#), [fitCF_MHmcmc_p\(\)](#), [lnLCF\(\)](#), [logLik.ECFOCF\(\)](#), [plot.ECFOCF\(\)](#), [plot.TableECFOCF\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example

par <- c(mu = 2.4911638591178051,
         sd = 0.96855483039640977,
         mu_season = 13.836059118657793,
         sd_season = 0.17440085345943984,
         p.10 = 1.3348233607728222,
         p.11 = 1.1960387774393837,
         p.12 = 0.63025680979544774,
         p.13 = 0.38648155002707452,
         p.14 = 0.31547864054366048,
         p.15 = 0.19720001827017075,
         p.16 = 0.083199496372073328,
         p.17 = 0.32969130595897905,
         p.18 = 0.36582777525265819,
         p.19 = 0.30301248314170637,
         p.20 = 0.69993987591518514,
         p.21 = 0.13642423871641118,
         p.22 = -1.3949268190534629,
         p=+Inf)

o_mu1p1season1 <- generateCF(x=par, n=1, verbose=TRUE)
o_mu1p1season1 <- generateCF(x=par, n=1000)
plot(o_mu1p1season1$CF)
hist(o_mu1p1season1$Beginning)

## End(Not run)
```

Gratiot

Leatherback nest counts from Gratiot et al. (2006) Figure 1

Description

Leatherback nest counts from Gratiot et al. (2006) Figure 1. These data have been collected by the ONG Kwata in French Guiana.

The data have been obtained from the graph of the publication (see reference).

Usage

```
Gratiot
```

Format

data.frame with the morning date in the first column and the nest counts on the second one.

Details

Leatherback nest counts from Gratiot et al. (2006) Figure 1

Author(s)

KWATA ONG

References

Gratiot, N., Gratiot, J., de Thoisy, B. & Kelle, L. 2006. Estimation of marine turtles nesting season from incomplete data ; statistical adjustment of a sinusoidal function. *Animal Conservation*, 9, 95-102.

See Also

Other Phenology model: `AutoFitPhenology()`, `BE_to_LBLE()`, `LBLE_to_BE()`, `LBLE_to_L()`, `L_to_LBLE()`, `MarineTurtles_2002`, `MinBMinE_to_Min()`, `adapt_parameters()`, `add_SE()`, `add_phenology()`, `extract_result()`, `fit_phenology()`, `likelihood_phenology()`, `logLik.phenology()`, `map_Gratiot`, `map_phenology()`, `par_init()`, `phenology()`, `phenology2fitRMU()`, `phenology_MHmcmc()`, `phenology_MHmcmc_p()`, `plot.phenology()`, `plot.phenologydata()`, `plot.phenologymap()`, `plot_delta()`, `plot_phi()`, `print.phenology()`, `print.phenologymap()`, `print.phenologyout()`, `remove_site()`, `result_Gratiot`, `result_Gratiot1`, `result_Gratiot2`, `result_Gratiot_Flat`, `summary.phenology()`, `summary.phenologymap()`, `summary.phenologyout()`

Examples

```
library(phenology)
# Read a file with data
data(Gratiot)
```

IPFit

Fit a model of Internesting Period for marine turtles.

Description

This function fits a model of internesting period using maximum likelihood or using Metropolis-Hastings algorithm with Bayesian model.

The fit using maximum likelihood is not the best strategy because the objective function is based on a stochastic model (and then a single set of parameters does not produce exactly the same output each time). The use of Metropolis-Hastings algorithm (a Markov chain Monte Carlo method) should be preferred.

Usage

```

IPFit(
  x = NULL,
  fixed.parameters = NULL,
  data = stop("Formatted data must be provided"),
  method = c("Nelder-Mead", "BFGS"),
  control = list(trace = 1, REPORT = 100, maxit = 500),
  itnmax = c(500, 100),
  hessian = TRUE,
  verbose = TRUE,
  parallel = TRUE,
  model = c("MH", "ML"),
  parametersMH,
  n.iter = 10000,
  n.chains = 1,
  n.adapt = 100,
  thin = 30,
  trace = TRUE,
  adaptive = TRUE,
  adaptive.lag = 500,
  adaptive.fun = function(x) {
    ifelse(x > 0.234, 1.3, 0.7)
  },
  intermediate = NULL,
  filename = "intermediate.Rdata"
)

```

Arguments

<code>x</code>	Initial parameters to be fitted
<code>fixed.parameters</code>	Parameters that are fixed.
<code>data</code>	Data as a vector
<code>method</code>	Method to be used by <code>optimx()</code>
<code>control</code>	List of controls for <code>optimx()</code>
<code>itnmax</code>	A vector with maximum iterations for each method.
<code>hessian</code>	Logical to estimate SE of parameters
<code>verbose</code>	If TRUE, show the parameters for each tested model
<code>parallel</code>	If TRUE, will use parallel computing
<code>model</code>	Can be ML for Maximum likelihood or MH for Metropolis Hastings
<code>parametersMH</code>	The priors. See <code>MHalgoGen</code>
<code>n.iter</code>	See <code>MHalgoGen</code>
<code>n.chains</code>	See <code>MHalgoGen</code>
<code>n.adapt</code>	See <code>MHalgoGen</code>

thin	See MHalgoGen
trace	See MHalgoGen
adaptive	See MHalgoGen
adaptive.lag	See MHalgoGen
adaptive.fun	See MHalgoGen
intermediate	See MHalgoGen
filename	See MHalgoGen

Details

IPFit fit a model of Interesting Period for marine turtles.

Value

Return a list of class IP with the fit informations and the fitted model.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Model of Interesting Period: [IPModel\(\)](#), [IPPredict\(\)](#), [plot.IP\(\)](#), [summary.IP\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
data <- structure(c(`0` = 0, `1` = 47, `2` = 15, `3` = 6, `4` = 5, `5` = 4,
  `6` = 2, `7` = 5, `8` = 57, `9` = 203, `10` = 205, `11` = 103,
  `12` = 35, `13` = 24, `14` = 12, `15` = 10, `16` = 13, `17` = 49,
  `18` = 86, `19` = 107, `20` = 111, `21` = 73, `22` = 47, `23` = 30,
  `24` = 19, `25` = 17, `26` = 33, `27` = 48, `28` = 77, `29` = 83,
  `30` = 65, `31` = 37, `32` = 27, `33` = 23, `34` = 24, `35` = 22,
  `36` = 41, `37` = 42, `38` = 44, `39` = 33, `40` = 39, `41` = 24,
  `42` = 18, `43` = 18, `44` = 22, `45` = 22, `46` = 19, `47` = 24,
  `48` = 28, `49` = 17, `50` = 18, `51` = 19, `52` = 17, `53` = 4,
  `54` = 12, `55` = 9, `56` = 6, `57` = 11, `58` = 7, `59` = 11,
  `60` = 12, `61` = 5, `62` = 4, `63` = 6, `64` = 11, `65` = 5,
  `66` = 6, `67` = 7, `68` = 3, `69` = 2, `70` = 1, `71` = 3, `72` = 2,
  `73` = 1, `74` = 2, `75` = 0, `76` = 0, `77` = 3, `78` = 1, `79` = 0,
  `80` = 2, `81` = 0, `82` = 0, `83` = 1), Year = "1994",
  Species = "Dermochelys coriacea",
  location = "Yalimapo beach, French Guiana",
  totalnumber = 2526L, class = "IP")
par(mar=c(4, 4, 1, 1)+0.4)
plot(data, xlim=c(0,100))
text(100, 190, labels=bquote(italic(.(attributes(data)$Species))), pos=2)
```

```

text(100, 150, labels=attributes(data)$location, pos=2, cex=0.8)
text(100, 110, labels=paste0(as.character(attributes(data)$totalnumber), " females"), pos=2)

##### Fit using Maximum-Likelihood

par <- c(meanIP = 9.8229005713237623,
        sdIP = 0.079176011861863474,
        minIP = 6.8128364577569309,
        pAbort = 1.5441529841959203,
        meanAbort = 2.7958742380756121,
        sdAbort = 0.99370406770777175,
        pCapture = -0.80294884905867658,
        meanECF = 4.5253772889275758,
        sdECF = 0.20334743335612529)

fML <- IPFit(x=par,
            fixed.parameters=c(N=20000),
            data=data,
            verbose=FALSE,
            model="ML")

# Plot the fitted ECF
plot(fML, result="ECF")

# Plot the Internesting Period distribution
plot(fML, result="IP")

# Plot the distribution of days between tentatives
plot(fML, result="Abort", xlim=c(0, 15))
#'
##### Fit using ML and non parametric ECF

par <- c(ECF.2 = 0.044151921569961131,
        ECF.3 = 2.0020778325280748,
        ECF.4 = 2.6128345101617083,
        ECF.5 = 2.6450582416622375,
        ECF.6 = 2.715198206774927,
        ECF.7 = 2.0288031327239904,
        ECF.8 = 1.0028041546528881,
        ECF.9 = 0.70977432157689235,
        ECF.10 = 0.086052204035003091,
        ECF.11 = 0.011400419961702518,
        ECF.12 = 0.001825219438794076,
        ECF.13 = 0.00029398731859899116,
        ECF.14 = 0.002784886479846703,
        meanIP = 9.9887100433529721,
        sdIP = 0.10580250625108811,
        minIP = 6.5159124624132048,
        pAbort = 2.5702251748938956,
        meanAbort = 2.2721679285648841,
        sdAbort = 0.52006431730489933,
        pCapture = 0.079471782729506113)

```

```

fML_NP <- IPFit(x=par,
               fixed.parameters=c(N=20000),
               data=data,
               verbose=FALSE,
               model="ML")

par <- fML_NP$ML$par

fML_NP <- IPFit(x=par,
               fixed.parameters=c(N=1000000),
               data=data,
               verbose=FALSE,
               model="ML")

par <- c(ECF.2 = 0.016195025683080871,
        ECF.3 = 2.0858089267994315,
        ECF.4 = 3.1307578727979348,
        ECF.5 = 2.7495760827322622,
        ECF.6 = 2.8770821670450939,
        ECF.7 = 2.1592708144943145,
        ECF.8 = 1.0016227335391867,
        ECF.9 = 0.80990178270345259,
        ECF.10 = 0.081051214954249967,
        ECF.11 = 0.039757901443389344,
        ECF.12 = 6.3324056808464527e-05,
        ECF.13 = 0.00037500864146146936,
        ECF.14 = 0.0010383506745475582,
        meanIP = 10.004121090603523,
        sdIP = 0.10229422354470977,
        minIP = 6.5051758088487883,
        pAbort = 2.5335985958484839,
        meanAbort = 2.3145895392189173,
        sdAbort = 0.51192514362374153,
        pCapture = 0.055440514236842105,
        DeltameanIP = -0.046478049165483697)

fML_NP_Delta <- IPFit(x=par,
                     fixed.parameters=c(N=20000),
                     data=data,
                     verbose=FALSE,
                     model="ML")

par <- fML_NP_Delta$ML$par

fML_NP_Delta <- IPFit(x=par,
                     fixed.parameters=c(N=1000000),
                     data=data,
                     verbose=FALSE,
                     model="ML")

# Test for stability of -Ln L value according to N
grandL.mean <- NULL
grandL.sd <- NULL

```

```

N <- c(10000, 20000, 30000, 40000, 50000,
      60000, 70000, 80000, 90000,
      100000, 200000, 300000, 400000, 500000,
      600000, 700000, 800000, 900000,
      1000000)
for (Ni in N) {
  print(Ni)
  smallL <- NULL
  for (replicate in 1:100) {
    smallL <- c(smallL,
      getFromNamespace(".IPInL", ns="phenology")
        (x=par, fixed.parameters=c(N=Ni), data=data))
  }
  grandL.mean <- c(grandL.mean, mean(smallL))
  grandL.sd <- c(grandL.sd, sd(smallL))
}

grandL.mean <- c(242.619750064524, 239.596145944548, 238.640010536147, 237.965573853263,
  237.727506424543, 237.240740566494, 237.527948232993, 237.297225856515,
  237.17073080938, 237.103397800143, 236.855939567838,
  236.704861853456, 236.82264801458, 236.606065021519, 236.685930841831,
  236.697562908131, 236.568003663293, 236.58097471402, 236.594282543024
)
grandL.sd <- c(6.54334049298099, 3.04916614991682, 2.57932397492509, 2.15990307710982,
  1.59826856034413, 1.54505295915354, 1.59734964880484, 1.41845032728396,
  1.43096821211286, 1.20048923027244, 0.912467350448495,
  0.75814052890774, 0.668841336554019, 0.539505594152166, 0.554662419326559,
  0.501551009304687, 0.415199780254872, 0.472274287714195, 0.386237047201706
)

plot_errbar(x=N, y=grandL.mean, errbar.y = 2*grandL.sd,
  xlab="N", ylab="-Ln L (2 SD)", bty="n", las=1)

# Plot the fitted ECF
plot(fML_NP_Delta, result="ECF")

# Plot the Internesting Period distribution
plot(fML_NP_Delta, result="IP")

# Plot the distribution of days between tentatives
plot(fML_NP_Delta, result="Abort", xlim=c(0, 15))

print(paste("Probability of capture", invlogit(-fML_NP_Delta$ML$par["pCapture"])))
# Confidence interval at 95%
print(paste(invlogit(-fML_NP_Delta$ML$par["pCapture"]-1.96*fML_NP_Delta$ML$SE["pCapture"]), "-",
  invlogit(-fML_NP_Delta$ML$par["pCapture"]+1.96*fML_NP_Delta$ML$SE["pCapture"])))

print(paste("Probability of abort", invlogit(-fML_NP_Delta$ML$par["pAbort"])))
# Confidence interval at 95%
print(paste(invlogit(-fML_NP_Delta$ML$par["pAbort"]-1.96*fML_NP_Delta$ML$SE["pAbort"]), "-",
  invlogit(-fML_NP_Delta$ML$par["pAbort"]+1.96*fML_NP_Delta$ML$SE["pAbort"])))

compare_AIC(parametric=fML$ML,

```

```

nonparameteric=fML_NP$ML,
nonparametricDelta=fML_NP_Delta$ML)

##### Fit using Metropolis-Hastings algorithm
# ECF.1 = 1 is fixed
par <- c(ECF.2 = 0.044151921569961131,
        ECF.3 = 2.0020778325280748,
        ECF.4 = 2.6128345101617083,
        ECF.5 = 2.6450582416622375,
        ECF.6 = 2.715198206774927,
        ECF.7 = 2.0288031327239904,
        ECF.8 = 1.0028041546528881,
        ECF.9 = 0.70977432157689235,
        ECF.10 = 0.086052204035003091,
        ECF.11 = 0.011400419961702518,
        ECF.12 = 0.001825219438794076,
        ECF.13 = 0.00029398731859899116,
        ECF.14 = 0.002784886479846703,
        meanIP = 9.9887100433529721,
        sdIP = 0.10580250625108811,
        minIP = 6.5159124624132048,
        pAbort = 2.5702251748938956,
        meanAbort = 2.2721679285648841,
        sdAbort = 0.52006431730489933,
        pCapture = 0.079471782729506113)

df <- data.frame(Density=rep("dunif", length(par)),
Prior1=c(rep(0, 13), 8, 0.001, 0, -8, 0, 0.001, -8),
Prior2=c(rep(10, 13), 12, 1, 10, 8, 2, 1, 8),
SDProp=unname(c(ECF.2 = 6.366805760909012e-05,
                ECF.3 = 6.366805760909012e-05,
                ECF.4 = 6.366805760909012e-05,
                ECF.5 = 6.366805760909012e-05,
                ECF.6 = 6.366805760909012e-05,
                ECF.7 = 6.366805760909012e-05,
                ECF.8 = 6.366805760909012e-05,
                ECF.9 = 6.366805760909012e-05,
                ECF.10 = 6.366805760909012e-05,
                ECF.11 = 6.366805760909012e-05,
                ECF.12 = 6.366805760909012e-05,
                ECF.13 = 6.366805760909012e-05,
                ECF.14 = 6.366805760909012e-05,
                meanIP = 6.366805760909012e-05,
                sdIP = 6.366805760909012e-05,
                minIP = 6.366805760909012e-05,
                pAbort = 6.366805760909012e-05,
                meanAbort = 6.366805760909012e-05,
                sdAbort = 6.366805760909012e-05,
                pCapture = 6.366805760909012e-05)),
Min=c(rep(0, 13), 8, 0.001, 0, -8, 0, 0.001, -8),
Max=c(rep(10, 13), 12, 1, 10, 8, 2, 1, 8),
Init=par, stringsAsFactors = FALSE)
rownames(df)<- names(par)

```

```
fMH <- IPFit(parametersMH=df,
fixed.parameters=c(N=10000),
data=data,
verbose=FALSE,
n.iter = 10000,
n.chains = 1, n.adapt = 100, thin = 1, trace = TRUE,
adaptive = TRUE,
model="MH")

# Plot the fitted ECF
plot(fMH, result="ECF")

## End(Not run)
```

IPModel

Estimates the pattern of interesting intervals for a set of parameters.

Description

This function fits a model of interesting period.
The parameters are:

- meanIP : The average number of days between two nesting processes
- DeltameanIP : The shift in days for IP at each new clutch.
- sdIP : The standard deviation of number of days between two nesting processes
- minIP : The minimum number of days between two nesting processes
- pAbort : The -logit of the probability to abort a nesting process
- meanAbort : The average of the number of days after the abortion of a nesting process
- sdAbort : The standard deviation of the number of days after the abortion of a nesting process
- pCapture : The -logit of the probability to capture a female on the beach
- meanECF : The average number of clutch a female will try to do being represented as ECF
- sdECF : The standard deviation of number of clutch a female will try to do
- N : The number of replicates to generate the distribution (default is 10000 if not indicated)
- ECF.x : The relative proportion of females nesting with ECF = x (ECF.1 being fixed to 1)

Usage

```
IPModel(
  par,
  parallel = TRUE,
  limits = list(meanIP = 40, meanECF = 4, minIP = 15, sdAbort = 1, sdIP = 1, sdECF = 1,
    DeltameanIP = 0.5, maxDays = 365)
)
```

Arguments

<code>par</code>	Set of parameters
<code>parallel</code>	If TRUE, will use parallel computing
<code>limits</code>	A list of limits for various parameters

Details

IPModel estimates the pattern of interesting intervals for a set of parameters.

Value

Return a list with two elements.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Model of Interesting Period: [IPFit\(\)](#), [IPPredict\(\)](#), [plot.IP\(\)](#), [summary.IP\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
par <- c(meanIP = 9.8,
sdIP = 0.1,
minIP = 7,

pAbort = -logit(0.1),
meanAbort = 2,
sdAbort = 0.05,

pCapture = -logit(0.8),

meanECF = 4,
sdECF = 0.1)

model <- IPModel(c(par, N=10000))

plot(model)

## End(Not run)
```

IPPredict	<i>Predict the possible clutch number based on observed Internesting Period.</i>
-----------	--

Description

This function predicts the possible clutch number based on observed Internesting Period.

Usage

```
IPPredict(x = NULL, par = NULL, N = NULL, IP = 0:100)
```

Arguments

x	A result of IPFit().
par	A set of parameters.
N	Number of replicates
IP	A vector of Internesting Period

Details

IPPredict calculates the possible clutch number based on observed Internesting Period.

Value

A data.frame

Author(s)

Marc Girondot

See Also

Other Model of Internesting Period: [IPFit\(\)](#), [IPModel\(\)](#), [plot.IP\(\)](#), [summary.IP\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
##### Fit using Maximum-Likelihood

par <- c(meanIP = 9.9959691992722917,
        sdIP = 0.10066664270893474,
        minIP = 7.5684588178888754,
        pAbort = 2.2510012544630911,
        meanAbort = 2.8969185085603386,
```

```

sdAbort = 0.92688983853803242,
pCapture = -1.0393803705929086,
meanECF = 3.9551519427394255,
sdECF = 0.31657679943365019)

IPPredict(par=par, IP=c(10, 80))

## End(Not run)

```

LBLE_to_BE

Transform a set of parameters from LengthB LengthE to Begin End.

Description

This function is used to transform a set of parameters that uses LengthB, Peak and LengthE to a set of parameters that uses Begin, Peak and End.

Usage

```
LBLE_to_BE(parameters = NULL, help = FALSE)
```

Arguments

parameters	Set of current parameters
help	If TRUE, an help is displayed

Details

LBLE_to_BE transforms a set of parameters from LengthB LengthE to Begin End.

Value

Return a set of modified parameters

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
refdate <- as.Date("2001-01-01")
data_Gratiot<-add_phenology(Gratiot, name="Complete", reference=refdate, format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Change the parameters to Begin End format
parg1<-LBLE_to_BE(parameters=parg)
# And change back to LengthB LengthE
parg2<-BE_to_LBLE(parameters=parg1)
```

LBLE_to_L	<i>Transform a set of parameters from LengthB LengthE format to Length</i>
-----------	--

Description

This function is used to transform a set of parameters that uses LengthB and LengthE to a set of parameters uses Length.

Usage

```
LBLE_to_L(parameters = stop("Set of parameters must be given"))
```

Arguments

parameters Set of current parameters

Details

LBLE_to_L transforms a set of parameters from LengthB LengthE format to Length.

Value

Return the set of modified parameters

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#),

```
print.phenology(), print.phenologymap(), print.phenologyout(), remove_site(), result_Gratiot,
result_Gratiot1, result_Gratiot2, result_Gratiot_Flat, summary.phenology(), summary.phenologymap(),
summary.phenologyout())
```

Examples

```
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
refdate <- as.Date("2001-01-01")
data_Gratiot<-add_phenology(Gratiot, name="Complete", reference=refdate, format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Change the parameters to Begin End format
parg1<-LBLE_to_L(parameters=parg)
# And change back to LengthB LengthE.
parg2<-L_to_LBLE(parameters=parg1)
```

likelihood_phenology *Estimate the likelihood of timeseries based on a set of parameters.*

Description

This function is used to estimate the likelihood based on a set of parameters.

Usage

```
likelihood_phenology(
  data = NULL,
  fitted.parameters = NULL,
  fixed.parameters = NULL,
  parallel = TRUE,
  result = NULL,
  model_before = NULL,
  method_Snbinom = NULL,
  cofactors = NULL,
  add.cofactors = NULL,
  zero = 1e-09,
  out = TRUE
)
```

Arguments

data Dataset generated with add_format.

fitted.parameters Set of parameters to be fitted.

fixed.parameters Set of fixed parameters.

parallel	If TRUE, parallel computing is used.
result	An object obtained after fit_phenology()
model_before	The change of parameters before to estimate daily counts.
method_Snbinom	Can be Furman, exact, or saddlepoint.
cofactors	data.frame with a column Date and a column for each cofactor.
add.cofactors	Names of the column of parameter cofactors to use as a cofactor.
zero	If the theoretical nest number is under this value, this value will be used.
out	If TRUE, return the global likelihood; if FALSE, the likelihood for each series.

Details

likelihood_phenology estimate likelihood for a set of parameters.

Value

The likelihood of the data with the parameters

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
## Not run:
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot <- add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg <- par_init(data_Gratiot, fixed.parameters=NULL)
# Estimate likelihood with this initial set of parameters
likelihood_phenology(data=data_Gratiot, fitted.parameters=parg, fixed.parameters=NULL)
# Or directly from a result object
likelihood_phenology(result=result_Gratiot)
# With new parametrization based on Omeyer et al. (2022)
# Omeyer, L. C. M., McKinley, T. J., Br  heret, N., Bal, G., Balchin, G. P., Bitsindou, A.,
# Chauvet, E., Collins, T., Curran, B. K., Formia, A., Girard, A., Girondot, M., Godley, B. J.,
```

```
# Mavoungou, J.-G., Poli, L., Tilley, D., VanLeeuwe, H. & Metcalfe, K. 2022. Missing data in
# sea turtle population monitoring: a Bayesian statistical framework accounting for incomplete
# sampling Front. Mar. Sci. (IF 3.661), 9, 817014.

parg <- c(tp=unnname(parg["Peak"]), tf=unnname(parg["Flat"]),
          s1=unnname(parg["LengthB"])/4.8, s2=unnname(parg["LengthE"])/4.8,
          alpha=unnname(parg["Max_Complete"]), Theta=unnname(parg["Theta"]))
likelihood_phenology(data=data_Gratiot, fitted.parameters=parg, fixed.parameters=NULL)

## End(Not run)
```

lnLCF

*Calculate the -log likelihood of data within a model.***Description**

Calculate the -log likelihood of data within a model.

Usage

```
lnLCF(
  x,
  data,
  fixed.parameters = NULL,
  parallel = TRUE,
  verbose = FALSE,
  attributeWAIC = TRUE
)
```

Arguments

<code>x</code>	A named vector of parameters (mu, sd, mu_season, sd_season, a, p and OTN).
<code>data</code>	CMR database formatted using TableECFOCF().
<code>fixed.parameters</code>	Parameters that are fixed.
<code>parallel</code>	If TRUE, parallel computing in ECFOCF_f is used.
<code>verbose</code>	if TRUE, show the parameters.
<code>attributeWAIC</code>	TRUE if WAIC must be returned.

Details

lnLCF calculate the -log likelihood of data within a model.

Value

Return the -log likelihood of data within a model.

Author(s)

Marc Girondot

See Also

Other Model of Clutch Frequency: [ECFOCF_f\(\)](#), [ECFOCF_full\(\)](#), [TableECFOCF\(\)](#), [fitCF\(\)](#), [fitCF_MHmcmc\(\)](#), [fitCF_MHmcmc_p\(\)](#), [generateCF\(\)](#), [logLik.ECFOCF\(\)](#), [plot.ECFOCF\(\)](#), [plot.TableECFOCF\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
ECFOCF_2002 <- TableECFOCF(MarineTurtles_2002)
lnLCF(x=c(mu=4.71768454279272,
          sd=1.075711951667,
          p=-1.79746277312909),
      data=ECFOCF_2002)

ECFOCF_2002 <- TableECFOCF(MarineTurtles_2002, date0=as.Date("2002-01-01"))
fp <- rep(0, dim(ECFOCF_2002)[3])
names(fp) <- paste0("p.", formatC(1:(dim(ECFOCF_2002)[3]), width=2, flag="0"))
par <- c(mu1 = 0.6404831115214353,
        sd1 = 0.69362774786433479,
        mu2 = 5.6404831115214353,
        sd2 = 5.69362774786433479,
        mu_season = 12.6404831115214353,
        sd_season = 1.69362774786433479,
        OTN=1)
par <- c(par, fp[attributes(ECFOCF_2002)$table["begin"]:attributes(ECFOCF_2002)$table["final"]])
fixed.parameters <- c(p=-Inf)

lnLCF(x=par, data=ECFOCF_2002, fixed.parameters=fixed.parameters)

## End(Not run)
```

LnRI_norm

*Return a remigration interval.***Description**

Model of remigration interval

The vector of parameters must include:

sx, survival for year x

if s is included, all years have the same survival

tx, Tag retention for year x

rx, probability of return for year x

cx, probability of return for year x
 px, probability of observation for year x

Usage

```
LnRI_norm(data, x, k1 = NULL)
```

Arguments

data	Data with remigration intervals
x	Vector of parameters
k1	Maximum number of years for remigration intervals.

Details

lnRI_norm returns a ln L

Value

Return a remigration interval.

Author(s)

Marc Girondot

See Also

Other Model of Remigration Interval: [Bayesian.remigration\(\)](#), [RI\(\)](#), [RI2BP\(\)](#), [plot.Remigration\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
# Each year a fraction of 0.9 is surviving
s <- c(s1=0.9, s2=0.9, s3=0.9, s4=0.9, s5=0.9)
# Probability of tag retention; 0.95 the first year then after no loss
t <- c(t1=0.95, t2=1, t3=1, t4=1, t5=1)
# Time-conditional return probability - This is the true remigration rate
r <- c(r1=0.1, r2=0.8, r3=0.7, r4=0.7, r5=1)
# Capture probability
p <- c(p1=0.6, p2=0.6, p3=0.6, p4=0.6, p5=0.6)
# Number of observations for 400 tagged females after 1, 2, 3, 4, and 5 years
OBS <- c(400, 10, 120, 40, 20, 10)
# Likelihood of the observed number based on the model
LnRI_norm(data=OBS, x = c(s, t, r, p, sd=2) )
LnRI_norm(data=OBS, x = c(s=0.97, t, r, p, sd=2) )
```

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

logLik.ECFOCF	<i>Return Log Likelihood of a fit done using fitCF</i>
---------------	--

Description

Return Log Likelihood of a fit generated by fitCF.

Usage

```
## S3 method for class 'ECFOCF'
logLik(object, ...)
```

Arguments

object	A result file generated by fitCF
...	Not used

Details

logLik.ECFOCF return Log Likelihood of a fit done using fitCF

Value

The Log Likelihood value for the fitted model with data

Author(s)

Marc Girondot

See Also

Other Model of Clutch Frequency: [ECFOCF_f\(\)](#), [ECFOCF_full\(\)](#), [TableECFOCF\(\)](#), [fitCF\(\)](#), [fitCF_MHmcmc\(\)](#), [fitCF_MHmcmc_p\(\)](#), [generateCF\(\)](#), [lnLCF\(\)](#), [plot.ECFOCF\(\)](#), [plot.TableECFOCF\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
ECFOCF_2002 <- TableECFOCF(MarineTurtles_2002, date0=as.Date("2002-01-01"))
par <- c(mu = 2.6404831115214353,
        size = 0.69362774786433479,
        mu_season = 12.6404831115214353,
        size_season = 1.69362774786433479,
        a2=0)
fp <- rep(0, dim(ECFOCF_2002)[3])
names(fp) <- paste0("p.", formatC(1:(dim(ECFOCF_2002)[3]), width=2, flag="0"))
```

```

par <- c(par, fp[attributes(ECFOCF_2002)$table["begin"]:attributes(ECFOCF_2002)$table["final"]])
fixed.parameters <- c(a1=Inf, p=-Inf)

lnLCF(x=par, data=ECFOCF_2002, fixed.parameters=fixed.parameters)

o_mu1season1a2p <- fitCF(x=par, fixed.parameters=fixed.parameters,
                        data=ECFOCF_2002)

logLik(o_mu1season1a2p)
AIC(o_mu1season1a2p)

## End(Not run)

```

logLik.fitRMU

Return Log Likelihood of a fit generated by fitRMU

Description

Return Log Likelihood of a fit generated by fitRMU

Usage

```

## S3 method for class 'fitRMU'
logLik(object, ...)

```

Arguments

object	A result file generated by fitRMU
...	Not used

Details

logLik.fitRMU Return Log Likelihood of a fit for fitRMU

Value

The Log Likelihood value for the fitted model with data

Author(s)

Marc Girondot

See Also

Other Fill gaps in RMU: [CI.RMU\(\)](#), [fitRMU\(\)](#), [fitRMU_MHmcmc\(\)](#), [fitRMU_MHmcmc_p\(\)](#), [plot.fitRMU\(\)](#)

Examples

```
## Not run:
library(phenology)
RMU.name.AtlanticW <- data.frame(mean=c("Yalimapo.French.Guiana",
                                         "Galibi.Suriname",
                                         "Irakumpapy.French.Guiana"),
                                  se=c("se_Yalimapo.French.Guiana",
                                        "se_Galibi.Suriname",
                                        "se_Irakumpapy.French.Guiana"))

data.AtlanticW <- data.frame(Year=c(1990:2000),
                              Yalimapo.French.Guiana=c(2076, 2765, 2890, 2678, NA,
                                                         6542, 5678, 1243, NA, 1566, 1566),
                              se_Yalimapo.French.Guiana=c(123.2, 27.7, 62.5, 126, NA,
                                                         230, 129, 167, NA, 145, 20),
                              Galibi.Suriname=c(276, 275, 290, NA, 267,
                                                  542, 678, NA, 243, 156, 123),
                              se_Galibi.Suriname=c(22.3, 34.2, 23.2, NA, 23.2,
                                                    4.3, 2.3, NA, 10.3, 10.1, 8.9),
                              Irakumpapy.French.Guiana=c(1076, 1765, 1390, 1678, NA,
                                                         3542, 2678, 243, NA, 566, 566),
                              se_Irakumpapy.French.Guiana=c(23.2, 29.7, 22.5, 226, NA,
                                                            130, 29, 67, NA, 15, 20))

cst <- fitRMU(data=data.AtlanticW, RMU.name=RMU.name.AtlanticW,
              colname.year="Year", model.trend="Constant",
              model.SD="Zero")

logLik(cst)
AIC(cst)

## End(Not run)
```

logLik.phenology

*Return Log Likelihood of a fit generated by fit_phenology***Description**

Return Log Likelihood of a fit generated by fit_phenology

Usage

```
## S3 method for class 'phenology'
logLik(object, ...)
```

Arguments

object	A result file generated by fit_phenology
...	Not used

Details

logLik.phenology Return Log Likelihood of a fit

Value

The Log Likelihood value of the fitted model and data

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
## Not run:
library(phenology)
data(result_Gratiot)
logLik(result_Gratiot)
AIC(result_Gratiot)

## End(Not run)
```

logLik.Tagloss	<i>Return Log Likelihood of a fit generated by Tagloss_fit</i>
----------------	--

Description

Return Log Likelihood of a fit generated by Tagloss_fit

Usage

```
## S3 method for class 'Tagloss'
logLik(object, ...)
```

Arguments

object	A result file generated by Tagloss_fit
...	Not used

Details

logLik.Tagloss returns Log Likelihood of a fit for tag loss

Value

The Log Likelihood value for the fitted model with data

Author(s)

Marc Girondot

See Also

Other Model of Tag-loss: [Tagloss_L\(\)](#), [Tagloss_LengthObs\(\)](#), [Tagloss_cumul\(\)](#), [Tagloss_daymax\(\)](#), [Tagloss_fit\(\)](#), [Tagloss_format\(\)](#), [Tagloss_mcmc\(\)](#), [Tagloss_mcmc_p\(\)](#), [Tagloss_model\(\)](#), [Tagloss_simulate\(\)](#), [o_4p_p1p2](#), [plot.Tagloss\(\)](#), [plot.TaglossData\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
data_f_21 <- Tagloss_format(outLR, model="21")
# Without the N20 the computing is much faster
data_f_21_fast <- subset(data_f_21, subset=(is.na(data_f_21$N20)))
par <- structure(c(48.8292784204825, 1039.02842229274, -89.3162940697861,
5.21817463244988, 8.00575451188548, 8.32971268127933, 161.265553603601,
602.935748681661, 2643.57415102633, 16.752815732218, 10.181616195839,
7.14279063312016), names = c("D1_2", "D2D1_2", "D3D2_2", "A_2",
"B_2", "C_2", "D1_1", "D2D1_1", "D3D2_1", "A_1", "B_1", "C_1"))
o <- Tagloss_fit(data=data_f_21_fast, fitted.parameters=par)
logLik(o)
AIC(o)

## End(Not run)
```

L_to_LBLE

Transform a set of parameters from Length format to LengthB LengthE

Description

This function is used to transform a set of parameters that uses Length to a set of parameters uses LengthB and LengthE.

Usage

```
L_to_LBLE(parameters = stop("Set of parameters must be given"))
```

Arguments

parameters Set of current parameters

Details

L_to_LBLE transforms a set of parameters from Length format to LengthB LengthE.

Value

Return the set of modified parameters

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
refdate <- as.Date("2001-01-01")
data_Gratiot<-add_phenology(Gratiot, name="Complete",
  reference=refdate, format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Change the parameters to Begin End format
parg1<-LBLE_to_L(parameters=parg)
# And change back to LengthB LengthE.
parg2<-L_to_LBLE(parameters=parg1)
```

map_Gratiot

Likelihood map of Leatherback nest counts

Description

Likelihood map of Leatherback nest counts from Gratiot et al. (2006) Figure 1. A intraseasonal periodic pattern was searched for varying Phi and Delta parameters.

Usage

```
map_Gratiot
```

Format

A list with Gratiot data and the result of the fit.

Details

Likelihood map of Leatherback nest counts from Gratiot et al. (2006) Figure 1

Author(s)

Marc Girondot <marc.girondot@u-psud.fr>

References

Gratiot, N., Gratiot, J., de Thoisy, B. & Kelle, L. 2006. Estimation of marine turtles nesting season from incomplete data ; statistical adjustment of a sinusoidal function. *Animal Conservation*, 9, 95-102.

See Also

Other Phenology model: `AutoFitPhenology()`, `BE_to_LBLE()`, `Gratiot`, `LBLE_to_BE()`, `LBLE_to_L()`, `L_to_LBLE()`, `MarineTurtles_2002`, `MinBMinE_to_Min()`, `adapt_parameters()`, `add_SE()`, `add_phenology()`, `extract_result()`, `fit_phenology()`, `likelihood_phenology()`, `logLik.phenology()`, `map_phenology()`, `par_init()`, `phenology()`, `phenology2fitRMU()`, `phenology_MHmcmc()`, `phenology_MHmcmc_p()`, `plot.phenology()`, `plot.phenologydata()`, `plot.phenologymap()`, `plot_delta()`, `plot_phi()`, `print.phenology()`, `print.phenologymap()`, `print.phenologyout()`, `remove_site()`, `result_Gratiot`, `result_Gratiot1`, `result_Gratiot2`, `result_Gratiot_Flat`, `summary.phenology()`, `summary.phenologymap()`, `summary.phenologyout()`

Examples

```
library(phenology)
# Read a file with likelihood map
data(map_Gratiot)
```

Description

This function generates a map of likelihood varying Phi and Delta.

Parameters are the same than for the `fit_phenology()` function except for trace that is disabled.

If Alpha, Beta or Tau are not indicated, Alpha and Tau are set to 0 and 1 and Beta is fitted.

Only one set of Alpha, Beta, Tau, Phi and Delta are used for all timeseries present in data.

Note that it is possible to fit or fixed Alpha[n], Beta[n], Tau[n], Phi[n] and Delta[n] with [n]=1, 2 or 3 and then it is possible to use this function to establish the likelihood map for a second or third sinusoids added to the global pattern.

If Delta is not specified, it is estimated from Phi and the same precision as Phi is used.

Usage

```
map_phenology(
  data = NULL,
  fitted.parameters = NULL,
  fixed.parameters = NA,
  Phi = seq(from = 0.2, to = 20, length.out = 100),
  Delta = NULL,
  progressbar = any(installed.packages()[, "Package"] == "pbapply"),
  cofactors = NULL,
  add.cofactors = NULL,
  zero = 1e-09,
  mc.cores = getOption("mc.cores", detectCores()),
  forking = getOption("forking", ifelse(.Platform$OS.type == "windows", FALSE, TRUE))
)
```

Arguments

<code>data</code>	dataset generated with <code>add_format</code>
<code>fitted.parameters</code>	Set of parameters to be fitted
<code>fixed.parameters</code>	Set of fixed parameters
<code>Phi</code>	Phi values to be analyzed
<code>Delta</code>	Delta value to be analyzed
<code>progressbar</code>	If FALSE, do not show the progress bar
<code>cofactors</code>	data.frame with a column Date and a column for each cofactor
<code>add.cofactors</code>	Names of the column of parameter cofactors to use as a cofactor
<code>zero</code>	If the theoretical nest number is under this value, this value will be used
<code>mc.cores</code>	Number of cores to use for parallel computing
<code>forking</code>	Should forking be used for parallel computing

Details

`map_phenology` generates a likelihood map.

Value

Display a likelihood map

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Phenology model: `AutoFitPhenology()`, `BE_to_LBLE()`, `Gratiot`, `LBLE_to_BE()`, `LBLE_to_L()`, `L_to_LBLE()`, `MarineTurtles_2002`, `MinBMinE_to_Min()`, `adapt_parameters()`, `add_SE()`, `add_phenology()`, `extract_result()`, `fit_phenology()`, `likelihood_phenology()`, `logLik.phenology()`, `map_Gratiot`, `par_init()`, `phenology()`, `phenology2fitRMU()`, `phenology_MHmcmc()`, `phenology_MHmcmc_p()`, `plot.phenology()`, `plot.phenologydata()`, `plot.phenologymap()`, `plot_delta()`, `plot_phi()`, `print.phenology()`, `print.phenologymap()`, `print.phenologyout()`, `remove_site()`, `result_Gratiot`, `result_Gratiot1`, `result_Gratiot2`, `result_Gratiot_Flat`, `summary.phenology()`, `summary.phenologymap()`, `summary.phenologyout()`

Examples

```
library("phenology")
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot <- add_phenology(Gratiot, name = "Complete",
  reference = as.Date("2001-01-01"), format = "%d/%m/%Y")
# Generate initial points for the optimisation
parg <- par_init(data_Gratiot, fixed.parameters = NULL)
# Run the optimisation
## Not run:
result_Gratiot <- fit_phenology(data = data_Gratiot,
  fitted.parameters = parg, fixed.parameters = NULL)

## End(Not run)
data(result_Gratiot)
# Extract the fitted parameters
parg1 <- extract_result(result_Gratiot)
# Add constant Alpha and Tau values
# [day d amplitude=(Alpha+Nd*Beta)^Tau with Nd being the number of counts for day d]
pfixed <- c(parg1, Alpha=0, Tau=1)
pfixed <- pfixed[-which(names(pfixed)=="Theta")]
# The only fitted parameter will be Beta
parg2 <- c(Beta=0.5, parg1["Theta"])
# Generate a likelihood map
# [default Phi=seq(from=0.1, to=20, length.out=100) but it is very long]
# Take care, it takes 20 hours ! The data map_Gratiot has the result
## Not run:
library(phenology)
map_Gratiot <- map_phenology(data = data_Gratiot,
  Phi = seq(from=0.1, to=30, length.out=100),
  fitted.parameters = parg2,
```

```

fixed.parameters = pfixed)
# Rapid example for debug
map_Gratiot_short <- map_phenology(data = data_Gratiot,
                                   Phi = seq(from=10, to=30, length.out=2),
                                   fitted.parameters = parg2,
                                   fixed.parameters = pfixed, forking = FALSE)

## End(Not run)
data(map_Gratiot)
# Plot the map
plot(map_Gratiot, col = heat.colors(128))
# Plot the min(-Ln L) for Phi varying at any delta value
plot_phi(map = map_Gratiot)
# Plot the min(-Ln L) for Delta varying with Phi equal to the value for maximum likelihood
plot_delta(map = map_Gratiot)
# Plot the min(-Ln L) for Delta varying with Phi the nearest to 15
plot_delta(map = map_Gratiot, Phi = 15)

```

MarineTurtles_2002 *Database of tagged marine turtles in 2002*

Description

Extraction of 2002 PIT tagged marine turtles database

Usage

MarineTurtles_2002

Format

data.frame with 2 columns:

- Date: The date the female has been seen on the beach (morning date of the night)
- ID: The unique identifier of the female

Details

Database of tagged marine turtles in 2002

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
library(phenology)
data(MarineTurtles_2002)
```

MinBMinE_to_Min	<i>Transform a set of parameters from MinB and MinE to Min</i>
-----------------	--

Description

This function is used to transform a set of parameters that uses MinB and MinE to a set of parameters that uses Min.

Usage

```
MinBMinE_to_Min(parameters = stop("A set of parameters must be indicated"))
```

Arguments

parameters Set of current parameters

Details

MinBMinE_to_Min transforms a set of parameters from MinB and MinE to Min

Value

Return a set of modified parameters

Author(s)

Marc Girondot

See Also

Other Phenology model: `AutoFitPhenology()`, `BE_to_LBLE()`, `Gratiot`, `LBLE_to_BE()`, `LBLE_to_L()`, `L_to_LBLE()`, `MarineTurtles_2002`, `adapt_parameters()`, `add_SE()`, `add_phenology()`, `extract_result()`, `fit_phenology()`, `likelihood_phenology()`, `logLik.phenology()`, `map_Gratiot`, `map_phenology()`, `par_init()`, `phenology()`, `phenology2fitRMU()`, `phenology_MHmcmc()`, `phenology_MHmcmc_p()`, `plot.phenology()`, `plot.phenologydata()`, `plot.phenologymap()`, `plot_delta()`, `plot_phi()`, `print.phenology()`, `print.phenologymap()`, `print.phenologyout()`, `remove_site()`, `result_Gratiot`, `result_Gratiot1`, `result_Gratiot2`, `result_Gratiot_Flat`, `summary.phenology()`, `summary.phenologymap()`, `summary.phenologyout()`

Examples

```
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
refdate <- as.Date("2001-01-01")
data_Gratiot<-add_phenology(Gratiot, name="Complete", reference=refdate, format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot)
# Change the parameters to PMinB and PMinE
parg1<-MinBMinE_to_Min(parameters=parg)
```

outLR

*Database of leatherback CMR in French Guiana***Description**

Database of leatherback CMR in French Guiana

Usage

```
outLR
```

Format

A dataframe with database of leatherback CMR in French Guiana

Details

Database of leatherback CMR in French Guiana

Author(s)

Marc Girondot <marc.girondot@u-psud.fr>

References

Rivalan, P., Godfrey, M.H., Prévot-Julliard, A.-C., Girondot, M., 2005. Maximum likelihood estimates of tag loss in leatherback sea turtles. *Journal of Wildlife Management* 69, 540-548.

Examples

```
## Not run:
library(phenology)
# Read a file with result
data(outLR)
data_f_21 <- Tagloss_format(outLR, model="21")

## End(Not run)
```

o_4p_p1p2

*Model of tagloss based on Rivalan data***Description**

Model of tagloss based on Rivalan data

Usage

```
o_4p_p1p2
```

Details

Model of tagloss based on Rivalan data

Author(s)

Marc Girondot

References

Rivalan, P., Godfrey, M.H., Prévot-Julliard, A.-C., Girondot, M., 2005. Maximum likelihood estimates of tag loss in leatherback sea turtles. *Journal of Wildlife Management* 69, 540-548.

See Also

Other Model of Tag-loss: [Tagloss_L\(\)](#), [Tagloss_LengthObs\(\)](#), [Tagloss_cumul\(\)](#), [Tagloss_daymax\(\)](#), [Tagloss_fit\(\)](#), [Tagloss_format\(\)](#), [Tagloss_mcmc\(\)](#), [Tagloss_mcmc_p\(\)](#), [Tagloss_model\(\)](#), [Tagloss_simulate\(\)](#), [logLik.Tagloss\(\)](#), [plot.Tagloss\(\)](#), [plot.TaglossData\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
data_f_21 <- Tagloss_format(outLR, model="21")

# Same data fitted with new model
par <- c(D1_1 = 100.15324837975547, A_1 = 5.9576927964120188,
        B_1 = 8.769924225871069, B_2 = 8.2353860179664125)
```

```

pfixed <- c(D2D1_1 = 2568, D3D2_1 = 2568, D2D1_2 = 2568, D3D2_2 = 2568)
o_4p_p1p2 <- Tagloss_fit(data=data_f_21, fitted.parameters = par,
                        fixed.parameters = pfixed,
                        model_before = "par['C_1']=par['B_1'];
                        par['A_2']=par['A_1'];
                        par['C_2']=par['B_2'];
                        par['D1_2']=par['D1_1']", hessian=TRUE)

data(o_4p_p1p2)
plot(o_4p_p1p2, model="1", col="red")
plot(o_4p_p1p2, model="2", col="blue", add=TRUE)
legend("topright", legend=c("2->1", "1->0"), lty=1, col=c("blue", "red"))
plot(o_4p_p1p2, model="Cumul")

## End(Not run)

```

Parameter_Global_Year *Transform a set of parameters from Year to global effect, or reverse*

Description

This function is used to transform a set of parameters that uses Peak, LengthB, LengthE, B, E, or Length to the same parameter with Year effect, or reverse.

The parameter series can be or a result from add_phenology() or from fit_phenology() or simply a vector of names.

If this is a vector of names, it is checked to remove _ characters.

Usage

```

Parameter_Global_Year(
  parameters = stop("A set of parameters must be indicated"),
  parname = c("Peak", "LengthB", "LengthE", "B", "E", "Length"),
  series = NULL,
  sep_year = "-",
  perYear = TRUE
)

```

Arguments

parameters	Set of current parameters
parname	Name of parameter to transform
series	Set of series (see description)
sep_year	Character used to separate the year
perYear	TRUE if year-specific values must be setup

Details

Parameter_Global_Year transforms a set of parameters from Year to global effect, or reverse

Value

Return a set of modified parameters

Author(s)

Marc Girondot <marc.girondot@gmail.com>

Examples

```
## Not run:
Parameter_Global_Year(parameters=c("Peak_Beach1-2018"=151, "Peak_Beach1-2019"=161),
                        parname="Peak", perYear=FALSE)
Parameter_Global_Year(parameters=c("Peak"=151),
                        series = c("beach1", "beach2"),
                        parname="Peak", perYear=TRUE)

## End(Not run)
```

par_init

Calculate initial set of parameters.

Description

This function is used to generate an initial set of parameters for fitting that is expected to be not too far from the final.

The parameters can be:

- Min, MinE, MinB, PMin, PMinB, PMinE;
- Max;
- Begin, Peak, Flat, End;
- Length, LengthB, LengthE;
- Theta;
- Alpha, Beta, Tau, Phi, Delta;
- Alpha1, Beta1, Tau1, Phi1, Delta1;
- Alpha2, Beta2, Tau2, Phi2, Delta2;
- Alpha3, Beta3, Tau3, Phi3, Delta3;

And the name of level if a cofactor is used.

The parameters Max, Min, MinE, MinB, Length, LengthB, LengthE, and Peak can be followed with _ and part of the name of the rookery.

The model for scale effect of sinusoid is: $\text{Alpha} + \text{Beta} * n(t) ^ \text{Tau}$ where $n(t)$ is the expected number for the day t without the sinusoid effect.

Usage

```
par_init(
  data = stop("A dataset must be provided"),
  fixed.parameters = NULL,
  add.cofactors = NULL
)
```

Arguments

`data` Dataset generated with `add_phenology()`

`fixed.parameters` Set of fixed parameters

`add.cofactors` Names of cofactors that will be used (see `fit_phenology()`)

Details

`par_init` calculates initial set of parameters.

Value

The initial set of parameters

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
## Not run:
library(phenology)
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot <- add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg <- par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
result_Gratiot <- fit_phenology(data=data_Gratiot,
```

```

    fitted.parameters=parg, fixed.parameters=NULL)
data(result_Gratiot)
# Plot the phenology and get some stats
output<-plot(result_Gratiot)

## When a series has only 0, it should be used in two steps
## Let see an example

# Let create a times series with only 0
data0 <- data.frame(Date=c("11/3/2015", "12/3/2015", "13/3/2015-18/3/2015", "25/3/2015"),
                    Number=c(0, 0, 0, 0),
                    Beach=rep("Site0", 4), stringsAsFactors=FALSE)
data1 <- data.frame(Date=c("15/3/2015", "16/3/2015", "20/3/2015-22/3/2015", "25/3/2015"),
                    Number=c(1, 0, 3, 0),
                    Beach=rep("Site1", 4), stringsAsFactors=FALSE)
data <- rbind(data0, data1)

# Here I include timeseries with no observation
try1 <- add_phenology(data, format="%d/%m/%Y", month_ref=1, include0=TRUE)
pfixed <- c(Min=0, Flat=0)
parg <- par_init(try1, fixed.parameters=pfixed)
# The Max value for the series without observations should not be fitted. The ML is for Max being 0
pfixed <- c(pfixed, parg[(substr(names(parg), 1, 4)=="Max_") & (parg == 0)])
parg <- parg[!(names(parg) %in% names(pfixed))]

## End(Not run)

```

phenology

Run a shiny application for basic functions of phenology package

Description

Run a shiny application for basic functions of phenology package. Thanks to Adriana Cortés Gómez and Joana Hancock for their help with translation.

Usage

```
phenology()
```

Details

phenology runs a shiny application for basic functions of phenology package

Value

Nothing

Author(s)

Marc Girondot

See Also

Other Phenology model: `AutoFitPhenology()`, `BE_to_LBLE()`, `Gratiot`, `LBLE_to_BE()`, `LBLE_to_L()`, `L_to_LBLE()`, `MarineTurtles_2002`, `MinBMinE_to_Min()`, `adapt_parameters()`, `add_SE()`, `add_phenology()`, `extract_result()`, `fit_phenology()`, `likelihood_phenology()`, `logLik.phenology()`, `map_Gratiot`, `map_phenology()`, `par_init()`, `phenology2fitRMU()`, `phenology_MHmcmc()`, `phenology_MHmcmc_p()`, `plot.phenology()`, `plot.phenologydata()`, `plot.phenologymap()`, `plot_delta()`, `plot_phi()`, `print.phenology()`, `print.phenologymap()`, `print.phenologyout()`, `remove_site()`, `result_Gratiot`, `result_Gratiot1`, `result_Gratiot2`, `result_Gratiot_Flat`, `summary.phenology()`, `summary.phenologymap()`, `summary.phenologyout()`

Examples

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

## End(Not run)
```

<code>phenology2fitRMU</code>	<i>Create the data to be used with <code>fitRMU()</code> for a summary(<code>phenology</code>).</i>
-------------------------------	---

Description

This function takes the result of `plot.phenology()` and generates the information to be used with `fitRMU()`.

The value of density can be `dnorm` or `dgamma`. `dnorm` is better if ML results are used and `dgamma` is for MCMC.

Here are some example of regular expressions (regex) in `grep`:

If format of timeseries is beachname-2005:

```
rookeries.names.grep="(.)-." +
```

```
years.grep="." + "(.)$" +
```

Examples:

```
gsub("(.)-.", "\\1", "beachname-2005"); gsub("." + "(.)$", "\\1", "beachname-2005")
```

If format of timeseries is beachname-2005-2006:

```
rookeries.names.grep="(.)-.-." +
```

```
years.grep="." + "([0-9][0-9][0-9][0-9])-." +
```

Examples:

```
gsub("(.)-.-.", "\\1", "beachname-2005-2006"); gsub("." + "([0-9][0-9][0-9][0-9])-." +, "\\1", "beachname-2005-2006")
```

If format of timeseries is beachname-20052006:

```
rookeries.names.grep="(.)-." +
```

```
years.grep="." + "([0-9][0-9][0-9][0-9])([0-9][0-9][0-9][0-9])$" +
```

Examples:

```
gsub("(.)-.", "\\1", "beachname-20052006"); gsub("." + "([0-9][0-9][0-9][0-9])([0-9][0-9][0-9][0-9])$", "\\1", "beachname-20052006")
```

The return is a list with these elements:

`RMU.data`, `years.byrow`, `colname.year`, and `RMU.names`.

If density is a vector, the density used is linked to the rank of the timeseries in `phenologyout`.

Usage

```
phenology2fitRMU(
  phenologyout = stop("A result obtained from summary(phenology)"),
  col.mean = "with_obs_Mean_ML",
  col.var = "with_obs_Var_ML",
  rookeries.names.grep = "(.+)-.+",
  years.grep = ".+-(.+)$",
  limit.cv = +Inf,
  limit.sd = +Inf,
  density = "dgamma"
)
```

Arguments

phenologyout	A result of plot.phenology() or summary()
col.mean	Name of the column to be used as mean value. Can be a vector.
col.var	Name of the column to be used as variance value. Can be a vector.
rookeries.names.grep	The pattern to return the rookery name from names of timeseries.
years.grep	The pattern to return the year from names of timeseries.
limit.cv	Remove data with higher coefficient of variation than the limit.
limit.sd	Remove data with higher standard deviation than the limit.
density	What density should be used. Can be dnorm or dgamma. Can be a vector.

Details

phenology2fitRMU is used to prepare output of phenology to be used by fitRMU()

Value

Return a list with elements ready to be used with fitRMU().

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
## Not run:
library("phenology")

## End(Not run)
```

phenology_MHmcmc

Run the Metropolis-Hastings algorithm for data

Description

Run the Metropolis-Hastings algorithm for data.

The number of iterations is `n.iter+n.adapt+1` because the initial likelihood is also displayed.

I recommend `thin=10`.

If initial point is maximum likelihood, `n.adapt = 0` is a good solution.

The parameters `intermediate` and `filename` are used to save intermediate results every 'intermediate' iterations (for example 1000). Results are saved in a file of name `filename`.

The parameter `previous` is used to indicate the list that has been save using the parameters `intermediate` and `filename`. It permits to continue a mcmc search.

These options are used to prevent the consequences of computer crash or if the run is very very long and computer processes are time limited.

Usage

```
phenology_MHmcmc(
  result = stop("An output from fit_phenology() must be provided"),
  n.iter = 10000,
  parametersMCMC = stop("A model generated with phenology_MHmcmc_p() must be provided"),
  n.chains = 1,
  n.adapt = 1000,
  thin = 1,
  WAIC = FALSE,
  WAIC.bybeach = TRUE,
  trace = FALSE,
  traceML = FALSE,
  adaptive = TRUE,
  adaptive.lag = 500,
  adaptive.fun = function(x) {
    ifelse(x > 0.234, 1.3, 0.7)
  },
  intermediate = NULL,
  filename = "intermediate.Rdata",
  previous = NULL
)
```

Arguments

result	An object obtained after a SearchR fit
n.iter	Number of iterations for each step
parametersMCMC	A set of parameters used as initial point for searching with information on priors
n.chains	Number of replicates
n.adapt	Number of iterations before to store outputs
thin	Number of iterations between each stored output
WAIC	Should WAIC information be saved?
WAIC.bybeach	Should the WAIC be saved by beach or by count?
trace	TRUE or FALSE or period, shows progress
traceML	TRUE or FALSE to show ML
adaptive	Should an adaptive process for SDProp be used
adaptive.lag	Lag to analyze the SDProp value in an adaptive content
adaptive.fun	Function used to change the SDProp
intermediate	Period for saving intermediate result, NULL for no save
filename	If intermediate is not NULL, save intermediate result in this file
previous	Previous result to be continued. Can be the filename in which intermediate results are saved.

Details

phenology_MHmcmc runs the Metropolis-Hastings algorithm for data (Bayesian MCMC)

Value

A list with resultMCMC being mcmc.list object, resultLnL being likelihoods and parametersMCMC being the parameters used

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
## Not run:
library(phenology)
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot <- add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg <- par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
result_Gratiot <- fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=NULL)
# Generate set of priors for Bayesian analysis
pmcmc <- phenology_MHmcmc_p(result_Gratiot, accept = TRUE)
# Here no WAIC data
result_Gratiot_mcmc <- phenology_MHmcmc(result = result_Gratiot, n.iter = 10000,
  parametersMCMC = pmcmc, n.chains = 1,
  n.adapt = 0, thin = 10, trace = FALSE,
  WAIC=FALSE)

# Here the WAIC is at the level of the beaches
result_Gratiot_mcmc <- phenology_MHmcmc(result = result_Gratiot, n.iter = 10000,
  parametersMCMC = pmcmc, n.chains = 1,
  n.adapt = 0, thin = 10, trace = FALSE,
  WAIC=TRUE, WAIC.bybeach=TRUE)

# Here the WAIC is at the level of the daily counts
result_Gratiot_mcmc <- phenology_MHmcmc(result = result_Gratiot, n.iter = 10000,
  parametersMCMC = pmcmc, n.chains = 1,
  n.adapt = 0, thin = 10, trace = FALSE,
  WAIC=TRUE, WAIC.bybeach=FALSE)

loo::loo(result_Gratiot_mcmc$WAIC)
loo::waic(result_Gratiot_mcmc$WAIC)
# You can use loo_compare() using a named list with these objects
# of ELPDweight(loos = list(m1=loo::loo(model1$WAIC), m2=loo::loo(model2$WAIC))

# Get standard error of parameters
summary(result_Gratiot_mcmc)

# Make diagnostics of the mcmc results using coda package
mcmc <- as.mcmc(result_Gratiot_mcmc)
require(coda)
heidel.diag(mcmc)
raftery.diag(mcmc)
autocorr.diag(mcmc)
acf(mcmc[[1]][,"LengthB"], lag.max=200, bty="n", las=1)
acf(mcmc[[1]][,"Max_Complete"], lag.max=50, bty="n", las=1)
batchSE(mcmc, batchSize=100)
# The batch standard error procedure is usually thought to
# be not as accurate as the time series methods used in summary
summary(mcmc)$statistics[, "Time-series SE"]
plot(result_Gratiot_mcmc, parameters="Max_Complete", las=1, xlim=c(-10, 210))
plot(result_Gratiot_mcmc, what="MarkovChain", ylim=c(20, 40))
plot(result_Gratiot_mcmc, what="LnL")
```

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

phenology_MHmcmc_p	<i>Generates set of parameters to be used with phenology_MHmcmc()</i>
--------------------	---

Description

Interactive script used to generate set of parameters to be used with phenology_MHmcmc().

Usage

```
phenology_MHmcmc_p(
  result = stop("An output from fit_phenology() must be provided"),
  default.density = "dunif",
  accept = FALSE
)
```

Arguments

result	An object obtained after a fit_phenology() fit
default.density	The default density, "dnorm" or "dunif"
accept	If TRUE, does not wait for use interaction

Details

phenology_MHmcmc_p generates set of parameters to be used with phenology_MHmcmc()

Value

A matrix with the parameters

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
## Not run:
library(phenology)
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot<-add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
result_Gratiot<-fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=NULL)
# Generate set of priors for Bayesian analysis
pmcmc <- phenology_MHmcmc_p(result_Gratiot, accept = TRUE)
result_Gratiot_mcmc <- phenology_MHmcmc(result = result_Gratiot, n.iter = 10000,
  parametersMCMC = pmcmc, n.chains = 1,
  n.adapt = 0, thin = 10, trace = FALSE,
  WAIC=FALSE)

# Get standard error of parameters
summary(result_Gratiot_mcmc)
# Make diagnostics of the mcmc results using coda package
mcmc <- as.mcmc(result_Gratiot_mcmc)
require(coda)
heidel.diag(mcmc)
raftery.diag(mcmc)
autocorr.diag(mcmc)
acf(mcmc[[1]][,"LengthB"], lag.max=200, bty="n", las=1)
acf(mcmc[[1]][,"Max_Gratiot"], lag.max=50, bty="n", las=1)
batchSE(mcmc, batchSize=100)
# The batch standard error procedure is usually thought to
# be not as accurate as the time series methods used in summary
summary(mcmc)$statistics[, "Time-series SE"]
plot(result_Gratiot_mcmc, parameters=3, las=1, xlim=c(-10, 300))

## End(Not run)
```

plot.ECFOCF

Plot a result of clutch frequency fit.

Description

This function plots the result of fitCF().
 The result data plots the observed ECF-OCF table.
 The result dataOCF plots the observed OCF table.
 The result dataECF plots the observed ECF table.
 The result CF plots the true clutch frequency.
 The result OCF plots the observed clutch frequency.
 The result ECF plots the estimated clutch frequency.
 The result ECFOCF plots the bivariate observed vs. estimated clutch frequency.

The result ECFOCF0 plots the bivariate observed vs. estimated clutch frequency without the 0 OCF.
 The result probabilities plots the probabilities of captures, p and a, or OTN.
 The result period plots the probabilities of nesting according to period.
 If category is left to NA, the compound value for all the population is plotted.
 When result="data" is used, this is a parser for plot.TableECFOCF().
 See this function for the parameters.
 The parameter y.axis is the shift of the x legends for result="prob".
 When a resultMCMC is indicated, if replicates is "all", all values from all chains are used; if a value lower than number of iterations is indicated, a regular thinning is used and if a value larger then number if iteration is indicated, a sampling with replacement is used.

Usage

```
## S3 method for class 'ECFOCF'
plot(
  x,
  ...,
  result = "CF",
  category = NA,
  parameters = "p1",
  legend = parameters,
  period = 1,
  resultMCMC = NULL,
  chain = "all",
  replicates = "all"
)
```

Arguments

x	A result for fitCF().
...	Graphic parameters, see plot.TableECFOCF() or par.
result	What result will be plotted: data, dataOCF, dataECF, ECF, OCF, ECFOCF, ECFOCF0, CF, Probabilities, period
category	What category will be plotted, numeric or NA for all.
parameters	Name of parameters to plot for Probabilities result.
legend	Legend to use for the parameters as a vector of names.
period	The period that will be plotted.
resultMCMC	A result from fitRMU_MHmcmc.
chain	Which chain to be used in resultMCMC.
replicates	How many replicates from resultMCMC.

Details

plot.ECFOCF plots a result of clutch frequency fit.

Value

Nothing

Author(s)

Marc Girondot

See Also

Other Model of Clutch Frequency: [ECFOCF_f\(\)](#), [ECFOCF_full\(\)](#), [TableECFOCF\(\)](#), [fitCF\(\)](#), [fitCF_MHmcmc\(\)](#), [fitCF_MHmcmc_p\(\)](#), [generateCF\(\)](#), [lnLCF\(\)](#), [logLik.ECFOCF\(\)](#), [plot.TableECFOCF\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
data(MarineTurtles_2002)
ECFOCF_2002 <- TableECFOCF(MarineTurtles_2002)
o_mu1p2_NB <- fitCF(x = c(mu = 4.6426989650675701,
                          sd = 75.828239144717074,
                          p1 = 0.62036295627161053,
                          p2 = -2.3923021862881511,
                          OTN = 0.33107456308054345),
                    data=ECFOCF_2002)

par(mar=c(4, 4, 1, 1)+0.4)
plot(o_mu1p2_NB, result="data", category=NA,
     bty="n", las=1, cex.points=3, cex.axis = 0.8)
plot(o_mu1p2_NB, result="data", category=NA,
     bty="n", las=1, cex.points=3, pch=NA,
     col.labels = "red", show.labels=TRUE, cex.0=0.2,
     show.0 = TRUE, col.0="blue", pch.0=4)
plot(o_mu1p2_NB, result="dataOCF", category=NA,
     bty="n", las=1)
plot(o_mu1p2_NB, result="dataECF", category=NA,
     bty="n", las=1)

plot(o_mu1p2_NB, result="CF", bty="n", las=1)

plot(o_mu1p2_NB, result="OCF", category=1, bty="n", las=1)
plot(o_mu1p2_NB, result="OCF", category=2, bty="n", las=1)

plot(o_mu1p2_NB, result="ECFOCF", bty="n", las=1)

plot(o_mu1p2_NB, result="ECFOCF0", bty="n", las=1)
plot(o_mu1p2_NB, result="ECFOCF0", category=1, bty="n", las=1)
plot(o_mu1p2_NB, result="ECFOCF0", category=2, bty="n", las=1)

plot(o_mu1p2_NB, result="Prob", category=c(1, 2), bty="n", las=1)
plot(o_mu1p2_NB, result="Prob", category=c(2, 1), bty="n", las=1)
```

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

plot.fitRMU	<i>Plot the synthesis of RMU fit.</i>
-------------	---------------------------------------

Description

The function plot.fitRMU plots the results of fitRMU().

In most of the cases, replicate.CI can be set to 0 for what="proportions" or "numbers".

The parameter CI.RMU can be used when this function is used several times with the same data.

Usage

```
## S3 method for class 'fitRMU'
plot(
  x,
  ...,
  resultMCMC = NULL,
  chain = 1,
  replicate.CI = 10000,
  CI.RMU = NULL,
  what = "proportions",
  criteria = "50%",
  aggregate = "both",
  order = NULL,
  control.legend = list(),
  show.legend = TRUE,
  col = rainbow,
  border = NA,
  names.legend = NULL
)
```

Arguments

x	A result file generated by fitRMU
...	Parameters used by plot
resultMCMC	MCMC result for fitRUM
chain	Chain to be plotted for MCMC
replicate.CI	Number of replicates to estimate CI
CI.RMU	A result of CI.RMU()
what	Can be proportions, numbers or total
criteria	What criteria will be used for proportions or numbers: mean or 50%
aggregate	Can be model or both

order	Give the order of series in plot, from bottom to top. Can be used to not show series.
control.legend	Parameters send to legend
show.legend	If FALSE, does not show legend
col	The function used to generate colors.
border	The border of polygons used to represent the proportions.
names.legend	Names to show in legend.

Details

plot.fitRMU plots the results of a fit RMU.

Value

Return A list with result of CI.RMU()

Author(s)

Marc Girondot

See Also

Other Fill gaps in RMU: [CI.RMU\(\)](#), [fitRMU\(\)](#), [fitRMU_MHmcmc\(\)](#), [fitRMU_MHmcmc_p\(\)](#), [logLik.fitRMU\(\)](#)

Examples

```
## Not run:
library("phenology")
RMU.names.AtlanticW <- data.frame(mean=c("Yalimapo.French.Guiana",
                                         "Galibi.Suriname",
                                         "Irakumpapy.French.Guiana"),
                                  se=c("se_Yalimapo.French.Guiana",
                                         "se_Galibi.Suriname",
                                         "se_Irakumpapy.French.Guiana"), stringsAsFactors = FALSE)
data.AtlanticW <- data.frame(Year=c(1990:2000),
                              Yalimapo.French.Guiana=c(2076, 2765, 2890, 2678, NA,
                                                         6542, 5678, 1243, NA, 1566, 1566),
                              se_Yalimapo.French.Guiana=c(123.2, 27.7, 62.5, 126, NA,
                                                         230, 129, 167, NA, 145, 20),
                              Galibi.Suriname=c(276, 275, 290, NA, 267,
                                                  542, 678, NA, 243, 156, 123),
                              se_Galibi.Suriname=c(22.3, 34.2, 23.2, NA, 23.2,
                                                  4.3, 2.3, NA, 10.3, 10.1, 8.9),
                              Irakumpapy.French.Guiana=c(1076, 1765, 1390, 1678, NA,
                                                         3542, 2678, 243, NA, 566, 566),
                              se_Irakumpapy.French.Guiana=c(23.2, 29.7, 22.5, 226, NA,
                                                            130, 29, 67, NA, 15, 20), stringsAsFactors = FALSE)

cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
              colname.year="Year", model.trend="Constant",
```

```

        model.SD="Zero")
expo <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
              colname.year="Year", model.trend="Exponential",
              model.SD="Zero")
YS <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
             colname.year="Year", model.trend="Year-specific",
             model.SD="Zero")
YS1 <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
              colname.year="Year", model.trend="Year-specific",
              model.SD="Zero", model.rookeries="First-order")
YS1_cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
                  colname.year="Year", model.trend="Year-specific",
                  model.SD="Constant", model.rookeries="First-order",
                  parameters=YS1$par)
YS2 <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
              colname.year="Year", model.trend="Year-specific",
              model.SD="Zero", model.rookeries="Second-order",
              parameters=YS1$par)
YS2_cst <- fitRMU(RMU.data=data.AtlanticW, RMU.names=RMU.names.AtlanticW,
                  colname.year="Year", model.trend="Year-specific",
                  model.SD="Constant", model.rookeries="Second-order",
                  parameters=YS1_cst$par)

compare_AIC(Constant=cst, Exponential=expo,
            YearSpecific=YS)

compare_AIC(YearSpecific_ProportionsFirstOrder_Zero=YS1,
            YearSpecific_ProportionsFirstOrder_Constant=YS1_cst)

compare_AIC(YearSpecific_ProportionsConstant=YS,
            YearSpecific_ProportionsFirstOrder=YS1,
            YearSpecific_ProportionsSecondOrder=YS2)

compare_AIC(YearSpecific_ProportionsFirstOrder=YS1_cst,
            YearSpecific_ProportionsSecondOrder=YS2_cst)

barplot_errbar(YS1_cst$proportions[1, ], y.plus = YS1_cst$proportions.CI.0.95[1, ],
               y.minus = YS1_cst$proportions.CI.0.05[1, ], las=1, ylim=c(0, 0.7),
               main="Proportion of the different rookeries in the region")

plot(cst, main="Use of different beaches along the time", what="total")
plot(expo, main="Use of different beaches along the time", what="total")
plot(YS2_cst, main="Use of different beaches along the time", what="total")

plot(YS1, main="Use of different beaches along the time")
plot(YS1_cst, main="Use of different beaches along the time")
plot(YS1_cst, main="Use of different beaches along the time", what="numbers")

parpre <- par(mar=c(4, 4, 2, 5)+0.4)
par(xpd=TRUE)
plot(YS, main="Use of different beaches along the time",
     control.legend=list(x=2000, y=0.4, legend=c("Yalimapo", "Galibi", "Irakumpapy")))
par(mar=parpre)

```

```
# Example to modify order of series
plot(cst, order=c("Galibi.Suriname", "Irakumpapy.French.Guiana"))
plot(cst, order=c("Galibi.Suriname", "Irakumpapy.French.Guiana", "Yalimapo.French.Guiana"))

# Example to change the color
plot(cst, order=c("Galibi.Suriname", "Irakumpapy.French.Guiana", "Yalimapo.French.Guiana"),
     col=function(n) rep(c("gray", "lightgrey"), floor(n/2)), border="black",
     names.legend=c("Yalimapo", "Galibi", "Irakumpapy"))

## End(Not run)
```

plot.IP

Plot a result of Interesting Period fit or data.

Description

This function plots the result of IPFit() or IPModel().

If col is defined with a number of colors, only these colors and shown in legend.

Usage

```
## S3 method for class 'IP'
plot(x, ..., N = NULL, clutch = 1, result = "data")
```

Arguments

x	A result for IPFit() or IPModel().
...	Graphic parameters, see par().
N	Number of replicates for IPModel().
clutch	The rank of clutch when DeltameanIP is used.
result	What result will be plotted: data, model, data&model, IP, Abort, ECF, reverseECF

Details

plot.IP plots a result of Interesting Period fit or data

Value

Nothing

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Model of Interesting Period: [IPFit\(\)](#), [IPModel\(\)](#), [IPPredict\(\)](#), [summary.IP\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
data <- c(0, 47, 15, 6, 5, 4, 2, 5, 57, 203, 205, 103, 35, 24, 12, 10,
  13, 49, 86, 107, 111, 73, 47, 30, 19, 17, 33, 48, 77, 83, 65,
  37, 27, 23, 24, 22, 41, 42, 44, 33, 39, 24, 18, 18, 22, 22, 19,
  24, 28, 17, 18, 19, 17, 4, 12, 9, 6, 11, 7, 11, 12, 5, 4, 6,
  11, 5, 6, 7, 3, 2, 1, 3, 2, 1, 2, 0, 0, 3, 1, 0, 2, 0, 0, 1)
class(data) <- unique(append("IP", class(data)))
plot(data)

##### Fit parametric ECF using Maximum-Likelihood

par <- c(meanIP = 9.9959691992722917,
  sdIP = 0.10066664270893474,
  minIP = 7.5684588178888754,
  pAbort = 2.2510012544630911,
  meanAbort = 2.8969185085603386,
  sdAbort = 0.92688983853803242,
  pCapture = -1.0393803705929086,
  meanECF = 3.9551519427394255,
  sdECF = 0.31657679943365019)

fML <- IPFit(x=par,
  fixed.parameters=c(N=1000000),
  data=data,
  verbose=FALSE,
  model="ML")

# Plot the fitted ECF
plot(fML, result="ECF")

# Plot the Interesting Period distribution
plot(fML, result="IP")

# Plot the distribution of days between tentatives
plot(fML, result="Abort")
plot(fML, result="Abort", xlim=c(0, 10))

# Plot the data
plot(fML, result="data")

# Plot the data and the model
plot(fML, result="data&model")

# Plot the cumulative proportion of ECF according to date of observation
plot(fML, result="reverseECF")
```

```
plot(fML_NP_Delta, result="reverseECF", col=grey.colors(128))
```

```
##### Fit using Metropolis-Hastings
```

```
# ECF.1 = 1 is fixed
```

```
par <- c(ECF.2 = 0.044151921569961131,
        ECF.3 = 2.0020778325280748,
        ECF.4 = 2.6128345101617083,
        ECF.5 = 2.6450582416622375,
        ECF.6 = 2.715198206774927,
        ECF.7 = 2.0288031327239904,
        ECF.8 = 1.0028041546528881,
        ECF.9 = 0.70977432157689235,
        ECF.10 = 0.086052204035003091,
        ECF.11 = 0.011400419961702518,
        ECF.12 = 0.001825219438794076,
        ECF.13 = 0.00029398731859899116,
        ECF.14 = 0.002784886479846703,
        meanIP = 9.9887100433529721,
        sdIP = 0.10580250625108811,
        minIP = 6.5159124624132048,
        pAbort = 2.5702251748938956,
        meanAbort = 2.2721679285648841,
        sdAbort = 0.52006431730489933,
        pCapture = 0.079471782729506113)
```

```
df <- data.frame(Density=rep("dunif", length(par)),
Prior1=c(rep(0, 13), 8, 0.001, 0, -8, 0, 0.001, -8),
Prior2=c(rep(10, 13), 12, 1, 10, 8, 2, 1, 8),
SDProp=unname(c(ECF.2 = 6.366805760909012e-05,
                ECF.3 = 6.366805760909012e-05,
                ECF.4 = 6.366805760909012e-05,
                ECF.5 = 6.366805760909012e-05,
                ECF.6 = 6.366805760909012e-05,
                ECF.7 = 6.366805760909012e-05,
                ECF.8 = 6.366805760909012e-05,
                ECF.9 = 6.366805760909012e-05,
                ECF.10 = 6.366805760909012e-05,
                ECF.11 = 6.366805760909012e-05,
                ECF.12 = 6.366805760909012e-05,
                ECF.13 = 6.366805760909012e-05,
                ECF.14 = 6.366805760909012e-05,
                meanIP = 6.366805760909012e-05,
                sdIP = 6.366805760909012e-05,
                minIP = 6.366805760909012e-05,
                pAbort = 6.366805760909012e-05,
                meanAbort = 6.366805760909012e-05,
                sdAbort = 6.366805760909012e-05,
                pCapture = 6.366805760909012e-05))),
Min=c(rep(0, 13), 8, 0.001, 0, -8, 0, 0.001, -8),
Max=c(rep(10, 13), 12, 1, 10, 8, 2, 1, 8),
Init=par, stringsAsFactors = FALSE)
rownames(df)<- names(par)
```

```
fMH <- IPFit(parametersMH=df,
fixed.parameters=c(N=10000),
data=data,
verbose=FALSE,
n.iter = 10000,
n.chains = 1, n.adapt = 100, thin = 1, trace = TRUE,
adaptive = TRUE,
model="MH")

# Plot the fitted ECF
plot(fMH, result="ECF")

# Plot the posteriors and priors
plot(fMH$MH, parameters="meanIP", xlim=c(6, 14))

plot(x=1:length(fMH$MH$resultLnL[[1]]), y=fMH$MH$resultLnL[[1]],
type="l", xlab="Iterations", ylab="Ln L", bty="n", las=1)

## End(Not run)
```

plot.phenology	<i>Plot the phenology from a result.</i>
----------------	--

Description

The function plot.phenology plots the phenology graph from a result object.

If cofactors have been added, the plot does not show their effects.

plot.objects can be "observations", "ML" for maximum likelihood, "ML.SD" or "MCMC.SD" for dispersion of observations, "ML.quantiles" or "MCMC.quantiles" if a mcmc object is given

Usage

```
## S3 method for class 'phenology'
plot(
  x,
  ...,
  series = "all",
  moon = FALSE,
  replicate.CI = 10000,
  resultmcmc = NULL,
  season = NULL,
  chain = 1,
  replicate.CI.mcmc = "all",
  level = 0.95,
  plot.objects = c("observations", "ML", "ML.SD", "MCMC.SD", "ML.quantiles",
    "MCMC.quantiles"),
  col.ML = "black",
  col.SD = "red",
```

```

col.SD.polygon = rgb(red = 1, green = 0, blue = 0, alpha = 0.2),
col.MCMC.quantiles = "purple",
col.MCMC.quantiles.polygon = rgb(red = 160/255, green = 32/255, blue = 240/255, alpha =
  0.2),
col.ML.quantiles = "black",
col.ML.quantiles.polygon = rgb(red = 0, green = 0, blue = 0, alpha = 0.2),
col.observations = "black",
col.minimum.observations = "blue",
col.grouped.observations = "green"
)

```

Arguments

<code>x</code>	A result file generated by <code>fit_phenology</code>
<code>...</code>	Parameters used by <code>plot</code>
<code>series</code>	Name or number of series to be plotted or 'all'
<code>moon</code>	If TRUE, the moon phase is plotted. Default is FALSE
<code>replicate.CI</code>	Number of replicates for estimation of confidence interval
<code>resultmcmc</code>	A mcmc object
<code>season</code>	Which season to plot
<code>chain</code>	The number of chain to be used in <code>resultmcmc</code>
<code>replicate.CI.mcmc</code>	Number of iterations to be used or "all"
<code>level</code>	Level to estimate confidence interval or credibility interval
<code>plot.objects</code>	What to plot? See description
<code>col.ML</code>	Color of the ML mean curve
<code>col.SD</code>	Color of the SD curve (distribution of observations)
<code>col.SD.polygon</code>	Color of the polygon of the SD curve. If FALSE not shown.
<code>col.MCMC.quantiles</code>	Color of the quantiles curve based on mcmc
<code>col.MCMC.quantiles.polygon</code>	Color of the credibility interval polygon based on MCMC. If FALSE not shown.
<code>col.ML.quantiles</code>	Color of the SE curve based on ML
<code>col.ML.quantiles.polygon</code>	Color of the confidence interval polygon based on ML. If FALSE not shown.
<code>col.observations</code>	Color of the points
<code>col.minimum.observations</code>	Color of the points indicating minimum counts
<code>col.grouped.observations</code>	Color of the lines indicating grouped observations

Details

plot.phenology plots the phenology.

Value

A list with four objects: synthesis is a data.frame with global estimate of nesting.
details_MCMC, details_ML and details_mean are lists with day by day information for each series.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
## Not run:
library(phenology)
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot <- add_phenology(Gratiot, name = "Complete",
  reference = as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg <- par_init(data_Gratiot, fixed.parameters=NULL)
parg <- c('Max_Complete' = 33.076044848500167, 25,
  'MinB_Complete' = 0.21758630798131923,
  'MinE_Complete' = 0.42493953463205936,
  'LengthB' = 96.158007568020523,
  'Peak' = 174.62435300274245,
  'LengthE' = 62.084876419654634,
  'Flat' = 0,
  'Theta' = 3.5864650991821954)
# Run the optimisation
result_Gratiot <- fit_phenology(data=data_Gratiot,
  fitted.parameters=parg,
  fixed.parameters=NULL)

data(result_Gratiot)
# Plot the phenology and get some stats
output <- plot(result_Gratiot)
# Plot only part of the nesting season
ptoutput <- plot(result_Gratiot, xlim=c(as.Date("2001-03-01"),as.Date("2001-08-31")))
```

```

# Use month names in English
Sys.setlocale(category = "LC_TIME", locale = "en_GB.UTF-8")
output <- plot(result_Gratiot)
# set back the month name in local R language
Sys.setlocale(category = "LC_TIME", locale = "")
# plot based on quantiles of mcmc object
plot(result_Gratiot, resultmcmc=result_Gratiot_mcmc,
      plot.objects=c("observations", "MCMC.quantiles"))
plot(result_Gratiot, resultmcmc=result_Gratiot_mcmc,
      plot.objects=c("observations", "ML.SD", "ML.quantiles"))
plot(result_Gratiot, resultmcmc=result_Gratiot_mcmc,
      plot.objects=c("observations", "ML.SD", "MCMC.quantiles"))
plot(result_Gratiot, resultmcmc=result_Gratiot_mcmc,
      plot.objects=c("observations", "ML.quantiles", "MCMC.quantiles"))

## End(Not run)

```

plot.phenologydata *Plot the phenology from a add_phenology().*

Description

The function plot.phenologydata plots the observations.

Usage

```

## S3 method for class 'phenologydata'
plot(
  x,
  ...,
  series = "all",
  moon = FALSE,
  season = NULL,
  col.observations = "black",
  col.minimum.observations = "blue",
  col.grouped.observations = "green"
)

```

Arguments

x	A result file generated by add_phenology
...	Parameters used by plot
series	Name or number of series to be plotted or 'all'
moon	If TRUE, the moon phase is plotted. Default is FALSE
season	Which season to plot

`col.observations`
Color of the points

`col.minimum.observations`
Color of the points indicating minimum counts

`col.grouped.observations`
Color of the lines indicating grouped observations

Details

`plot.phenologydata` plots the observation data of phenology.

Value

Nothing.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Phenology model: `AutoFitPhenology()`, `BE_to_LBLE()`, `Gratiot`, `LBLE_to_BE()`, `LBLE_to_L()`, `L_to_LBLE()`, `MarineTurtles_2002`, `MinBMinE_to_Min()`, `adapt_parameters()`, `add_SE()`, `add_phenology()`, `extract_result()`, `fit_phenology()`, `likelihood_phenology()`, `logLik.phenology()`, `map_Gratiot`, `map_phenology()`, `par_init()`, `phenology()`, `phenology2fitRMU()`, `phenology_MHmcmc()`, `phenology_MHmcmc_p()`, `plot.phenology()`, `plot.phenologymap()`, `plot_delta()`, `plot_phi()`, `print.phenology()`, `print.phenologymap()`, `print.phenologyout()`, `remove_site()`, `result_Gratiot`, `result_Gratiot1`, `result_Gratiot2`, `result_Gratiot_Flat`, `summary.phenology()`, `summary.phenologymap()`, `summary.phenologyout()`

Examples

```
## Not run:
library(phenology)
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot <- add_phenology(Gratiot, name = "Complete",
  reference = as.Date("2001-01-01"), format="%d/%m/%Y")

plot(data_Gratiot)

## End(Not run)
```

plot.phenologymap	<i>Plot a likelihood map with Delta and Phi varying.</i>
-------------------	--

Description

This function plots a likelihood map obtained after map_phenology.

Usage

```
## S3 method for class 'phenologymap'
plot(x, ..., col = heat.colors(128), xlab = "Phi", ylab = "Delta")
```

Arguments

x	A map generated with map_phenology.
...	not used
col	Colors could be heat.colors(128) or rainbow(64) or col=gray(c(seq(0, 1, length.out=128)))
xlab	Label for x axis
ylab	Label for y axis

Details

plot.phenologymap plots a likelihood map with Delta and Phi varying.

Value

Return None

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
## Not run:
library("phenology")
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot<-add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
result_Gratiot<-fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=NULL)
data(result_Gratiot)
# Extract the fitted parameters
parg1<-extract_result(result_Gratiot)
# Add constant Alpha and Tau values
# [day d amplitude=(Alpha+Nd*Beta)^Tau with Nd being the number of counts for day d]
pfixed<-c(parg1, Alpha=0, Tau=1)
pfixed<-pfixed[-which(names(pfixed)=="Theta")]
# The only fitted parameter will be Beta
parg2<-c(Beta=0.5, parg1["Theta"])
# Generate a likelihood map
# [default Phi=seq(from=0.1, to=20, length.out=100) but it is very long]
# Take care, it takes 20 hours ! The data map_Gratiot has the result
map_Gratiot<-map_phenology(data=data_Gratiot,
  Phi=seq(from=0.1, to=20, length.out=100),
  fitted.parameters=parg2, fixed.parameters=pfixed)
data(map_Gratiot)
# Plot the map
plot(map_Gratiot, col=heat.colors(128))

## End(Not run)
```

plot.Remigration	<i>Plot the remigration intervals.</i>
------------------	--

Description

Plot the remigration intervals.

Usage

```
## S3 method for class 'Remigration'
plot(x, legend = TRUE, ...)
```

Arguments

x	Object obtained from Bayesian.remigration()
legend	TRUE or FALSE or c(x, y)
...	Parameters transmitted to plot

Details

plot.Remigration plots the remigration intervals.

Value

An invisible dataframe with values used for plotting.

Author(s)

Marc Girondot

See Also

Other Model of Remigration Interval: [Bayesian.remigration\(\)](#), [LnRI_norm\(\)](#), [RI\(\)](#), [RI2BP\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example

# Each year a fraction of 0.9 is surviving
s <- c(s=0.9)
# Probability of tag retention; 0.8
t <- c(t=0.8)
# Time-conditional return probability - This is the true remigration rate
r <- c(r1=0.1, r2=0.8, r3=0.7, r4=0.7, r5=1)
# Capture probability
p <- c(p1=0.6, p2=0.6, p3=0.6, p4=0.6, p5=0.6)
# Number of observations for 400 tagged females after 1, 2, 3, 4, and 5 years
OBS <- c(400, 10, 120, 40, 20, 10)

kl_s <- length(s)
kl_t <- length(t)
kl_r <- length(r)
kl_p <- length(p)

pMCMC <- data.frame(Density=c("newdbeta", "newdbeta", rep("dunif", kl_r),
                             rep("newdbeta", kl_p), "dunif"),
                    Prior1=c(s, t, rep(0, kl_r), rep(0.2, kl_p), 0),
                    Prior2=c(0.02, 0.02, rep(1, kl_r), rep(0.08, kl_p), 10),
                    SDProp=c(0.05, 0.05, rep(0.05, kl_r), rep(0.05, kl_p), 0.05),
                    Min=c(0, 0, rep(0, kl_r), rep(0, kl_p), 0),
                    Max=c(1, 1, rep(1, kl_r), rep(1, kl_p), 10),
                    Init=c(s, t, r, p, 1), stringsAsFactors = FALSE,
```

```

                                row.names=c("s",
                                              "t",
                                              names(r),
                                              names(p), "sd")
                                )
rMCMC <- Bayesian.remigration(parameters = pMCMC,
n.iter = 1000000,
n.adapt = 300000,
trace=10000,
data=OBS)

plot(rMCMC)

## End(Not run)

```

plot.TableECFOCF	<i>Plot a TableECFOCF dataset.</i>
------------------	------------------------------------

Description

This function plots a CMR file summarized using TableECFOCF().

Usage

```

## S3 method for class 'TableECFOCF'
plot(
  x,
  ...,
  result = "ecfocf",
  period = 1,
  cex.points = 4,
  pch = 19,
  col = "black",
  cex.axis = 0.8,
  cex.labels = 0.5,
  col.labels = "red",
  show.labels = FALSE,
  show.0 = FALSE,
  pch.0 = 4,
  cex.0 = 0.5,
  col.0 = "blue",
  show.scale = TRUE,
  max.scale = NULL
)

```

Arguments

<code>x</code>	A CMR file summarized using <code>TableECFOCF()</code>
<code>...</code>	Graphic parameters
<code>result</code>	What should be plotted: ECFOCF or data, ECF, OCF.
<code>period</code>	The period that will be plotted.
<code>cex.points</code>	The maximum magnification to be used for points relative to the current setting of <code>cex</code> .
<code>pch</code>	Character to be used for points.
<code>col</code>	Color to be used for points.
<code>cex.axis</code>	The magnification to be used for axis annotation relative to the current setting of <code>cex</code> .
<code>cex.labels</code>	The magnification to be used for figures.
<code>col.labels</code>	Color of figures.
<code>show.labels</code>	Logical to be used to show figures.
<code>show.0</code>	Logical to show 0 counts.
<code>pch.0</code>	Character used for 0 counts.
<code>cex.0</code>	The magnification to be used for character for 0 counts.
<code>col.0</code>	Color of characters for 0 counts.
<code>show.scale</code>	If TRUE, show the scale as a legend
<code>max.scale</code>	Maximum value for scale; if NULL it is maximum of observations.

Details

`plot.TableECFOCF` plots a `TableECFOCF` dataset.

Value

Nothing

Author(s)

Marc Girondot

See Also

Other Model of Clutch Frequency: [ECFOCF_f\(\)](#), [ECFOCF_full\(\)](#), [TableECFOCF\(\)](#), [fitCF\(\)](#), [fitCF_MHmcmc\(\)](#), [fitCF_MHmcmc_p\(\)](#), [generateCF\(\)](#), [lnLCF\(\)](#), [logLik.ECFOCF\(\)](#), [plot.ECFOCF\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
data(MarineTurtles_2002)
ECFOCF_2002 <- TableECFOCF(MarineTurtles_2002)
par(mar=c(4, 4, 1, 1)+0.4)
plot(ECFOCF_2002, bty="n", las=1, cex.points=3,
      cex.axis = 0.8, main="Year 2002")
plot(ECFOCF_2002, bty="n", las=1, cex.points=5, cex.0=0.2,
      col="red", show.0 = TRUE, col.0="blue")
plot(ECFOCF_2002, bty="n", las=1, cex.points=3, col="lightgrey",
      col.labels = "red", show.labels=TRUE)
plot(ECFOCF_2002, bty="n", las=1, cex.points=3, pch=NA,
      col.labels = "red", show.labels=TRUE)
plot(ECFOCF_2002, bty="n", las=1, cex.points=3, pch=NA,
      col.labels = "red", show.labels=TRUE, cex.0=0.2,
      show.0 = TRUE, col.0="blue", pch.0=4)
plot(ECFOCF_2002, bty="n", las=1, result="OCF")
plot(ECFOCF_2002, bty="n", las=1, result="ECF")
plot(ECFOCF_2002, bty="n", las=1, result="ECF", type="l", main="2002 season",
      xlab="Clutch frequency")
par(new=TRUE)
plot(ECFOCF_2002, bty="n", las=1, result="OCF", type="l", main="",
      ylim=ScalePreviousPlot()$ylim[c("begin", "end")],
      xlab="", ylab="",
      col="red",
      xaxt="n", yaxt="n", axes=FALSE)
legend("topright", legend=c("OCF", "ECF"), lty=1, col=c("red", "black"))

ECFOCF_2002 <- TableECFOCF(MarineTurtles_2002, date0=as.Date("2002-01-01"))

plot(ECFOCF_2002, period=13)

## End(Not run)
```

plot.Tagloss

Plot the daily rate of tag loss.

Description

Plot the daily rate of tag loss.

To use this function without a result of Tagloss_fit(), see the hack in examples.

Usage

```
## S3 method for class 'Tagloss'
plot(
  x,
```

```

t = NULL,
fitted.parameters = NULL,
fixed.parameters = NULL,
scale = 1,
model_before = NULL,
model_after = NULL,
model = c("1", "2", "R1", "R2", "L1", "L2", "cumul", "cumul1", "N2", "N1", "N0", "NLR",
          "N0R", "NL0", "N00"),
col = rev(grey.colors(4, start = 0.9, end = 0.3)),
text.col = grey.colors(4, start = 0.9, end = 0.3),
label.col = "black",
add = FALSE,
mcmc = FALSE,
Hessian = NULL,
replicates = NULL,
method = NULL,
probs = c(0.025, 0.975),
progressbar = FALSE,
decoration = FALSE,
...
)

```

Arguments

x	Object obtained from Tagloss_fit()
t	Time for which values of model must be plotted
fitted.parameters	Set of parameters
fixed.parameters	Another set of parameters without standard error associated
scale	Scale value. When Cumul is used, scale is always 1.
model_before	Transformation of parameters before to use Tagloss_model()
model_after	Transformation of parameters after to use Tagloss_model()
model	Can be 1, 2, R1, R2, L1, L2 or Cumul (2 tags) or Cumul1 (1 tag)
col	The colors of shading areas of cumul or the color of line
text.col	The text color for cumul model
label.col	The text color used for labels when decoration is true
add	Should the data be added to a previous plot?
mcmc	The mcmc result
Hessian	Hessian matrix of parameters
replicates	Number of replicates for confidence interval
method	Which method to use to estimate confidence interval
probs	Quantiles to show for confidence interval
progressbar	Is shown a progressbar?
decoration	Try to add name of parameters on the graph
...	Parameters transmitted to plot

Details

plot.tagloss plots the daily rate of tag loss.

Value

An invisible dataframe with values used for plotting.

Author(s)

Marc Girondot

See Also

Other Model of Tag-loss: [Tagloss_L\(\)](#), [Tagloss_LengthObs\(\)](#), [Tagloss_cumul\(\)](#), [Tagloss_daymax\(\)](#), [Tagloss_fit\(\)](#), [Tagloss_format\(\)](#), [Tagloss_mcmc\(\)](#), [Tagloss_mcmc_p\(\)](#), [Tagloss_model\(\)](#), [Tagloss_simulate\(\)](#), [logLik.Tagloss\(\)](#), [o_4p1p2](#), [plot.TaglossData\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
t <- 1:1000

par <- c(D1_1=200, D2D1_1=100, D3D2_1=200,
        A_1=-logit(0.02), B_1=-logit(0.05), C_1=-logit(0.07))
phenology::plot.Tagloss(x=list(), t=t, fitted.parameters=par, model="1")
phenology::plot.Tagloss(x=list(), t=t, fitted.parameters=par, model="1",
                        scale=1000, decoration = TRUE)

par <- c(D1_2=200, D2D1_2=100, D3D2_2=200,
        A_2=-logit(0.05), B_2=-logit(0.03), C_2=-logit(0.03))
phenology::plot.Tagloss(x=list(), t=t, fitted.parameters=par, ylim=c(0, 1),
                        scale = 10, model="2", decoration = TRUE)

par <- c(D1_L2=200, D2D1_L2=100, D3D2_L2=200,
        A_L2=-logit(0.05), B_L2=-logit(0.07), C_L2=-logit(0.07))
phenology::plot.Tagloss(x=list(), t=t, fitted.parameters=par, model="L2")

par <- c(D1_R2=200, D2D1_R2=0, D3D2_R2=700,
        A_R2=-logit(0.02), B_R2=-logit(0.05), C_R2=-logit(0.07))
phenology::plot.Tagloss(x=list(), t=t, fitted.parameters=par, model="R2",
                        col="red", add=TRUE)

par <- c(D1_L1=200, D2D1_L1=2000, D3D2_L1=2000,
        A_L1=-logit(0.05), B_L1=-logit(0.02), C_L1=-logit(0.1))
phenology::plot.Tagloss(x=list(), t=t, fitted.parameters=par, model="L1")

# To plot the history of individuals
par <- c(D1_1=200, D2D1_1=100, D3D2_1=200,
        A_1=-logit(5E-4), B_1=-logit(4E-4), C_1=-logit(5E-4),
        D1_2=200, D2D1_2=100, D3D2_2=200,
```

```

      A_2=-logit(6E-4), B_2=-logit(5E-4), C_2=-logit(6E-4))
phenology:::plot.Tagloss(x=list(), t=1:1000, fitted.parameters=par,
                        model="Cumul",
                        decoration = TRUE)
phenology:::plot.Tagloss(x=list(), t=1:1000, fitted.parameters=par,
                        model="Cumul",
                        decoration = TRUE, col=c("red", "green", "blue"))

# To plot the history of individuals
par <- c(D1_R1=200, D2D1_R1=300, D3D2_R1=200,
        A_R1=-logit(5E-4), B_R1=-logit(4E-4), C_R1=-logit(5E-4),
        D1_R2=200, D2D1_R2=200, D3D2_R2=200,
        A_R2=-logit(6E-4), B_R2=-logit(5E-4), C_R2=-logit(6E-4),
        D1_L1=200, D2D1_L1=400, D3D2_L1=200,
        A_L1=-logit(5E-4), B_L1=-logit(4E-4), C_L1=-logit(5E-4),
        D1_L2=200, D2D1_L2=100, D3D2_L2=200,
        A_L2=-logit(6E-4), B_L2=-logit(5E-4), C_L2=-logit(6E-4))
phenology:::plot.Tagloss(x=list(), t=1:1000, fitted.parameters=par, model="Cumul",
                        decoration = TRUE)
phenology:::plot.Tagloss(x=list(), t=1:1000, fitted.parameters=par, model="R1",
                        decoration = TRUE)
phenology:::plot.Tagloss(x=list(), t=1:1000, fitted.parameters=par, model="R2",
                        decoration = TRUE)

# Example of fit

data_f_21 <- Tagloss_format(outLR, model="21")
data_f_21_fast <- subset(data_f_21, subset=(is.na(data_f_21$N20)))
par <- c('D1_2' = 49.78891736351531,
        'D2D1_2' = 1059.3635769732305,
        'D3D2_2' = 12.434313273804602,
        'A_2' = 5.2238379144659683,
        'B_2' = 8.0050044071275543,
        'C_2' = 8.4317863609499675,
        'D1_1' = 701.80273287212935,
        'D2D1_1' = 0.010951749100596819,
        'D3D2_1' = 3773.6290607434876,
        'A_1' = 205.42435592344776,
        'B_1' = 9.9598342503239863,
        'C_1' = 6.7234868237164722)
phenology:::plot.Tagloss(x=list(), t=1:1000, fitted.parameters=par, model="1",
                        decoration = TRUE)
phenology:::plot.Tagloss(x=list(), t=1:1000, fitted.parameters=par, model="2",
                        decoration = TRUE)
o <- Tagloss_fit(data=data_f_21_fast, fitted.parameters=par, hessian = TRUE)
plot(x=o, model="1", replicates=0,
     method=NULL, decoration = TRUE)
plot(x=o, model="1", replicates=1000,
     method="Hessian", decoration = TRUE)

## End(Not run)

```

plot.TaglossData	<i>Plot data used for tagloss analysis.</i>
------------------	---

Description

This function plots the result of Tagloss_format().
 The default ramp of colors is a grey ramp.

Usage

```
## S3 method for class 'TaglossData'
plot(
  x,
  ...,
  categories = c("N22", "N21", "N11", "N10", "N20"),
  col = grey(seq(from = 0.9, to = 0, length.out = length(categories))),
  title.legend = "Tag history",
  categories.legend = categories,
  show.legend = TRUE
)
```

Arguments

x	A result for Tagloss_format.
...	Graphic parameters, see par().
categories	Categories to display.
col	The ramp of colors used for the categories.
title.legend	Title for legend box.
categories.legend	Name of categories to show in legend box.
show.legend	Should the legend box be shown ?

Details

plot.TaglossData plots formatted data used for tagloss analysis

Value

Nothing

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Model of Tag-loss: `Tagloss_L()`, `Tagloss_LengthObs()`, `Tagloss_cumul()`, `Tagloss_daymax()`, `Tagloss_fit()`, `Tagloss_format()`, `Tagloss_mcmc()`, `Tagloss_mcmc_p()`, `Tagloss_model()`, `Tagloss_simulate()`, `logLik.Tagloss()`, `o_4p_p1p2`, `plot.Tagloss()`

Examples

```
## Not run:
library(phenology)
# Example
data_f_21 <- Tagloss_format(outLR, model="21")
plot(data_f_21)

## End(Not run)
```

plot_delta

Plot a likelihood lineplot obtained after map_phenology.

Description

This function plots a likelihood lineplot obtained after `map_phenology`.

Usage

```
plot_delta(map = NULL, Phi = NULL, ...)
```

Arguments

<code>map</code>	A map generated with <code>map_phenology</code>
<code>Phi</code>	Phi value or NULL
<code>...</code>	Parameters for plot

Details

`plot_delta` plots the likelihood delta for fixed Phi value.

Value

Return None

Author(s)

Marc Girondot

See Also

Other Phenology model: `AutoFitPhenology()`, `BE_to_LBLE()`, `Gratiot`, `LBLE_to_BE()`, `LBLE_to_L()`, `L_to_LBLE()`, `MarineTurtles_2002`, `MinBMinE_to_Min()`, `adapt_parameters()`, `add_SE()`, `add_phenology()`, `extract_result()`, `fit_phenology()`, `likelihood_phenology()`, `logLik.phenology()`, `map_Gratiot`, `map_phenology()`, `par_init()`, `phenology()`, `phenology2fitRMU()`, `phenology_MHmcmc()`, `phenology_MHmcmc_p()`, `plot.phenology()`, `plot.phenologydata()`, `plot.phenologymap()`, `plot_phi()`, `print.phenology()`, `print.phenologymap()`, `print.phenologyout()`, `remove_site()`, `result_Gratiot`, `result_Gratiot1`, `result_Gratiot2`, `result_Gratiot_Flat`, `summary.phenology()`, `summary.phenologymap()`, `summary.phenologyout()`

Examples

```
## Not run:
library("phenology")
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot<-add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
result_Gratiot<-fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=NULL)
data(result_Gratiot)
# Extract the fitted parameters
parg1<-extract_result(result_Gratiot)
# Add constant Alpha and Tau values
# [day d amplitude=(Alpha+Nd*Beta)^Tau with Nd being the number of counts for day d]
pfixed<-c(parg1, Alpha=0, Tau=1)
pfixed<-pfixed[-which(names(pfixed)=="Theta")]
# The only fitted parameter will be Beta
parg2<-c(Beta=0.5, parg1["Theta"])
# Generate a likelihood map
# [default Phi=seq(from=0.1, to=20, length.out=100) but it is very long]
# Take care, it takes 20 hours ! The data map_Gratiot has the result
map_Gratiot<-map_phenology(data=data_Gratiot,
  Phi=seq(from=0.1, to=20, length.out=100),
  fitted.parameters=parg2, fixed.parameters=pfixed)
data(map_Gratiot)
# Plot the min(-Ln L) for Delta varying with Phi equal to the value for maximum likelihood
plot_delta(map=map_Gratiot)
# Plot the min(-Ln L) for Delta varying with Phi the nearest to 15
plot_delta(map=map_Gratiot, Phi=15)

## End(Not run)
```

Description

The function "plot_phi" plots the best likelihood for each Phi value.

Usage

```
plot_phi(map = NULL, ...)
```

Arguments

map	A map generated with map_phenology
...	Parameters for plot

Details

plot_phi plots the best likelihood for fixed Phi value.

Value

Return None

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
library("phenology")
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot<-add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
## Not run:
result_Gratiot<-fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=NULL)
```

```
## End(Not run)
data(result_Gratiot)
# Extract the fitted parameters
parg1<-extract_result(result_Gratiot)
# Add constant Alpha and Tau values
# [day d amplitude=(Alpha+Nd*Beta)^Tau with Nd being the number of counts for day d]
pfixed<-c(parg1, Alpha=0, Tau=1)
pfixed<-pfixed[-which(names(pfixed)=="Theta")]
# The only fitted parameter will be Beta
parg2<-c(Beta=0.5, parg1["Theta"])
# Generate a likelihood map [default Phi=seq(from=0.1, to=20, length.out=100) but it is very long]
# Take care, it takes 20 hours ! The data map_Gratiot has the result
## Not run:
map_Gratiot<-map_phenology(data=data_Gratiot,
  Phi=seq(from=0.1, to=20, length.out=100),
  fitted.parameters=parg2, fixed.parameters=pfixed)

## End(Not run)
data(map_Gratiot)
# Plot the min(-Ln L) for Phi varying at any delta value
plot_phi(map=map_Gratiot)
```

print.phenology

Print the result information from a result x.

Description

The function print displays a phenology result.

Usage

```
## S3 method for class 'phenology'
print(x, ...)
```

Arguments

x	A result file generated by fit_phenology
...	Not used

Details

print.phenology prints the information from a result x.

Value

None

Author(s)

Marc Girondot

See Also

Other Phenology model: `AutoFitPhenology()`, `BE_to_LBLE()`, `Gratiot`, `LBLE_to_BE()`, `LBLE_to_L()`, `L_to_LBLE()`, `MarineTurtles_2002`, `MinBMinE_to_Min()`, `adapt_parameters()`, `add_SE()`, `add_phenology()`, `extract_result()`, `fit_phenology()`, `likelihood_phenology()`, `logLik.phenology()`, `map_Gratiot`, `map_phenology()`, `par_init()`, `phenology()`, `phenology2fitRMU()`, `phenology_MHmcmc()`, `phenology_MHmcmc_p()`, `plot.phenology()`, `plot.phenologydata()`, `plot.phenologymap()`, `plot_delta()`, `plot_phi()`, `print.phenologymap()`, `print.phenologyout()`, `remove_site()`, `result_Gratiot`, `result_Gratiot1`, `result_Gratiot2`, `result_Gratiot_Flat`, `summary.phenology()`, `summary.phenologymap()`, `summary.phenologyout()`

Examples

```
library(phenology)
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot <- add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg <- par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
## Not run:
result_Gratiot <- fit_phenology(data=data_Gratiot,
  fitted.parameters=parg,
  fixed.parameters=NULL)

## End(Not run)
data(result_Gratiot)
# Show the result
result_Gratiot
```

<code>print.phenologymap</code>	<i>Print information on a phenologymap object.</i>
---------------------------------	--

Description

`print.phenologymap` print information on a phenologymap object

Usage

```
## S3 method for class 'phenologymap'
print(x, ...)
```

Arguments

<code>x</code>	A map generated with <code>map_phenology</code> .
<code>...</code>	Not used

Value

Return None

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
library("phenology")
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot<-add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
## Not run:
result_Gratiot<-fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=NULL)

## End(Not run)
data(result_Gratiot)
# Extract the fitted parameters
parg1<-extract_result(result_Gratiot)
# Add constant Alpha and Tau values
# [day d amplitude=(Alpha+Nd*Beta)^Tau with Nd being the number of counts for day d]
pfixed<-c(parg1, Alpha=0, Tau=1)
pfixed<-pfixed[-which(names(pfixed)=="Theta")]
# The only fitted parameter will be Beta
parg2<-c(Beta=0.5, parg1["Theta"])
# Generate a likelihood map
# [default Phi=seq(from=0.1, to=20, length.out=100) but it is very long]
# Take care, it takes 20 hours ! The data map_Gratiot has the result
## Not run:
map_Gratiot<-map_phenology(data=data_Gratiot,
  Phi=seq(from=0.1, to=20, length.out=100), fitted.parameters=parg2,
  fixed.parameters=pfixed)
```

```
## End(Not run)
data(map_Gratiot)
# Print the information on a map
map_Gratiot
```

print.phenologyout	<i>Print the information from a output x.</i>
--------------------	---

Description

The function print.phenologyout displays the output of a summary(phenology).

Usage

```
## S3 method for class 'phenologyout'
print(x, ...)
```

Arguments

x	An output generated by plot_phenology
...	Not used

Details

print.phenologyout prints the information from a result x.

Value

None

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
## Not run:
library(phenology)
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot<-add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
result_Gratiot <- fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=NULL)
data(result_Gratiot)
# Plot the phenology and get some stats
output <- plot(result_Gratiot)
# Show the output
output

## End(Not run)
```

remove_site

*Removes site information from a set of parameters.***Description**

This function is used to remove the information of the site from a set of parameters. It can be used to other timeseries after.

Usage

```
remove_site(parameters = NULL, help = FALSE)
```

Arguments

parameters	Set of parameters
help	If TRUE, an help is displayed

Details

remove_site removes beach information from a set of parameters.

Value

Return a set of modified parameters

Author(s)

Marc Girondot

See Also

Other Phenology model: `AutoFitPhenology()`, `BE_to_LBLE()`, `Gratiot`, `LBLE_to_BE()`, `LBLE_to_L()`, `L_to_LBLE()`, `MarineTurtles_2002`, `MinBMinE_to_Min()`, `adapt_parameters()`, `add_SE()`, `add_phenology()`, `extract_result()`, `fit_phenology()`, `likelihood_phenology()`, `logLik.phenology()`, `map_Gratiot`, `map_phenology()`, `par_init()`, `phenology()`, `phenology2fitRMU()`, `phenology_MHmcmc()`, `phenology_MHmcmc_p()`, `plot.phenology()`, `plot.phenologydata()`, `plot.phenologymap()`, `plot_delta()`, `plot_phi()`, `print.phenology()`, `print.phenologymap()`, `print.phenologyout()`, `result_Gratiot`, `result_Gratiot1`, `result_Gratiot2`, `result_Gratiot_Flat`, `summary.phenology()`, `summary.phenologymap()`, `summary.phenologyout()`

Examples

```
library(phenology)
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot<-add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
## Not run:
result_Gratiot<-fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=NULL)

## End(Not run)
data(result_Gratiot)
# Extract parameters from result
parg<-extract_result(result_Gratiot)
# Remove site information
parg1<-remove_site(parg)
```

result_Gratiot

Result of the fit of Leatherback nest counts

Description

Result of the fit of Leatherback nest counts from Gratiot et al. (2006) Figure 1 The phenology has been fitted with MinE, MinB, Max, Flat, LengthB, LengthE, Peak, Theta.

Usage

```
result_Gratiot
```

Format

A list with Gratiot data and the result of the fit.

Details

Result of the fit of Leatherback nest counts from Gratiot et al. (2006) Figure 1

Author(s)

Marc Girondot <marc.girondot@u-psud.fr>

References

Gratiot, N., Gratiot, J., de Thoisy, B. & Kelle, L. 2006. Estimation of marine turtles nesting season from incomplete data ; statistical adjustment of a sinusoidal function. *Animal Conservation*, 9, 95-102.

See Also

Other Phenology model: `AutoFitPhenology()`, `BE_to_LBLE()`, `Gratiot`, `LBLE_to_BE()`, `LBLE_to_L()`, `L_to_LBLE()`, `MarineTurtles_2002`, `MinBMinE_to_Min()`, `adapt_parameters()`, `add_SE()`, `add_phenology()`, `extract_result()`, `fit_phenology()`, `likelihood_phenology()`, `logLik.phenology()`, `map_Gratiot`, `map_phenology()`, `par_init()`, `phenology()`, `phenology2fitRMU()`, `phenology_MHmcmc()`, `phenology_MHmcmc_p()`, `plot.phenology()`, `plot.phenologydata()`, `plot.phenologymap()`, `plot_delta()`, `plot_phi()`, `print.phenology()`, `print.phenologymap()`, `print.phenologyout()`, `remove_site()`, `result_Gratiot1`, `result_Gratiot2`, `result_Gratiot_Flat`, `summary.phenology()`, `summary.phenologymap()`, `summary.phenologyout()`

Examples

```
## Not run:
library(phenology)
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot <- add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg <- par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
result_Gratiot <- fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=NULL)
# Read a file with result
data(result_Gratiot)

## End(Not run)
```

result_Gratiot1

Result of the fit of Leatherback nest counts

Description

Result of the fit of Leatherback nest counts from Gratiot et al. (2006) Figure 1 The phenology has been fitted with MinE, MinB, Max, Flat, LengthB, LengthE, Peak, Theta, Alpha, Beta, Tau, Phi, Delta

Usage

```
result_Gratiot1
```

Format

A list with Gratiot data and the result of the fit.

Details

Result of the fit of Leatherback nest counts from Gratiot et al. (2006) Figure 1

Author(s)

Marc Girondot <marc.girondot@u-psud.fr>

References

Gratiot, N., Gratiot, J., de Thoisy, B. & Kelle, L. 2006. Estimation of marine turtles nesting season from incomplete data ; statistical adjustment of a sinusoidal function. *Animal Conservation*, 9, 95-102.

See Also

Other Phenology model: `AutoFitPhenology()`, `BE_to_LBLE()`, `Gratiot`, `LBLE_to_BE()`, `LBLE_to_L()`, `L_to_LBLE()`, `MarineTurtles_2002`, `MinBMinE_to_Min()`, `adapt_parameters()`, `add_SE()`, `add_phenology()`, `extract_result()`, `fit_phenology()`, `likelihood_phenology()`, `logLik.phenology()`, `map_Gratiot`, `map_phenology()`, `par_init()`, `phenology()`, `phenology2fitRMU()`, `phenology_MHmcmc()`, `phenology_MHmcmc_p()`, `plot.phenology()`, `plot.phenologydata()`, `plot.phenologymap()`, `plot_delta()`, `plot_phi()`, `print.phenology()`, `print.phenologymap()`, `print.phenologyout()`, `remove_site()`, `result_Gratiot`, `result_Gratiot2`, `result_Gratiot_Flat`, `summary.phenology()`, `summary.phenologymap()`, `summary.phenologyout()`

Examples

```
library(phenology)
# Read a file with result
data(result_Gratiot1)
```

```
result_Gratiot2
```

Result of the fit of Leatherback nest counts

Description

Result of the fit of Leatherback nest counts from Gratiot et al. (2006) Figure 1 The phenology has been fitted with MinE, MinB, Max, Flat, LengthB, LengthE, Peak, Theta, Alpha, Beta, Tau, Phi, Delta, Alpha1, Beta1, Tau1, Phi1, Delta1.

Usage

```
result_Gratiot2
```

Format

A list with Gratiot data and the result of the fit.

Details

Result of the fit of Leatherback nest counts from Gratiot et al. (2006) Figure 1

Author(s)

Marc Girondot <marc.girondot@u-psud.fr>

References

Gratiot, N., Gratiot, J., de Thoisy, B. & Kelle, L. 2006. Estimation of marine turtles nesting season from incomplete data ; statistical adjustment of a sinusoidal function. *Animal Conservation*, 9, 95-102.

See Also

Other Phenology model: `AutoFitPhenology()`, `BE_to_LBLE()`, `Gratiot`, `LBLE_to_BE()`, `LBLE_to_L()`, `L_to_LBLE()`, `MarineTurtles_2002`, `MinBMinE_to_Min()`, `adapt_parameters()`, `add_SE()`, `add_phenology()`, `extract_result()`, `fit_phenology()`, `likelihood_phenology()`, `logLik.phenology()`, `map_Gratiot`, `map_phenology()`, `par_init()`, `phenology()`, `phenology2fitRMU()`, `phenology_MHmcmc()`, `phenology_MHmcmc_p()`, `plot.phenology()`, `plot.phenologydata()`, `plot.phenologymap()`, `plot_delta()`, `plot_phi()`, `print.phenology()`, `print.phenologymap()`, `print.phenologyout()`, `remove_site()`, `result_Gratiot`, `result_Gratiot1`, `result_Gratiot_Flat`, `summary.phenology()`, `summary.phenologymap()`, `summary.phenologyout()`

Examples

```
library(phenology)
# Read a file with result
data(result_Gratiot2)
```

result_Gratiot_Flat	<i>Result of the fit of Leatherback nest counts</i>
---------------------	---

Description

Result of the fit of Leatherback nest counts from Gratiot et al. (2006) Figure 1. The phenology has been fitted with MinE, MinB, Max, LengthB, LengthE, Peak, Theta. The Flat parameter is set to 0 and is not fitted.

Usage

```
result_Gratiot_Flat
```

Format

A list with Gratiot data and the result of the fit.

Details

Result of the fit of Leatherback nest counts from Gratiot et al. (2006) Figure 1

Author(s)

Marc Girondot <marc.girondot@u-psud.fr>

References

Gratiot, N., Gratiot, J., de Thoisy, B. & Kelle, L. 2006. Estimation of marine turtles nesting season from incomplete data ; statistical adjustment of a sinusoidal function. *Animal Conservation*, 9, 95-102.

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
library(phenology)
# Read a file with result
data(result_Gratiot_Flat)
```

RI

Return an expected remigration interval.

Description

Model of remigration interval

Note that r, s and t are conditional probabilities. If c is null, then return probabilities are estimated from r. r can be named vector. For example:

`r <- c(r1=0.5, r2=0.60, r3=1)` is equivalent to `c <- c(c1=0.5, c2=0.3, c3=0.2)`

The vector of r described the probability that a female returned after 1, 2, 3 years among those who have not nested before. The vector of r is the same but defining the return probability for an initial

female.

Usage

```
RI(s, t, r = NULL, c = NULL, p)
```

Arguments

s	Time-conditional probability of survival
t	Time-conditional probability of tag retention
r	Time-conditional probability of return
c	Probability of return
p	Annual probability of observation

Details

RI returns an expected remigration interval

Value

Return a remigration interval.

Author(s)

Marc Girondot

See Also

Other Model of Remigration Interval: [Bayesian.remigration\(\)](#), [LnRI_norm\(\)](#), [RI2BP\(\)](#), [plot.Remigration\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
s <- c(s1=1, s2=1, s3=1, s4=1, s5=1)
t <- c(t1=0.95, t2=1, t3=1, t4=1, t5=1)
r <- c(r1=0.1, r2=0.8, r3=0.7, r4=0.7, r5=1)
p <- c(p1=0.6, p2=0.6, p3=0.6, p4=0.6, p5=0.6)

# r is equivalent to
c <- c(c1=0.1, c2=0.72, c3=0.126, c4=0.0378, c5=0.0162)
# Then the true remigration interval is:
ri_true <- sum(1:5*c[1:5])

# BP is then
RI2BP(proportion=c, RI=c(mean=1:5, sd=rep(0, 5)))
```

```

s_ri <- NULL
for (sx in seq(from=0.01, to=1, by=0.01)) {
  s[] <- sx
  ri1 <- RI(s=s, t=t, r=r, p=p)
  s_ri <- c(s_ri, sum(1:5*ri1)/sum(ri1))
}

par(mar=c(4, 4, 1, 1)+0.4)

plot(x=seq(from=0.01, to=1, by=0.01), y=s_ri, type="l",
     las=1, bty="n", ylim=c(0, 4),
     xlab="Annual survival probabilities", ylab="Naive Remigration Interval",
     main="")
segments(x0=0.01, x1=1, y0=ri_true, y1=ri_true, lty=2, col="red")
legend("topright", legend="True remigration interval", lty=2, col="red")

## End(Not run)

```

RI2BP

*Calculate Breeding Proportion from Remigration Interval.***Description**

This function calculates breeding proportion (BP) from Remigration Interval (RI). RI can be a vector of RI, one per individual, or an aggregated value with mean and sd and several categories can exist. See examples.

Usage

```

RI2BP(
  proportion = 1,
  RI = c(mean = 2, sd = 0),
  sampling = 10000,
  replicates = 100
)

```

Arguments

proportion	A vector of the proportion of different categories
RI	The RI of individuals - See description
sampling	Number of individuals for sampling
replicates	Number of replicates to estimate SE

Details

RI2BP calculate Breeding Proportion from Remigration Interval.

Value

Return a vector with mean and se.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Model of Remigration Interval: [Bayesian.remigration\(\)](#), [LnRI_norm\(\)](#), [RI\(\)](#), [plot.Remigration\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
RI <- c(mean=3, sd=0)
RI2BP(RI=RI)

RI <- c(mean=2, sd=0.3)
RI2BP(RI=RI)

RI <- c(mean=4, sd=0)
RI2BP(RI=RI)

RI <- c(mean=c(2, 10), sd=c(0, 0))
proportion <- c(0.5, 0.5)
RI2BP(proportion=proportion, RI=RI)

c <- c(c1=0.1, c2=0.72, c3=0.126, c4=0.0378, c5=0.0162)
plot(1:5, c, xlab="RI", ylab="Proportion", las=1, bty="n", type="h")
RI2BP(proportion=c, RI=c(mean=1:5, sd=rep(0, 5)))

# To generate random RI with known mean and sd using
# a truncated lognormal distribution (because 0 does not exist)
mean <- 2.5 - 1
sd <- 1
location <- log(mean^2 / sqrt(sd^2 + mean^2))
shape <- sqrt(log(1 + (sd^2 / mean^2)))
RI <- round(rlnorm(100, meanlog = location, sdlog = shape))+1
mean(RI); sd(RI) # All is ok !
plot(table(RI), xlab="RI", ylab="Proportion", las=1, bty="n", type="h")
RI2BP(proportion=proportion, RI=RI)

## End(Not run)
```

Sea_Turtles_Bayesian *R code for Girondot, M., & Guillon, J.-M. (2026) Book chapter*

Description

Only the help of this function is usefull.

Usage

```
Sea_Turtles_Bayesian()
```

Details

Sea_Turtles_Bayesian is R code from Girondot, M., & Guillon, J.-M. (2026) Book chapter

Value

The reference of the chapter

Author(s)

Marc Girondot

Examples

```
## Not run:
suppressPackageStartupMessages(library(phenology))
suppressPackageStartupMessages(library(openxlsx))

# Fit phenology of leatherbacks nesting in Ya:lima:po beach in 2002

Nests_Yal2002 <- structure(list(V1 = c("01/03/02", "02/03/02", "03/03/02", "04/03/02",
"05/03/02", "06/03/02", "07/03/02", "08/03/02", "09/03/02", "10/03/02",
"11/03/02", "12/03/02", "13/03/02", "14/03/02", "15/03/02", "16/03/02",
"17/03/02", "18/03/02", "19/03/02", "20/03/02", "21/03/02", "22/03/02",
"23/03/02", "24/03/02", "25/03/02", "26/03/02", "27/03/02", "28/03/02",
"29/03/02", "30/03/02", "31/03/02", "01/04/02", "02/04/02", "03/04/02",
"04/04/02", "05/04/02", "06/04/02", "07/04/02", "08/04/02", "09/04/02",
"10/04/02", "11/04/02", "12/04/02", "13/04/02", "14/04/02", "15/04/02",
"16/04/02", "17/04/02", "18/04/02", "19/04/02", "20/04/02", "21/04/02",
"22/04/02", "23/04/02", "24/04/02", "25/04/02", "26/04/02", "27/04/02",
"28/04/02", "29/04/02", "30/04/02", "01/05/02", "02/05/02", "03/05/02",
"04/05/02", "05/05/02", "06/05/02", "07/05/02", "08/05/02", "09/05/02",
"10/05/02", "11/05/02", "12/05/02", "13/05/02", "14/05/02", "15/05/02",
"16/05/02", "17/05/02", "18/05/02", "19/05/02", "20/05/02", "21/05/02",
"22/05/02", "23/05/02", "24/05/02", "25/05/02", "26/05/02", "27/05/02",
"28/05/02", "29/05/02", "30/05/02", "31/05/02", "01/06/02", "02/06/02",
"03/06/02", "04/06/02", "05/06/02", "06/06/02", "07/06/02", "08/06/02",
"09/06/02", "10/06/02", "11/06/02", "12/06/02", "13/06/02", "14/06/02",
"15/06/02", "16/06/02", "17/06/02", "18/06/02", "19/06/02", "20/06/02",
```

```

"21/06/02", "22/06/02", "23/06/02", "24/06/02", "25/06/02", "26/06/02",
"27/06/02", "28/06/02", "29/06/02", "30/06/02", "01/07/02", "02/07/02",
"03/07/02", "04/07/02", "05/07/02", "06/07/02", "07/07/02", "08/07/02",
"09/07/02", "10/07/02", "11/07/02", "12/07/02", "13/07/02", "14/07/02",
"15/07/02", "16/07/02", "17/07/02", "18/07/02", "19/07/02", "20/07/02",
"21/07/02", "22/07/02", "23/07/02", "24/07/02", "25/07/02", "26/07/02",
"27/07/02", "28/07/02", "29/07/02", "30/07/02", "31/07/02", "01/08/02",
"02/08/02", "03/08/02", "04/08/02", "05/08/02", "06/08/02", "07/08/02",
"08/08/02", "09/08/02", "10/08/02", "11/08/02", "12/08/02", "13/08/02",
"14/08/02", "15/08/02", "16/08/02", "17/08/02", "18/08/02", "19/08/02",
"20/08/02", "21/08/02", "22/08/02", "23/08/02", "24/08/02", "25/08/02",
"26/08/02", "27/08/02", "28/08/02", "29/08/02", "30/08/02", "31/08/02",
"01/09/02", "02/09/02", "03/09/02", "04/09/02", "05/09/02", "06/09/02",
"07/09/02", "08/09/02", "09/09/02", "10/09/02", "11/09/02", "12/09/02",
"13/09/02", "14/09/02", "15/09/02", "16/09/02", "17/09/02", "18/09/02",
"19/09/02", "20/09/02", "21/09/02", "22/09/02", "23/09/02", "24/09/02",
"25/09/02", "26/09/02", "27/09/02", "28/09/02", "29/09/02", "30/09/02",
"01/10/02", "02/10/02", "03/10/02", "04/10/02", "05/10/02", "06/10/02",
"07/10/02", "08/10/02", "09/10/02", "10/10/02", "11/10/02", "12/10/02",
"13/10/02", "14/10/02", "15/10/02", "16/10/02", "17/10/02", "18/10/02",
"19/10/02", "20/10/02", "21/10/02", "22/10/02", "23/10/02", "24/10/02",
"25/10/02", "26/10/02", "27/10/02", "28/10/02", "29/10/02", "30/10/02",
"31/10/02", "01/11/02", "02/11/02", "03/11/02", "04/11/02", "05/11/02",
"06/11/02", "07/11/02", "08/11/02", "09/11/02", "10/11/02", "11/11/02",
"12/11/02", "13/11/02", "14/11/02", "15/11/02", "16/11/02", "17/11/02",
"18/11/02", "19/11/02", "20/11/02", "21/11/02", "22/11/02"),
V2 = c(3L, 2L, 1L, 1L, 6L, 2L, 2L, 2L, 1L, NA, 4L, NA, 1L,
NA, 1L, NA, 3L, 1L, 4L, 4L, 2L, 3L, 1L, 7L, 7L, 5L, 4L, 7L,
4L, 4L, NA, NA, NA, 10L, 21L, 9L, 9L, 17L, 11L, 16L, 18L,
32L, NA, NA, 30L, NA, 40L, 20L, 16L, 18L, 39L, 44L, 32L,
57L, 33L, 43L, 24L, 25L, 15L, 20L, 51L, 40L, 50L, 46L, 51L,
35L, 14L, 50L, 56L, 68L, 69L, 72L, 66L, NA, NA, 63L, 34L,
48L, 31L, NA, NA, NA, NA, 74L, 81L, 58L, NA, NA, 56L, 36L,
37L, 44L, 61L, NA, NA, 35L, NA, 62L, 44L, 69L, NA, NA, NA,
56L, 46L, 27L, 42L, 56L, NA, 69L, 59L, 55L, 35L, 47L, 38L,
NA, 30L, 28L, 38L, 24L, NA, NA, NA, 27L, 35L, 30L, 36L, 41L,
NA, NA, 27L, 21L, 27L, 15L, 22L, NA, 22L, 25L, 26L, 23L,
23L, NA, 14L, NA, NA, NA, NA, NA, 10L, 10L, 7L, 4L, 4L, 5L,
2L, NA, NA, 15L, 7L, 3L, NA, NA, NA, NA, 2L, NA, NA, NA,
NA, NA, 3L, 3L, 2L, 4L, 1L, 0L, NA, NA, 4L, 3L, 3L, 0L, NA,
NA, NA, 0L, 0L, 0L, 0L, 0L, NA, NA, 0L, 0L, 1L, 0L, 0L, NA,
NA, 1L, NA, 2L, NA, 0L, NA, NA, 2L, NA, 0L, NA, 0L, NA, NA,
0L, 1L, 0L, 1L, 1L, NA, NA, NA, 0L, 0L, 0L, 0L, 0L, NA, 3L,
NA, 0L, NA, 0L, NA, NA, 0L, NA, 1L, NA, 0L, NA, NA, 1L, NA,
NA, NA, NA, NA, NA, 0L, NA, 1L, NA, NA, NA, NA, 1L, 1L,
0L, 0L, NA, NA, NA, NA, NA, NA, 1L)), class = "data.frame", row.names = c(NA,
-267L))

```

```
Nests_Yal2002_f <- add_phenology(Nests_Yal2002, month_ref = 1, format = "%d/%m/%y")
```

```
pfixed <- c(Flat=0)
```

```
parg <- par_init(Nests_Yal2002_f, fixed.parameters=pfixed)
```

```
parg <- MinBMinE_to_Min(parg)
```

```

result_Nests_Yal2002_f <- fit_phenology(data = Nests_Yal2002_f, fitted.parameters = parg,
                                       fixed.parameters = pfixed,
                                       control = list(trace = 1, REPORT = 1, maxit = 100))

Priors <- phenology_MHmcmc_p(result_Nests_Yal2002_f, accept = TRUE)

result_Nests_Yal2002_mcmc <- phenology_MHmcmc(result_Nests_Yal2002_f, n.iter = 20000,
                                             adaptive = TRUE, parametersMCMC = Priors, trace = 100)

# Plot the phenology of leatherbacks nesting in Ya:lima:po in 2002

par(mar=c(4, 4, 1, 1))
a <- plot(result_Nests_Yal2002_f, resultmcmc = result_Nests_Yal2002_mcmc,
         plot.objects = c("observations", "MCMC.SD", "MCMC.quantiles"),
         main="", ylim=c(0, 100), las=1)

totalY2002 <- a$details_mcmc$Site
totalY2002 <- cbind(totalY2002, Date=totalY2002[, "Ordinal"]+as.Date("2002-01-01"))

# Table of OCF and ECF for leatherbacks in Ya:lima:po 2002

ECFOCF_2002 <- TableECFOCF(MarineTurtles_2002)

dOCFECF <- c(OCF=sum(rowSums(ECFOCF_2002)*(0:15)),
            ECF=sum(colSums(ECFOCF_2002)*(0:15)))

date_monitored <- sort(unique(MarineTurtles_2002$Date))

# Different priors for capture probabilities

# How many days have been monitored during the nesting season
s_season <- length(seq(from=as.Date("2002-03-01"), to=as.Date("2002-08-01"), by="1 days"))
n_season <- length(unique(date_monitored))

# Using the summary of number of turtles
n <- sum(totalY2002[match(date_monitored, totalY2002$Date), "50%"])
s <- sum(totalY2002[which(totalY2002$Date == "2002-03-01"):
                  which(totalY2002$Date == "2002-08-01"), "50%"])

n_low <- sum(totalY2002[match(date_monitored, totalY2002$Date), "2.5%"])
s_low <- sum(totalY2002[which(totalY2002$Date == "2002-03-01"):
                  which(totalY2002$Date == "2002-08-01"), "2.5%"])

n_high <- sum(totalY2002[match(date_monitored, totalY2002$Date), "97.5%"])
s_high <- sum(totalY2002[which(totalY2002$Date == "2002-03-01"):
                  which(totalY2002$Date == "2002-08-01"), "97.5%"])

variance_summary <- ((n_low/s_low-n_high/s_high)/(1.96*2))^2
mean_summary <- n/s

# Using the posterior predictive distribution of number of turtles

```

```

dates <- seq(from=as.Date("2002-01-01"), to = as.Date("2002-12-31"), by="1 day")
dates_season <- which(dates == "2002-03-01"):which(dates == "2002-08-01")
dates_patrols <- match(date_monitored, dates)

p_sur_s <- NULL
for (j in 1:nrow(result_Nests_Yal2002_mcmc$resultMCMC$`1`)) {
  # The same but using all replicates
  par <- c(result_Nests_Yal2002_mcmc$resultMCMC$`1`[j, ],
           result_Nests_Yal2002_f$fixed.parameters)
  par <- getFromNamespace(".format_par", ns="phenology")(xpar=par, serie="Site")
  countd <- getFromNamespace(".daily_count", ns="phenology")(d=0:364, xpar=par)
  countd[Nests_Yal2002_f$Site$ordinal+1] <- Nests_Yal2002_f$Site$nombbre
  p_sur_s <- c(p_sur_s, sum(countd[dates_patrols])/sum(countd[dates_season]))
}

quantile(p_sur_s, probs=c(0.025, 0.5, 0.975))
variance_PPd <- var(p_sur_s)
mean_PPd <- mean(p_sur_s)
outbeta <- dbeta_new(x=0.5, mu=mean_PPd, v=variance_PPd)
shape1 <- attributes(outbeta)$shape1
shape2 <- attributes(outbeta)$shape2

# First estimate of capture probability

# Using a uniform betal prior

plot(x=seq(from=0, to=1, by=0.01), y=dbeta(x=seq(from=0, to=1, by=0.01), shape1=1, shape2=1),
     type="l", las=1, bty="n", ylab="Density", xlab="x")

Priors <- setPriors1(Name = "pa",
                   Parameters=c(shape1=1, shape2=1),
                   SDProp = 0.01,
                   Density = "dbeta",
                   Init = n_season/s_season,
                   Min=0, Max=1
)

Lprob <- function(data, x) {
  return(-dbinom(x= dOCFECF['OCF'], size= dOCFECF['ECF'], prob=x, log=TRUE))
}

outmcmc <- MHalgoGen(parameters=Priors, data=data, likelihood=Lprob, n.adapt = 100)

plot(outmcmc, what = "markovchain", parameters = "pa", legend = "topleft")
plot(outmcmc, what = "Ln1")
plot(outmcmc, what = "posterior", parameters = "pa")
as.parameters(outmcmc, index = "quantile")

getFromNamespace(".BinomialConfidence", ns="HelpersMG")(x=dOCFECF["OCF"], n=dOCFECF["ECF"],
                 method = "wilson")

layout(mat = matrix(1:2, ncol=2))
par(mar=c(4, 4, 1, 1))

```

```

plot(outmcmc, what = "markovchain", parameters = "pa",
     legend = c(x=ScalePreviousPlot(x=0.05, y=0.20)$x,
                y=ScalePreviousPlot(x=0.05, y=0.20)$y),
     ylim=c(0, 1))
text(x=ScalePreviousPlot(x=0.05, y=0.95)$x,
     y=ScalePreviousPlot(x=0, y=0.9)$y, "A", cex=2)
plot(outmcmc, what = "posterior", parameters = "pa",
     legend = c(x=ScalePreviousPlot(x=0.05, y=0.20)$x,
                y=ScalePreviousPlot(x=0.05, y=0.20)$y),
     ylim=c(0, 20), xlim=c(0, 1),
     breaks = seq(from=0, to=1, by=0.01))
text(x=ScalePreviousPlot(x=0.05, y=0.85)$x, y=ScalePreviousPlot(x=0, y=0.90)$y, "B", cex=2)

# Example with CF and pa being non identifiable

dOCFd <- rowSums(ECFOCF_2002)[-1]
names(dOCFd) <- as.character(1:length(dOCFd))
sum(dOCFd)
LprobP <- function(data, x) return(-sum(dpois(x=as.numeric(rep(names(data), data)),
      lambda=x["pa"]*x["CF"], log=TRUE)))

LprobP(data=dOCFd, x=c(pa=n/s, CF=3))

db <- dbeta_new(x=0.5, mu=mean_PPd, v=variance_PPd)
shape1 <- attributes(db)$shape1
shape2 <- attributes(db)$shape2

Priors <- setPriors1(Name = "pa",
                   Parameters=c(shape1=1, shape2=1),
                   SDProp = 0.1,
                   Density = "dbeta",
                   Init = n_season/s_season,
                   Min=0, Max=1
)
Priors <- Priors + setPriors1(Name="CF", Init = 3,
                          Parameters = c(mean=7, sd=3),
                          Min=1, Max=15, Density = "dnorm",
                          SDProp = 1)

outmcmc <- MHalgoGen(parameters=Priors, data=dOCFd, likelihood=LprobP,
                    n.iter = 500000, n.adapt = 50000, thin = 10, trace=1000)

plot(outmcmc, what = "markovchain", parameters = "pa", legend="bottomleft", ylim=c(0, 1))
plot(outmcmc, what = "markovchain", parameters = "CF")

plot(outmcmc, what = "posterior", parameters = "pa")
plot(outmcmc, what = "posterior", parameters = "CF")

plot(as.numeric(outmcmc$resultMCMC$`1`[, "pa"]),
     as.numeric(outmcmc$resultMCMC$`1`[, "CF"]), pch=".", cex=3,
     las=1, xlab="p", ylab="CF", bty="n", ylim=c(1, 15), xlim=c(0, 1), yaxt="n")
axis(2, 1:15, las=1)
cor(as.numeric(outmcmc$resultMCMC$`1`[, "pa"]), as.numeric(outmcmc$resultMCMC$`1`[, "CF"]))

```

```

plot(outmcmc, what = "Ln1")
as.parameters(outmcmc, index = "quantile")

svg(filename = "Figure 2_b.svg", width = 12, height = 4)
layout(mat = matrix(1:3, ncol=3))
par(mar=c(4, 4, 1, 1))
plot(outmcmc, what = "posterior", parameters = "pa", # legend="topright",
      legend = c(x=ScalePreviousPlot(x=0.75, y=0.95)$x, y=ScalePreviousPlot(x=0.75, y=0.95)$y),
      xlim=c(0, 1),
      breaks = seq(from=0, to=1, by=0.01))
text(x=ScalePreviousPlot(x=0.05, y=0.95)$x, y=ScalePreviousPlot(x=0, y=1)$y, "A", cex=2)
plot(outmcmc, what = "posterior", parameters = "CF",
      legend = c(x=ScalePreviousPlot(x=0.75, y=0.95)$x, y=ScalePreviousPlot(x=0.75, y=0.95)$y),
      breaks = seq(from=1, to=15, by=0.1), xlim=c(1, 15), xaxt="n", ylim=c(0, 0.2))
axis(1, 1:15)
text(x=ScalePreviousPlot(x=0.05, y=0.95)$x, y=ScalePreviousPlot(x=0, y=1)$y, "B", cex=2)
plot(as.numeric(outmcmc$resultMCMC$`1`[, "pa"]),
      as.numeric(outmcmc$resultMCMC$`1`[, "CF"]), pch=".", cex=3,
      las=1, xlab="pa", ylab="CF", bty="n", ylim=c(1, 15), xlim=c(0, 1), yaxt="n")
axis(2, 1:15, las=1)
text(x=0.6, y=7, labels=paste0("r = ",
                                specify_decimal(cor(as.numeric(outmcmc$resultMCMC$`1`[, "pa"]),
                                                    as.numeric(outmcmc$resultMCMC$`1`[, "CF"])), 2)))
text(x=ScalePreviousPlot(x=0.05, y=0.95)$x, y=ScalePreviousPlot(x=0, y=1)$y, "C", cex=2)
dev.off()

# Fit using a model of ECF OCF using p

# p is the product of capture probability, and the probability that females
# use that beach, p, or p1 and p2 - if px is high, females are resident and if it
# low, it is transient females.

## One category of individuals

o_mu1p1_CFP <- fitCF(x = c(mu = 2.1653229641404539,
                           sd = 1.1465246643327098,
                           p = 0.25785366120357966),
                    fixed.parameters=NULL,
                    data=ECFOCF_2002, hessian = TRUE)

Priors <- fitCF_MHmcmc_p(result = o_mu1p1_CFP, density = "dunif", accept = TRUE)

outmcmc_mu1p1_CFP <- fitCF_MHmcmc(
  result = o_mu1p1_CFP,
  n.iter = 10000,
  parametersMCMC = Priors,
  n.chains = 1, WAIC = TRUE,
  n.adapt = 1000,
  thin = 10, trace=1000, adaptive = TRUE
)

## Two categories of individuals

```

```

o_mu1p2 <- fitCF(x = c(mu = 5.3628701816871462,
                      sd = 0.39390555498088764,
                      p1 = 0.61159637544418755,
                      p2 = -2.4212753004659189,
                      OTN = 0.31898004668901009),
                data=ECFOCF_2002, hessian = TRUE)

plot(o_mu1p2, result="CF", las=1, bty="n")
plot(o_mu1p2, result="prob")

Priors <- fitCF_MHmcmc_p(result = o_mu1p2, density = "dunif", accept = TRUE)

Priors["p1", c("Prior1", "Min")] <- 0
Priors["p2", c("Prior2", "Max")] <- 0

outmcmc_mu1p2 <- fitCF_MHmcmc(
  result = o_mu1p2,
  n.iter = 10000,
  parametersMCMC = Priors,
  n.chains = 1, WAIC = TRUE,
  n.adapt = 1000,
  thin = 10, trace=1000, adaptive = TRUE
)

## Compare both models

### Using AICc

compare_AICc(Model_1_categories = o_mu1p1_CFP,
             Model_2_categories = o_mu1p2)

### Using WAIC

library(loo)

Model_2_categories <- outmcmc_mu1p2$WAIC
Model_1_categories <- outmcmc_mu1p1_CFP$WAIC

Model_2_categories <- Model_2_categories[, !apply(Model_2_categories,
                                                MARGIN = 2, FUN = function(x) all(x == 0))]
Model_1_categories <- Model_1_categories[, !apply(Model_1_categories,
                                                MARGIN = 2, FUN = function(x) all(x == 0))]

# apply(WAIC1, MARGIN = 2, FUN = function(x) length(unique(x))==1)

Model_1_categories_loo <- loo::loo(Model_1_categories)
Model_2_categories_loo <- loo::loo(Model_2_categories)

print(loo_compare(list(Model_1_categories=Model_1_categories_loo,
                      Model_2_categories=Model_2_categories_loo)), simplify = FALSE)

ELPDweight(list(Model_1_categories=Model_1_categories_loo,

```

```

Model_2_categories=Model_2_categories_loo))

# Plot Results of selected model

plot(outmcmc_mu1p2, what = "markovchain", parameters = "p1")
plot(outmcmc_mu1p2, what = "markovchain", parameters = "p2")
plot(outmcmc_mu1p2, what = "markovchain", parameters = "OTN")
plot(outmcmc_mu1p2, what = "markovchain", parameters = "mu")
plot(outmcmc_mu1p2, what = "markovchain", parameters = "sd")

layout(mat = matrix(1:2, ncol=2))
par(mar=c(4, 4, 1, 1))
plot(o_mu1p2, resultMCMC = outmcmc_mu1p2, result = "cf", main="")
text(x=ScalePreviousPlot(x=0.05, y=0.95)$x, y=ScalePreviousPlot(x=0, y=1)$y, "A", cex=2)
plot(x=o_mu1p2, resultMCMC = outmcmc_mu1p2, result = "probabilities",
     main="", parameters=c("p1", "p2", "OTN1", "OTN2"),
     legend=c("p1", "p2", "Prob. Cat 1", "Prob. Cat 2"), xlab="Probabilities")
text(x=ScalePreviousPlot(x=0.05, y=0.95)$x, y=ScalePreviousPlot(x=0, y=1)$y, "B", cex=2)

# Fit using a model of ECF OCF using p and a

# a is capture probability, and p is the probability that females
# use that beach, p, or p1 and p2 - if px is high, females are resident and if it
# low, it is transient females.
# a is set up from the fraction of nesting turtles known by phenology
# that nest during dates with patrols

## One category of individuals

o_mu1p1_a <- fitCF(x = c(mu = 2.1653229641404539,
                        sd = 1.1465246643327098,
                        p = 0.25785366120357966),
                  fixed.parameters=c(a = -logit(mean_PPd)),
                  data=ECFOCF_2002, hessian = FALSE)

Priors <- fitCF_MHmcmc_p(result = o_mu1p1_a, density = "dunif", accept = TRUE)

outmcmc_mu1p1_a <- fitCF_MHmcmc(
  result = o_mu1p1_a,
  n.iter = 10000,
  parametersMCMC = Priors,
  n.chains = 1, WAIC = TRUE,
  n.adapt = 1000,
  thin = 10, trace=1000, adaptive = TRUE)

## Two categories of individuals

o_mu1p2_a <- fitCF(x = c('mu' = 5.3628715952689952,
                        'sd' = 0.39390925142405148,
                        'p1' = 0.61159618691939643,
                        'p2' = -2.4212738230438715,
                        'OTN' = 0.318979364987661),
                  fixed.parameters = c(a = -logit(mean_PPd)),

```

```

      data=ECFOCF_2002, hessian = FALSE,
      method = "Nelder-Mead",
      itnmax=500,
      control = list(trace = 1, REPORT = 100, maxit = 500))

plot(o_mu1p2_a, result="CF", las=1, bty="n")
plot(o_mu1p2_a, result="probabilities")

Priors <- fitCF_MHmcmc_p(result = o_mu1p2_a,
                        density = "dunif",
                        accept = TRUE)

# Ensure that Category 1 and 2 will not change of definition during fit
Priors["p1", c("Prior1", "Min")] <- 0
Priors["p2", c("Prior2", "Max")] <- 0

outmcmc_mu1p2_a <- fitCF_MHmcmc(
  result = o_mu1p2_a,
  n.iter = 10000,
  parametersMCMC = Priors,
  n.chains = 1, WAIC = TRUE,
  n.adapt = 1000,
  thin = 10, trace=1000, adaptive = TRUE)

## Compare both models

### Using AICc

compare_AICc(Model_1_categories = o_mu1p1_CfP,
             Model_2_categories = o_mu1p2)

### Using ELPD

library(loo)

Model_2_categories <- outmcmc_mu1p2_a$WAIC
Model_1_categories <- outmcmc_mu1p1_a$WAIC

Model_2_categories <- Model_2_categories[, !apply(Model_2_categories,
  MARGIN = 2, FUN = function(x) all(x == 0))]
Model_1_categories <- Model_1_categories[, !apply(Model_1_categories,
  MARGIN = 2, FUN = function(x) all(x == 0))]

# apply(WAIC1, MARGIN = 2, FUN = function(x) length(unique(x))==1)

Model_1_categories_loo <- loo::loo(Model_1_categories)
Model_2_categories_loo <- loo::loo(Model_2_categories)

print(loo_compare(list(Model_1_categories=Model_1_categories_loo,
                      Model_2_categories=Model_2_categories_loo)), simplify = FALSE)

ELPDweight(list(Model_1_categories=Model_1_categories_loo,
                Model_2_categories=Model_2_categories_loo))

```

```

MaxNests <- 15

plot(outmcmc_mu1p2_a, what = "markovchain", parameters = "p1")
plot(outmcmc_mu1p2_a, what = "markovchain", parameters = "p2")
plot(outmcmc_mu1p2_a, what = "markovchain", parameters = "OTN")
plot(outmcmc_mu1p2_a, what = "markovchain", parameters = "mu")
plot(outmcmc_mu1p2_a, what = "markovchain", parameters = "sd")

plot(o_mu1p2_a, resultMCMC = outmcmc_mu1p2_a, result="CF", las=1, bty="n", main="")
plot(o_mu1p2_a, resultMCMC = outmcmc_mu1p2_a, result="probabilities",
     parameters = c("p1", "p2", "OTN1", "OTN2"),
     legend = c("p1", "p2", "Prob.\nCat 1", "Prob.\nCat 2"),
     main="", xlab="Probabilities")

# Je dois sortir la distribution du CF moyen

x <- 1:MaxNests
y_tot <- NULL
for (j in 1:nrow(outmcmc_mu1p2_a$resultMCMC$`1`)) {
  mu <- outmcmc_mu1p2_a$resultMCMC$`1`[j, "mu"]
  sd <- outmcmc_mu1p2_a$resultMCMC$`1`[j, "sd"]
  p1 <- invlogit(-outmcmc_mu1p2_a$resultMCMC$`1`[j, "p1"])
  p2 <- invlogit(-outmcmc_mu1p2_a$resultMCMC$`1`[j, "p2"])
  OTN <- invlogit(-outmcmc_mu1p2_a$resultMCMC$`1`[j, "OTN"])

  y <- dlnorm(x, meanlog=log(abs(mu)), sdlog=abs(sd))
  y <- sum((y / sum(y)) * x)
  y <- y * OTN * p1 + y * (1-OTN) * p2
  y_tot <- c(y_tot, y)
}

# Nombre moyen de nids par femelle
quantile(y_tot)

load(file=file.path("dataOut", "result_Nests_Yal2002_f.Rdata"))
load(file=file.path("dataOut", "result_Nests_Yal2002_mcmc.Rdata"))

dates <- seq(from=as.Date("2002-01-01"), to = as.Date("2002-12-31"), by="1 day")
dates_season <- which(dates == "2002-03-01"):which(dates == "2002-08-01")

NbNests_2002 <- NULL
for (j in 1:nrow(result_Nests_Yal2002_mcmc$resultMCMC$`1`)) {
  # The same but using all replicates
  par <- c(result_Nests_Yal2002_mcmc$resultMCMC$`1`[j, ],
           result_Nests_Yal2002_f$fixed.parameters)
  par <- getFromNamespace(".format_par", ns="phenology")(xpar=par, serie="Site")
  countd <- getFromNamespace(".daily_count", ns="phenology")(d=0:364, xpar=par)
  countd[result_Nests_Yal2002_f$data$Site$ordinal] <- result_Nests_Yal2002_f$data$Site$nombre
  NbNests_2002 <- c(NbNests_2002, sum(countd[dates_season]))
}

N2002 <- NbNests_2002[seq(from=1, to=20000, length.out=1000)]/y_tot

```

```

quantile(N2002, probs=c(0.025, 0.5, 0.975))

N2002_residente <- N2002*invlogit(-outmcmc_mu1p2_a$resultMCMC$`1`[, "OTN"])
quantile(N2002_residente, probs=c(0.025, 0.5, 0.975))

save(N2002_residente, file=file.path("dataOut", "N2002_residente.Rdata"))

N2002_gamma <- fitdist(N2002, distr = "gamma")
N2002_lnorm <- fitdist(N2002, distr = "lnorm")

AIC(N2002_gamma)
AIC(N2002_lnorm)

plot(N2002_lnorm)

hist(N2002, xlab=c(950, 1350), freq=FALSE)

lines(x = seq(from=950, to=1350, by=1), dlnorm(x=seq(from=950, to=1350, by=1),
  meanlog = N2002_lnorm$estimate["meanlog"], sdlog=N2002_lnorm$estimate["sdlog"]), col="red")

hist(N2002_residente, freq = FALSE)

layout(mat = matrix(1:4, ncol=2))
par(mar=c(4, 4, 1, 1))
plot(o_mu1p2_a, resultMCMC = outmcmc_mu1p2_a, result="CF", las=1, bty="n", main="")
text(x=ScalePreviousPlot(x=0.05, y=0.95)$x, y=ScalePreviousPlot(x=0, y=1)$y, "A", cex=2)
plot(o_mu1p2_a, resultMCMC = outmcmc_mu1p2_a, result="probabilities",
  parameters = c("p1", "p2", "OTN1", "OTN2"),
  legend = c("p1", "p2", "Proportion\nCat 1", "Proportion\nCat 2"), main="",
  xlab="", ylab="Probability")
text(x=ScalePreviousPlot(x=0.05, y=0.95)$x, y=ScalePreviousPlot(x=0, y=1)$y, "B", cex=2)
par(mar=c(4, 4, 1, 1))
plot(1:1000, NbNests[seq(from=1, to=20000, length.out=1000)]/y_tot, type="l", las=1,
  bty="n", ylim=c(0, 1400), yaxt="n", xlab="MCMC iterations", ylab="Number of females")
lines(x = 1:1000, y=N2002_residente, col="blue")
text(x=800, y=300, labels="Resident", col="blue")
text(x=800, y=900, labels="Total", col="black")
axis(2, at=seq(from=0, to=1400, by=100), las=1)
text(x=ScalePreviousPlot(x=0.05, y=0.95)$x, y=ScalePreviousPlot(x=0, y=1)$y, "C", cex=2)
par(mar=c(4, 5, 1, 1))
hist(N2002, xlim=c(400, 1400), las=1,
  freq = FALSE, main="", xaxt="n", xlab="Number of females", ylab="", ylim=c(0, 0.025),
  breaks = seq(from=400, to=1400, by=100))

N2002_lnorm <- fitdist(N2002, distr = "lnorm")
lines(x = seq(from=950, to=1350, by=1), dlnorm(x=seq(from=950, to=1350, by=1),
  meanlog = N2002_lnorm$estimate["meanlog"], sdlog=N2002_lnorm$estimate["sdlog"]),
  col="black", lwd=2)

text(x=1150, y=0.0125, labels="Total")

par(new=TRUE)
hist(N2002_residente, xlim=c(400, 1400), axes = FALSE,

```

```

      freq = FALSE, main="", xaxt="n", xlab="", ylab="", ylim=c(0, 0.025),
      breaks = seq(from=400, to=1400, by=10), col="blue")
text(x=500, y=0.0225, labels="Resident", col="blue")

N2002_residente_lnorm <- fitdist(N2002_residente, distr = "lnorm")
lines(x = seq(from=400, to=800, by=1), dlnorm(x=seq(from=400, to=800, by=1),
  meanlog = N2002_residente_lnorm$estimate["meanlog"],
  sdlog=N2002_residente_lnorm$estimate["sdlog"]), lwd=2, col="blue")

mtext("Density", side=2, line=4)
axis(1, at=seq(from=400, to=1400, by=50))
text(x=ScalePreviousPlot(x=0.05, y=0.95)$x, y=ScalePreviousPlot(x=0, y=1)$y, "D", cex=2)

# Example to analyze a trend

# Generate timeseries of nest counts

library(nlme)

set.seed(1)
years <- (1996:2100)
data <- data.frame(Year=years, Number=NA, time=1:length(years), ID=factor(1))
rownames(data) <- as.character(years)

data[1:4, "Number"] <- abs(rnorm(4, mean=1000, sd=100))

for (i in 5:nrow(data))
  data[i, "Number"] <- data[i-1, "Number"]*invlogit(rnorm(1, mean=logit(0.1), sd=0.1))+
    data[i-2, "Number"]*invlogit(rnorm(1, mean=logit(0.5), sd=0.5))+
    data[i-3, "Number"]*invlogit(rnorm(1, mean=logit(0.4), sd=0.3))

data <- cbind(data, meanlog=log(data$Number), sd=rlnorm(nrow(data), mean=log(200), sd=0.2), n=1000)

plot(data$Year, data$Number)

years <- 2020:2069

data_matrix <- matrix(data = NA, ncol=length(years), nrow=1000)
colnames(data_matrix) <- as.character(years)

for (y in as.character(years)) {
  data_matrix[, y] <- abs(rnorm(1000, mean = data[y, "Number"], sd = data[y, "sd"]))
}

data <- data[as.character(years), ]
plot(data$Year, data$Number, ylim=c(0, 2000), las=1, bty="n")

## Method 1

phis <- NULL
predict_r <- matrix(data = NA, nrow = 1000, ncol=length(years))

```

```

for (i in 1:nrow(data_matrix)) {
  print(i)
  data_matrix_r <- as.data.frame(data_matrix[i, ])
  colnames(data_matrix_r) <- "Number"
  data_matrix_r <- cbind(Year=as.numeric(rownames(data_matrix_r)), data_matrix_r,
    time=1:nrow(data_matrix_r), ID=rep(factor(1), nrow(data_matrix_r)))

  model_ar0ma1 <- lme(
    fixed = Number ~ poly(Year, 2),
    random = ~ 1 | ID,
    correlation = corAR1(value=0.5, form = ~ time | ID),
    data = data_matrix_r,
    method = "ML"
  )
  phis <- c(phis, coef(model_ar0ma1$modelStruct$corStruct, unconstrained = FALSE))
  predict_r[i, ] <- model_ar0ma1$fitted[, "fixed"]
}

hist(phis)

plot(x=0, y=0, xlim=range(as.numeric(colnames(data_matrix))), ylim=c(0, 2000),
  type="n", las=1, bty="n", xlab="Years", ylab="Posterior distribution for N", xaxt="n")
axis(1, years)

for (i in 1:nrow(predict_r))
  lines(as.numeric(years), predict_r[i, ], col=rgb(red=0.90, blue=0.90, green = 0.90, alpha = 0.1))

for (y in as.character(years)) {
  points(rep(as.numeric(y), 1000), data_matrix[, y], pch=".")
}

text(x=2020, y=500, paste0("Increasing probability is ",
  specify_decimal(100*sum(predict_r[, 1] < predict_r[, 20]) /
    nrow(predict_r), 2), "%"), pos=4)
text(x=2020, y=300, paste0("Decreasing probability is ",
  specify_decimal(100-100*sum(predict_r[, 1] < predict_r[, 20]) /
    nrow(predict_r), 2), "%"), pos=4)

predict_rq <- apply(predict_r, MARGIN = 2, FUN=function(x) quantile(x, probs=c(0.025, 0.5, 0.975)))
lines(years, predict_rq["50%", ], lty=1, col="red", lwd=2)
lines(years, predict_rq["2.5%", ], lty=2, col="red", lwd=2)
lines(years, predict_rq["97.5%", ], lty=2, col="red", lwd=2)

## Method 2

i <- 1
data_matrix_r <- as.data.frame(data_matrix[i, ])
colnames(data_matrix_r) <- "Number"
data_matrix_r_tot <- cbind(Year=as.numeric(rownames(data_matrix_r)),
  data_matrix_r, time=1:nrow(data_matrix_r),
  ID=rep(factor(1), nrow(data_matrix_r)))

```

```

for (i in 2:nrow(predict_r)) {
  data_matrix_r <- as.data.frame(data_matrix[i, ])
  colnames(data_matrix_r) <- "Number"
  data_matrix_r_tot <- rbind(data_matrix_r_tot, cbind(Year=as.numeric(rownames(data_matrix_r)),
                                                    data_matrix_r, time=1:nrow(data_matrix_r), ID=rep(factor(i),
                                                    nrow(data_matrix_r))))
}

library(brms)

model_ar1 <- brm(
  bf(
    Number | se(sd / sqrt(n), sigma = TRUE) ~ poly(time, 2),
    autocor = cor_ar(~ time, p = 1, cov = TRUE)
  ),
  data = data,
  family = gaussian(),
  chains = 4,
  cores = 4
)

summary(model_ar1)
plot(model_ar1)

k <- rstantools::posterior_predict(
  object=model_ar1)

layout(matrix(1:2, ncol=1))

par(mar=c(4, 4, 1, 1))
plot(x=0, y=0, xlim=range(as.numeric(colnames(data_matrix))), ylim=c(0, 2000),
     type="n", las=1, bty="n", xlab="Years",
     ylab="Posterior distribution for N", xaxt="n")
axis(1, years)

for (y in as.character(years)) {
  points(rep(as.numeric(y), 1000), data_matrix[, y], pch=19,
          col=rgb(red = 0, blue=0, green = 0, alpha = 0.005))
}

text(x=2020, y=300, paste0("Increasing probability is ",
  specify_decimal(100*sum(predict_r[, 1] < predict_r[, ncol(predict_r)]) /
  nrow(predict_r), 2), "%"), pos=4)
text(x=2020, y=100, paste0("Decreasing probability is ",
  specify_decimal(100-100*sum(predict_r[, 1] < predict_r[, ncol(predict_r)]) /
  nrow(predict_r), 2), "%"), pos=4)

predict_rq <- apply(predict_r, MARGIN = 2, FUN=function(x)
  quantile(x, probs=c(0.025, 0.5, 0.975)))
lines(years, predict_rq["50%", ], lty=1, col="red", lwd=2)

```

```

lines(years, predict_rq["2.5%", ], lty=2, col="red", lwd=2)
lines(years, predict_rq["97.5%", ], lty=2, col="red", lwd=2)

par(xpd=TRUE)
text(x=ScalePreviousPlot(x=0, y=1)$x, y=ScalePreviousPlot(x=0, y=1)$y,
labels = "A", cex=2, pos=4)

par(mar=c(4, 4, 1, 1))
plot_errbar(x=data$Year, y=data$Number,
            errbar.y = 1.96*data$sd,
            errbar.tick = 1/100,
            xlim=range(as.numeric(colnames(data_matrix))), ylim=c(0, 2000),
            type="n", las=1, bty="n", xlab="Years",
            ylab="Posterior distribution for N", xaxt="n")
axis(1, years)

for (y in as.character(years)) {
  points(rep(as.numeric(y), 1000), data_matrix[, y], pch=".")
}

kq <- apply(k, MARGIN = 2, FUN=function(x) quantile(x, probs=c(0.025, 0.5, 0.975)))
lines(years, kq["50%", ], lty=1, col="red", lwd=2)
lines(years, kq["2.5%", ], lty=2, col="red", lwd=2)
lines(years, kq["97.5%", ], lty=2, col="red", lwd=2)

text(x=2020, y=300, paste0("Increasing probability is ",
                           specify_decimal(100*sum(k[, 1] < k[, ncol(k)]) /
                           nrow(k), 2), "%"), pos=4)
text(x=2020, y=100, paste0("Decreasing probability is ",
                           specify_decimal(100-100*sum(k[, 1] < k[, ncol(k)]) /
                           nrow(k), 2), "%"), pos=4)

par(xpd=TRUE)
text(x=ScalePreviousPlot(x=0, y=1)$x, y=ScalePreviousPlot(x=0, y=1)$y,
      labels = "B", cex=2, pos=4)

#'
## End(Not run)

```

shift_sinusoid

Shift sinusoid information.

Description

This function is used to shift sinusoid parameters from ", '1' or '2'.

Usage

```
shift_sinusoid(parameters = NULL, from = "", to = "1")
```

Arguments

parameters	set of parameters
from	The number of series to change
to	The number of series to change

Details

shift_sinusoid shift sinusoid information.

Value

Return a set of modified parameters

Author(s)

Marc Girondot

Examples

```
# Read a file with data
library("phenology")
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot<-add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Fix parameter FFlat to 0
pfixed=c(Flat=0)
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=pfixed)
# Fit is done
## Not run:
result_Gratiot_Flat<-fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=pfixed)

## End(Not run)
data(result_Gratiot_Flat)
parg<-extract_result(result_Gratiot_Flat)
# Add data for one sinusoid superimposed
# [day d amplitude=(Alpha+Nd*Beta)^Tau with Nd being the number of counts for day d]
parg<-c(parg, Alpha=0.5, Beta=0.8, Delta=3, Phi=15)
# Tau is fixed to 1
pfixed=c(Flat=0, Tau=1)
# Run the optimisation
## Not run:
result_Gratiot1<-fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=pfixed)
# Plot the phenology
```

```

output1<-plot(result_Gratiot1, moon=TRUE)
#'
## End(Not run)
data(result_Gratiot1)
# Extract the fitted parameters
parg1<-extract_result(result_Gratiot1)
# Shift sinusoid information to the '1'
parg2<-shift_sinusoid(parameters=parg1, from="", to="1")
# Tau is fixed to 1
pfixed=c(Flat=0, Tau1=1, Tau=1)
# Add data for another sinusoid superimposed
# [day d amplitude=(Alpha+Nd*Beta)^Tau with Nd being the number of counts for day d]
parg<-c(parg2, Alpha=0.5, Beta=0.8, Delta=3, Phi=10)
# Run the optimisation
## Not run:
result_Gratiot2<-fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=pfixed)
# Plot the phenology
output2<-plot(result_Gratiot2, moon=TRUE)

## End(Not run)
data(result_Gratiot2)

```

summary.IP

Print the result information from a IP object.

Description

The function summary.IP shows result and estimates confidence interval.

Usage

```

## S3 method for class 'IP'
summary(object, ..., N = NULL, probs = c(0.025, 0.975))

```

Arguments

object	A file of class IP
...	Not used
N	Number of replicates
probs	Probability of confidence interval

Details

summary.IP prints the information from a IP object.

Value

Nothing

Author(s)

Marc Girondot

See Also

Other Model of Interesting Period: [IPFit\(\)](#), [IPModel\(\)](#), [IPPredict\(\)](#), [plot.IP\(\)](#)

Examples

```
## Not run:
library(phenology)
# Read a file with data

## End(Not run)
```

summary.phenology	<i>Print the result information from a result object.</i>
-------------------	---

Description

The function summary.phenology shows result and estimates confidence interval.

If several years are analyzed, the sum_synthesis result can be obtained only if there is not a mix of bisextile and non-bisextile years.

If save.all is true, it will return a object \$save.all as an array with dimensions:

```
c("without_obs_ML", "with_obs_ML", "without_obs_MCMC", "with_obs_MCMC"),
series,
replicates
```

The replicates dimension will be the max of replicate.CI.mcmc and replicate.CI. If they are not equal, NA will be present for the missing replicates. See examples.

Usage

```
## S3 method for class 'phenology'
summary(
  object,
  resultmcmc = NULL,
  season = NULL,
  chain = 1,
  series = "all",
  replicate.CI.mcmc = "all",
  replicate.CI = 10000,
  level = 0.95,
  print = TRUE,
  save.all = FALSE,
  ...
)
```

Arguments

object	A result file generated by fit_phenology
resultmcmc	A mcmc object
season	The number of season to analyze
chain	The number of chain to be used in resultmcmc
series	Names of the series to be analyzed or "all"
replicate.CI.mcmc	Number of iterations to be used or "all"
replicate.CI	Number of replicates for ML resampling
level	Level to estimate confidence interval or credibility interval
print	Should information be shown
save.all	If TRUE, returns all the values for replicates
...	Not used

Details

summary.phenology prints the information from a result object.

Value

A list with five objects: synthesis is a data.frame with global estimate of nesting. details_MCMC, details_ML, details_mean are lists with day by day information for each series, and sum_synthesis is the synthesis of the sum of all analyzed time-series.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenologymap\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
## Not run:
library(phenology)
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot<-add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
```

```

# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
result_Gratiot<-fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=NULL)
data(result_Gratiot)
# Display information from the result
s <- summary(result_Gratiot)
# Using mcmc
s <- summary(object=result_Gratiot, resultmcmc=result_Gratiot_mcmc)
sa <- summary(object=result_Gratiot, resultmcmc=result_Gratiot_mcmc, save.all = TRUE)
hist(sa$save.all["with_obs_MCMC", "Complete", ])

## End(Not run)

```

summary.phenologymap *Print information on a phenologymap object.*

Description

summary.phenologymap print information on a phenologymap object

Usage

```
## S3 method for class 'phenologymap'
summary(object, ...)
```

Arguments

object	A map generated with map_phenology.
...	Not used

Value

Return None

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologyout\(\)](#)

Examples

```
library("phenology")
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot<-add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
## Not run:
result_Gratiot<-fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=NULL)

## End(Not run)
data(result_Gratiot)
# Extract the fitted parameters
parg1<-extract_result(result_Gratiot)
# Add constant Alpha and Tau values
# [day d amplitude=(Alpha+Nd*Beta)^Tau with Nd being the number of counts for day d]
pfixed<-c(parg1, Alpha=0, Tau=1)
pfixed<-pfixed[-which(names(pfixed)=="Theta")]
# The only fitted parameter will be Beta
parg2<-c(Beta=0.5, parg1["Theta"])
# Generate a likelihood map
# [default Phi=seq(from=0.1, to=20, length.out=100) but it is very long]
# Take care, it takes 20 hours ! The data map_Gratiot has the result
## Not run:
map_Gratiot<-map_phenology(data=data_Gratiot,
  Phi=seq(from=0.1, to=20, length.out=100),
  fitted.parameters=parg2, fixed.parameters=pfixed)

## End(Not run)
data(map_Gratiot)
# Print the information on a map
summary(map_Gratiot)
```

summary.phenologyout *Print the summary information from a output object.*

Description

The function summary.phenologyout displays the output from a plot.

Usage

```
## S3 method for class 'phenologyout'
summary(object, ...)
```

Arguments

object	An output generated by plot.phenology() ou summary.phenology()
...	Not used

Details

summary.phenologyout prints the information from a result object.

Value

None

Author(s)

Marc Girondot

See Also

Other Phenology model: [AutoFitPhenology\(\)](#), [BE_to_LBLE\(\)](#), [Gratiot](#), [LBLE_to_BE\(\)](#), [LBLE_to_L\(\)](#), [L_to_LBLE\(\)](#), [MarineTurtles_2002](#), [MinBMinE_to_Min\(\)](#), [adapt_parameters\(\)](#), [add_SE\(\)](#), [add_phenology\(\)](#), [extract_result\(\)](#), [fit_phenology\(\)](#), [likelihood_phenology\(\)](#), [logLik.phenology\(\)](#), [map_Gratiot](#), [map_phenology\(\)](#), [par_init\(\)](#), [phenology\(\)](#), [phenology2fitRMU\(\)](#), [phenology_MHmcmc\(\)](#), [phenology_MHmcmc_p\(\)](#), [plot.phenology\(\)](#), [plot.phenologydata\(\)](#), [plot.phenologymap\(\)](#), [plot_delta\(\)](#), [plot_phi\(\)](#), [print.phenology\(\)](#), [print.phenologymap\(\)](#), [print.phenologyout\(\)](#), [remove_site\(\)](#), [result_Gratiot](#), [result_Gratiot1](#), [result_Gratiot2](#), [result_Gratiot_Flat](#), [summary.phenology\(\)](#), [summary.phenologymap\(\)](#)

Examples

```
## Not run:
library(phenology)
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
data_Gratiot <- add_phenology(Gratiot, name="Complete",
  reference=as.Date("2001-01-01"), format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot, fixed.parameters=NULL)
# Run the optimisation
result_Gratiot <- fit_phenology(data=data_Gratiot,
  fitted.parameters=parg, fixed.parameters=NULL)
data(result_Gratiot)
# Plot the phenology and get some stats
output <- plot(result_Gratiot)
# Show the output
summary(output)

## End(Not run)
```

TableECFOCF

*Format a CMR dataset into a file that fitCF can use.***Description**

This function formats a CMR dataset to a file that fitCF can use.

If date0 is not null, a 3D TableECFOCF is generated.

3D table (ECF, OCF, period) has two attributes:

- table with 5 elements:
begin, end are the first and last elements with counts
final is the last period with information
min and max are the first and last period where a nest could have been laid based on MaxNests value
- characteristics with 5 elements:
MinimumDaysBetween2Nest, MeanDaysBetween2Nest MaxNests, date0, length_season
p parameter can be setup to +Inf until begin and after end

Usage

```
TableECFOCF(
  data = stop("A dataframe with a column 'ID' and a column 'Date'"),
  columnID = "ID",
  columnDate = "Date",
  MinimumDaysBetween2Nest = 7,
  MeanDaysBetween2Nest = 9.8,
  MaxNests = 15,
  date0 = NULL,
  length_season = floor(365/MeanDaysBetween2Nest) + 1
)
```

Arguments

data	CMR file.
columnID	Name of the column in data for unique identifier of females.
columnDate	Name of the column in data for morning date when female has been seen on the beach.
MinimumDaysBetween2Nest	Number of minimum days between two nests.
MeanDaysBetween2Nest	Average number of days between two nests.
MaxNests	Maximum number of nests by a female.
date0	Date for the ordinal day 0.
length_season	The total length of season based on groups of interclutch intervals.

Details

TableECFOCF formats a CMR dataset into a file that fitCF can use.

Value

Return a matrix with counts for all OCF and ECF combinations.

Author(s)

Marc Girondot

See Also

Other Model of Clutch Frequency: [ECFOCF_f\(\)](#), [ECFOCF_full\(\)](#), [fitCF\(\)](#), [fitCF_MHmcmc\(\)](#), [fitCF_MHmcmc_p\(\)](#), [generateCF\(\)](#), [lnLCF\(\)](#), [logLik.ECFOCF\(\)](#), [plot.ECFOCF\(\)](#), [plot.TableECFOCF\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
data(MarineTurtles_2002)
ECFOCF_2002 <- TableECFOCF(MarineTurtles_2002)
plot(ECFOCF_2002)
ECFOCF_2002 <- TableECFOCF(MarineTurtles_2002, date0=as.Date("2002-01-01"))
plot(ECFOCF_2002, period=11)

## End(Not run)
```

Tagloss_cumul

Return the cumulative rate of tag loss.

Description

This function compute a model of cumulative tag loss rate for days t based on a set of parameters, par.

If hessian is not null, it will estimate standard error of the output using numerical delta method is replicates is null or using resampling if replicates is not null.

Parameters are described in [Tagloss_fit](#).

Usage

```
Tagloss_cumul(
  t,
  par = NULL,
  Hessian = NULL,
  mcmc = NULL,
  method = NULL,
```

```

    model_before = NULL,
    model_after = NULL,
    model = NULL,
    replicates = NULL,
    x = NULL
  )

```

Arguments

t	Time for which values of model must be estimated
par	Parameters
Hessian	Hessian matrix of parameters
mcmc	A mcmc result
method	Can be NULL, "delta", "SE", "Hessian", "MCMC", or "PseudoHessianFromMCMC"
model_before	Function to be used before estimation of daily tagloss rate
model_after	Function to be used after estimation of daily tagloss rate
model	The model of parameter to use, can be N2, N1 or N0; or NLR, NL0, N0R, or N00 or NULL if hessian is NULL.
replicates	Number of replicates to estimate se of output for resampling method
x	A Tagloss fitted model

Details

Tagloss_cumul returns the cumulative rate of tag loss.

Value

Return the cumulative rate of tag loss if hessian is null or a data.frame with distribution of cumulative rate of tag loss if hessian is not null.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Model of Tag-loss: [Tagloss_L\(\)](#), [Tagloss_LengthObs\(\)](#), [Tagloss_daymax\(\)](#), [Tagloss_fit\(\)](#), [Tagloss_format\(\)](#), [Tagloss_mcmc\(\)](#), [Tagloss_mcmc_p\(\)](#), [Tagloss_model\(\)](#), [Tagloss_simulate\(\)](#), [logLik.Tagloss\(\)](#), [o_4p_p1p2](#), [plot.Tagloss\(\)](#), [plot.TaglossData\(\)](#)

Examples

```

## Not run:
# Example
library(phenology)

```

```

# Data from Rivalan et al. 2005 - Table 2, line 1 - Fig 1D
par <- c(a0_2=-5.43E-2, a1_2=-103.52, a2_2=0, a3_2=0, a4_2=5.62E-4)
(y <- Tagloss_cumul(t=(1:6)*365, par=par))
plot(y[, "time"], y[, "N2"], type="l", bty="n",
      xlab="Days after tagging", ylab="N2 proportion")

# Data from Rivalan et al. 2005 - Table 2, line 2 - Fig 1E
par <- c(a0_2=-6.80E-2, a1_2=-81.15, a2_2=-2.20E-4, a3_2=6348.01, a4_2=1.65E-3)
(y <- Tagloss_cumul(t=(1:6)*365, par=par))
plot(y[, "time"], y[, "N2"], type="l",
      xlab="Days after tagging", ylab="N2 proportion")

# Data from Rivalan et al. 2005 - Table 2, line 3 - Fig 1F
par <- c(a0_2=-6.93E-2, a1_2=-78.92, a2_2=8.45E-4, a3_2=-16272.76, a4_2=2.87E-4)
(y <- Tagloss_cumul(t=(1:6)*365, par=par))
plot(y[, "time"], y[, "N2"], type="l",
      xlab="Days after tagging", ylab="N2 proportion")

# Data from Rivalan et al. 2005 - Table 2, line 4 - Fig 1C
par <- c(a0_2=-1.68E-3, a1_2=-4141.68, a2_2=0, a3_2=0, a4_2=0)
(y <- Tagloss_cumul(t=(1:6)*365, par=par))
plot(y[, "time"], y[, "N2"], type="l",
      xlab="Days after tagging", ylab="N2 proportion")

# Data from Rivalan et al. 2005 - Table 2, line 5 - Fig 1B
par <- c(a0_2=-3.77E-4, a1_2=-2000, a2_2=-0.001, a3_2=0, a4_2=5.00E-8)
(y <- Tagloss_cumul(t=(1:6)*365, par=par))
plot(y[, "time"], y[, "N2"], type="l",
      xlab="Days after tagging", ylab="N2 proportion")

# Data from Rivalan et al. 2005 - Table 2, line 6 - Fig 1A
par <- c(a0_2=-1E5, a1_2=-2000, a2_2=0, a3_2=4000, a4_2=8.34E-4)
(y <- Tagloss_cumul(t=(1:6)*365, par=par))
plot(y[, "time"], y[, "N2"], type="l",
      xlab="Days after tagging", ylab="N2 proportion")

# Data from Rivalan et al. 2005 - Table 2, line 1 - Fig 1D
# With tagloss rate dependency on tage number
par <- c(a0_2=-5.43E-2, a1_2=-103.52, a2_2=0, a3_2=0, a4_2=5.62E-4,
        a0_1=-5.43E-2, a1_1=-103.52, a2_1=0, a3_1=0, a4_1=5.62E-4, delta_1=3.2E-4)
phenology::plot.Tagloss(x=list(), t=1:1000, fitted.parameters=par,
                        model="Cumul",
                        decoration = TRUE)

p2 <- Tagloss_model(t=1:(6*365), par=par, model="2")
p1 <- Tagloss_model(t=1:(6*365), par=par, model="1")
par(mar=c(4, 5, 2, 1))
plot(x=1:(6*365), y=p2, bty="n", type="l", las=1, ylim=c(0,0.003), ylab="")
mtext("Daily tag loss", side=2, line=4)
lines(x=1:(6*365), y=p1, col="red")
legend("topright", legend=c("2>1", "1>0"), lty=1, col=c("black", "red"))

Tagloss_cumul(t=(1:6)*365, par=par)

```

```

# Without tagloss rate dependency on tag number
par <- c(a0_2=-5.43E-2, a1_2=-103.52, a2_2=0, a3_2=0, a4_2=5.62E-4,
        a0_1=-5.43E-2, a1_1=-103.52, a2_1=0, a3_1=0, a4_1=5.62E-4)
phenology:::plot.Tagloss(x=list(), t=1:1000, fitted.parameters=par,
                        model="Cumul",
                        decoration = TRUE)
Tagloss_cumul(t=(1:6)*365, par=par)

#### Data from Casale et al. 2017
# Table 1 - Model II
par <- c(CasaleModelIIa0_2=-0.0511, CasaleModelIIa1_2=-100, CasaleModelIIa4_2=0.00014)
phenology:::plot.Tagloss(x=list(), t=1:1000, fitted.parameters=par,
                        model="Cumul",
                        decoration = TRUE)
Tagloss_cumul(t=(1:6)*365, par=par)

# Table 1 - Model IV
par <- c(CasaleModelIVa0_2=-0.0132, CasaleModelIVa1_2=-100,
        CasaleModelIVa2_2=0.0327, CasaleModelIVa3_2=109.98,
        CasaleModelIVa4_2=0.00011)
phenology:::plot.Tagloss(x=list(), t=1:1000, fitted.parameters=par,
                        model="Cumul",
                        decoration = TRUE)
Tagloss_cumul(t=(1:6)*365, par=par)

# Table 1 - Model I
par <- c(CasaleModelIc_2=0.00027)
phenology:::plot.Tagloss(x=list(), t=1:1000, fitted.parameters=par,
                        model="Cumul",
                        decoration = TRUE)
Tagloss_cumul(t=(1:6)*365, par=par)

# Table 1 - Model III
par <- c(CasaleModelIIIa0_2=1.14E-10, CasaleModelIIIa1_2=-110.04,
        CasaleModelIIIa4_2=0.00055)
phenology:::plot.Tagloss(x=list(), t=1:1000, fitted.parameters=par,
                        model="Cumul",
                        decoration = TRUE)
Tagloss_cumul(t=(1:6)*365, par=par)

# Table 1 - Model V
par <- c(CasaleModelVa0_2=4.04E-10, CasaleModelVa1_2=-90,
        CasaleModelVa2_2=-0.0326, CasaleModelVa3_2=100.31,
        CasaleModelVa4_2=0.00006)
phenology:::plot.Tagloss(x=list(), t=1:1000, fitted.parameters=par,
                        model="Cumul",
                        decoration = TRUE)
Tagloss_cumul(t=(1:6)*365, par=par)

## End(Not run)

```

Tagloss_daymax	<i>Return the maximum number of days an individual has been observed in a dataset.</i>
----------------	--

Description

This function must be used to get the value of mx in Tagloss_L.

Usage

```
Tagloss_daymax(individuals, what = "max")
```

Arguments

individuals	Set of individuals
what	By default is max, but can be min, mean or all

Details

Tagloss_daymax returns the maximum number of days an individual has been observed in a dataset.

Value

Return the maximum number of days an individual has been observed in a dataset.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Model of Tag-loss: [Tagloss_L\(\)](#), [Tagloss_LengthObs\(\)](#), [Tagloss_cumul\(\)](#), [Tagloss_fit\(\)](#), [Tagloss_format\(\)](#), [Tagloss_mcmc\(\)](#), [Tagloss_mcmc_p\(\)](#), [Tagloss_model\(\)](#), [Tagloss_simulate\(\)](#), [logLik.Tagloss\(\)](#), [o_4p_p1p2](#), [plot.Tagloss\(\)](#), [plot.TaglossData\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
data_f_21 <- Tagloss_format(outLR, model="21")
daymax(data_f_21)

## End(Not run)
```

Tagloss_fit

*fit a model of tag loss using a CMR database.***Description**

This function fits a model of tag loss using a CMR database.

The names of parameters can be:

Model Pfaller et al. (2019):

Left tag lost when 2 are present D1_L2, D2D1_L2, D3D2_L2, A_L2, B_L2, C_L2, delta_L2

Right tag lost when 2 are present D1_R2, D2D1_R2, D3D2_R2, A_R2, B_R2, C_R2, delta_R2

Left tag lost when 1 is present D1_L1, D2D1_L1, D3D2_L1, A_L1, B_L1, C_L1, delta_L1

Right tag lost when 1 is present D1_R1, D2D1_R1, D3D2_R1, A_R1, B_R1, C_R1, delta_R1

One tag lost when 2 are present D1_2, D2D1_2, D3D2_2, A_2, B_2, C_2, delta_2

One tag lost when 1 is present D1_1, D2D1_1, D3D2_1, A_1, B_1, C_1, delta_1

pA, pB and pC are the daily probabilities of tag loss with $pA = -\text{logit}(A)$, $pB = -\text{logit}(B)$ and $pC = -\text{logit}(C)$

.

delta is used as: $p = p + \text{delta}$. Nothe that delta can be negative

Tag loss rate is pA at day 1

Tag loss rate changes gradually from pA to pB that is reached at day D1

Tag loss rate is pB from day D1 to day $D2 = D1 + D2D1$

Tag loss rate changes gradually from pB to pC that is reached at day $D3 = D2 + D3D2$

When parameters from Rivalan et al. (2005) are used:

One tag lost when 2 are present a0_2, a1_2, a2_2, a3_2, a4_2, delta_2

One tag lost when 1 is present a0_1, a1_1, a2_1, a3_1, a4_1, delta_1

When parameters from Casale et al. (2017) are used:

Model I

One tag lost when 2 are present CasaleModelIc_2

One tag lost when 1 is present CasaleModelIc_1

Model II

One tag lost when 2 are present CasaleModelIIa0_2, CasaleModelIIa1_2, CasaleModelIIa4_2

One tag lost when 1 is present CasaleModelIIa0_1, CasaleModelIIa1_1, CasaleModelIIa4_1

Model III

One tag lost when 2 are present CasaleModelIIIIa0_2, CasaleModelIIIIa1_2, CasaleModelIIIIa4_2

One tag lost when 1 is present CasaleModelIIIIa0_1, CasaleModelIIIIa1_1, CasaleModelIIIIa4_1

Model IV

One tag lost when 2 are present CasaleModelIVa0_2, CasaleModelIVa1_2, CasaleModelIVa2_2,
CasaleModelIVa3_2, CasaleModelIVa4_2

One tag lost when 1 is present CasaleModelIVa0_1, CasaleModelIVa1_1, CasaleModelIVa2_1,
CasaleModelIVa3_1, CasaleModelIVa4_1

Model V

One tag lost when 2 are present CasaleModelVa0_2, CasaleModelVa1_2, CasaleModelVa2_2,
CasaleModelVa3_2, CasaleModelVa4_2

One tag lost when 1 is present CasaleModelVa0_1, CasaleModelVa1_1, CasaleModelVa2_1, CasaleModelVa3_1,
CasaleModelVa4_1

If only one parameter is fitted, method must be "Brent" and upper and lower parameters must be set up with finite values.

model_before can be "'par['a0_1']=par['a0_2'];par['a1_1']=par['a1_2']". model_after can be "p1=p2"

Usage

```
Tagloss_fit(
  data = stop("A database formatted using Tagloss_format() must be used"),
  fitted.parameters = NULL,
  fixed.parameters = NULL,
  model_before = NULL,
  model_after = NULL,
  control = list(trace = 1, maxit = 10000),
  method = "Nelder-Mead",
  lower = -Inf,
  upper = Inf,
  hessian = FALSE,
  mc.cores = detectCores(all.tests = FALSE, logical = TRUE),
  groups = NULL
)
```

Arguments

data	An object formatted using Tagloss_format
fitted.parameters	Set of parameters to be fitted
fixed.parameters	Set of fixed parameters
model_before	Transformation of parameters before to use Tagloss_model()
model_after	Transformation of parameters after to use Tagloss_model()

control	Control parameters to be send to optim()
method	optim() method
lower	Lower value for parameter when Brent method is used
upper	Upper value for parameter when Brent method is used
hessian	Does the hessian matrix should be estimated
mc.cores	Number of cores to use for parallel computing
groups	Number of groups for parallel computing

Details

Tagloss_fit fits a model of tag loss using a CMR database.

Value

Return a list object with the model describing tag loss.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

References

Rivalan, P., Godfrey, M.H., Prévot-Julliard, A.-C., Girondot, M., 2005. Maximum likelihood estimates of tag loss in leatherback sea turtles. *Journal of Wildlife Management* 69, 540-548.

Casale, P., Freggi, D., Salvemini, P., 2017. Tag loss is a minor limiting factor in sea turtle tagging programs relying on distant tag returns: the case of Mediterranean loggerhead sea turtles. *European Journal of Wildlife Research* 63.

Pfaller JB, Williams KL, Frick MG, Shamblin BM, Nairn CJ, Girondot M (2019) Genetic determination of tag loss dynamics in nesting loggerhead turtles: A new chapter in “the tag loss problem”. *Marine Biology* 166: 97 doi 10.1007/s00227-019-3545-x

See Also

Other Model of Tag-loss: [Tagloss_L\(\)](#), [Tagloss_LengthObs\(\)](#), [Tagloss_cumul\(\)](#), [Tagloss_daymax\(\)](#), [Tagloss_format\(\)](#), [Tagloss_mcmc\(\)](#), [Tagloss_mcmc_p\(\)](#), [Tagloss_model\(\)](#), [Tagloss_simulate\(\)](#), [logLik.Tagloss\(\)](#), [o_4p_p1p2](#), [plot.Tagloss\(\)](#), [plot.TaglossData\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
data_f_21 <- Tagloss_format(outLR, model="21")

# model fitted by Rivalan et al. 2005
par <- c(a0_2=-5.43E-2, a1_2=-103.52, a4_2=5.62E-4,
        delta_1=3.2E-4)
pfixed <- c(a2_2=0, a3_2=0, a2_1=0, a3_1=0)
```

```

model_before <- "par['a0_1']=par['a0_2'];par['a1_1']=par['a1_2'];par['a4_1']=par['a4_2']"
o <- Tagloss_fit(data=data_f_21, fitted.parameters=par, fixed.parameters=pfixed,
                 model_before=model_before)
plot(o, t=1:1000, model="cumul")
plot(o, t=1:1000, model="1")
plot(o, t=1:1000, model="2", add=TRUE, col="red")

# Same data fitted with new model
par <- c(D1_1 = 100.15324837975547, A_1 = 5.9576927964120188,
        B_1 = 8.769924225871069, B_2 = 8.2353860179664125)
pfixed <- c(D2D1_1 = 2568, D3D2_1 = 2568, D2D1_2 = 2568, D3D2_2 = 2568)
o_4p_p1p2 <- Tagloss_fit(data=data_f_21, fitted.parameters = par,
                         fixed.parameters = pfixed,
                         model_before = "par['C_1']=par['B_1'];
                         par['A_2']=par['A_1'];
                         par['C_2']=par['B_2'];
                         par['D1_2']=par['D1_1']", hessian=TRUE)

# Without the N20 the computing is much faster
data_f_21_fast <- subset(data_f_21, subset=(is.na(data_f_21$N20)))
par <- c('D1_2' = 49.78891736351531,
        'D2D1_2' = 1059.3635769732305,
        'D3D2_2' = 12.434313273804602,
        'A_2' = 5.2238379144659683,
        'B_2' = 8.0050044071275543,
        'C_2' = 8.4317863609499675,
        'D1_1' = 701.80273287212935,
        'D2D1_1' = 0.010951749100596819,
        'D3D2_1' = 3773.6290607434876,
        'A_1' = 205.42435592344776,
        'B_1' = 9.9598342503239863,
        'C_1' = 6.7234868237164722)
o <- Tagloss_fit(data=data_f_21_fast, fitted.parameters=par, hessian = TRUE)
plot(o, model="1", col="red")
plot(o, model="2", col="blue", add=TRUE)
legend("topright", legend=c("2->1", "1->0"), lty=1, col=c("blue", "red"))

## End(Not run)

```

Tagloss_format

Format a CMR dataset into a file that Tagloss_L can use.

Description

This function formats a CMR dataset to a file that Tagloss_L can use.

The format of data is a data.frame with 4 columns:

ID is the column with the permanent identification code

L is the column with the non-permanent code located at left

R is the column with the non-permanent code located at right

Date is the column with the date of observation

Note that R and L columns can be exchanged if 21 model is used.

Usage

```
Tagloss_format(
  data,
  model = "21",
  progressbar = TRUE,
  column.Date = "Date",
  column.ID = "ID",
  column.left = "L",
  column.right = "R",
  keeponly2 = TRUE
)
```

Arguments

<code>data</code>	CMR file
<code>model</code>	Can be "21" or "LR"
<code>progressbar</code>	Is a progressbar been shown?
<code>column.Date</code>	Name of the column with Date
<code>column.ID</code>	Name of the column with ID
<code>column.left</code>	Name of the column with left tag
<code>column.right</code>	Name of the column with right tag
<code>keeponly2</code>	If TRUE, keep only individuals tagged with 2 tags

Details

Tagloss_format formats a CMR dataset into a file that Tagloss_L can use.

Value

Return the maximum number of days an individual has been observed in a dataset.

Author(s)

Marc Girondot

See Also

Other Model of Tag-loss: [Tagloss_L\(\)](#), [Tagloss_LengthObs\(\)](#), [Tagloss_cumul\(\)](#), [Tagloss_daymax\(\)](#), [Tagloss_fit\(\)](#), [Tagloss_mcmc\(\)](#), [Tagloss_mcmc_p\(\)](#), [Tagloss_model\(\)](#), [Tagloss_simulate\(\)](#), [logLik.Tagloss\(\)](#), [o_4p_p1p2](#), [plot.Tagloss\(\)](#), [plot.TaglossData\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
head(outLR)
```

```

data_f_21 <- Tagloss_format(outLR, model="21")
data_f_LR <- Tagloss_format(outLR, model="LR")

## End(Not run)

```

Tagloss_L	<i>Return the -log likelihood of a set of individuals under a model of tagloss.</i>
-----------	---

Description

This function must be used within `optim()`.
`model_before` is applied to the `par` parameter.
`model_after` is applied after `par` is separated in `p1`, `p2`, `pL1`, `pL2`, `pR1` and `pR2` parameters.
`progressbar` is set to `FALSE` if `mc.cores` is different from 1.
 If `days.maximum` is not indicated, it is estimated using `Tagloss_daymax()`.

Usage

```

Tagloss_L(
  individuals,
  par,
  days.maximum = NULL,
  fixed.parameters = NULL,
  model_before = NULL,
  model_after = NULL,
  names.par = NULL,
  groups = NULL,
  mc.cores = detectCores(all.tests = FALSE, logical = TRUE),
  progressbar = FALSE
)

```

Arguments

<code>individuals</code>	Set of individuals
<code>par</code>	Set of parameters
<code>days.maximum</code>	Maximum number of days. Can be determined using <code>Tagloss_daymax()</code>
<code>fixed.parameters</code>	Set of fixed parameters
<code>model_before</code>	Transformation of parameters before to use <code>Tagloss_model()</code>
<code>model_after</code>	Transformation of parameters after to use <code>Tagloss_model()</code>
<code>names.par</code>	Name of parameters. Normally unused.
<code>groups</code>	Number of groups for parallel computing
<code>mc.cores</code>	Number of cores to use for parallel computing
<code>progressbar</code>	Is shown a progressbar?

Details

Tagloss_L returns the -log likelihood of a set of individuals under a model of tagloss.

Value

Return the -log likelihood of a set of individuals

Author(s)

Marc Girondot

See Also

Other Model of Tag-loss: [Tagloss_LengthObs\(\)](#), [Tagloss_cumul\(\)](#), [Tagloss_daymax\(\)](#), [Tagloss_fit\(\)](#), [Tagloss_format\(\)](#), [Tagloss_mcmc\(\)](#), [Tagloss_mcmc_p\(\)](#), [Tagloss_model\(\)](#), [Tagloss_simulate\(\)](#), [logLik.Tagloss\(\)](#), [o_4p_p1p2](#), [plot.Tagloss\(\)](#), [plot.TaglossData\(\)](#)

Examples

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

# Example with 21 format of data

data_f_21 <- Tagloss_format(outLR, model="21")
par <- structure(c(49.5658922243074, 808.136085362158, 106.283783786853,
5.22150592456511, 8.00608716525864, 8.32718202233396, 150.612916258503,
715.865805125223, 2242.06574225966, 119.212383120678, 10.1860735529433,
7.14231725937626), names = c("D1_2", "D2D1_2", "D3D2_2", "A_2",
"B_2", "C_2", "D1_1", "D2D1_1", "D3D2_1", "A_1", "B_1", "C_1"))
pfixed <- NULL
# All the data are analyzed; the N20 are very long to compute
Tagloss_L(individuals=data_f_21, par=par, days.maximum=Tagloss_daymax(data_f_21),
fixed.parameters=pfixed, mc.cores=1, progressbar=TRUE)
# Without the N20 the computing is much faster
data_f_21_fast <- subset(data_f_21, subset=(is.na(data_f_21$N20)))
Tagloss_L(individuals=data_f_21_fast, par=par, days.maximum=Tagloss_daymax(data_f_21_fast),
fixed.par=pfixed, mc.cores=1, progressbar=TRUE)
o <- Tagloss_fit(data=data_f_21_fast, fitted.parameters=par)
# Here it is the result of the previous function
o <- structure(list(par = structure(c(49.5658922243074, 808.136085362158,
106.283783786853, 5.22150592456511, 8.00608716525864, 8.32718202233396,
150.612916258503, 715.865805125223, 2242.06574225966, 119.212383120678,
10.1860735529433, 7.14231725937626), names = c("D1_2", "D2D1_2",
"D3D2_2", "A_2", "B_2", "C_2", "D1_1", "D2D1_1", "D3D2_1", "A_1",
"B_1", "C_1")), value = 5841.93084262461, counts = structure(c(1093L,
NA), names = c("function", "gradient")), convergence = 0L, message = NULL,
hessian = structure(c(0.0469808583147824, 0.000133240973809734,
6.68478605803102e-05, -2.53581288234273, -1.25931342154217,
-0.124650568977813, -2.46700437855907e-05, -1.11413100967184e-05,
-3.18323145620525e-06, 0, -0.0182945996130002, -0.00510601694259094,
0.000133240973809734, 1.45519152283669e-05, 7.50333128962666e-06,
```

```

-0.00452587300969753, -0.0191316757991444, -0.0255117811320815,
-1.13686837721616e-06, -1.36424205265939e-06, -2.27373675443232e-07,
0, 0.000335830918629654, -0.000448608261649497, 6.68478605803102e-05,
7.50333128962666e-06, 4.32009983342141e-06, -0.00226373231271282,
-0.00954059942159802, -0.0127809016703395, -4.54747350886464e-07,
-4.54747350886464e-07, -2.27373675443232e-07, 0, 0.000176896719494835,
-0.000224190443987027, -2.53581288234273, -0.00452587300969753,
-0.00226373231271282, 223.422489398217, 41.4073996353181,
3.77875949197914, 0.000986460690910462, 0.000398813426727429,
0.000117665877041873, 0, 0.727547330825473, 0.194675862985605,
-1.25931342154217, -0.0191316757991444, -0.00954059942159802,
41.4073996353181, 189.534394394286, 28.3386068531399, 0.00216437001654413,
0.00241834641201422, 0.000652562448522076, 0, 0.841939595375152,
1.0472297162778, -0.124650568977813, -0.0255117811320815,
-0.0127809016703395, 3.77875949197914, 28.3386068531399,
70.250493081403, -0.00022441781766247, -0.000161662683240138,
0.000257614374277182, 0, -0.578908839088399, 1.08917492980254,
-2.46700437855907e-05, -1.13686837721616e-06, -4.54747350886464e-07,
0.000986460690910462, 0.00216437001654413, -0.00022441781766247,
0.000148247636388987, 0.000145519152283669, 3.97903932025656e-05,
0, 0.0156976511789253, 0.0678746800986119, -1.11413100967184e-05,
-1.36424205265939e-06, -4.54747350886464e-07, 0.000398813426727429,
0.00241834641201422, -0.000161662683240138, 0.000145519152283669,
0.000145519152283669, 3.9676706364844e-05, 0, 0.0138438736030366,
0.0678776359563926, -3.18323145620525e-06, -2.27373675443232e-07,
-2.27373675443232e-07, 0.000117665877041873, 0.000652562448522076,
0.000257614374277182, 3.97903932025656e-05, 3.9676706364844e-05,
1.77351466845721e-05, 0, 0.00317095327773131, 0.0316927071253303,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -0.0182945996130002,
0.000335830918629654, 0.000176896719494835, 0.727547330825473,
0.841939595375152, -0.578908839088399, 0.0156976511789253,
0.0138438736030366, 0.00317095327773131, 0, 8.85630879565724,
4.44044781033881, -0.00510601694259094, -0.000448608261649497,
-0.000224190443987027, 0.194675862985605, 1.0472297162778,
1.08917492980254, 0.0678746800986119, 0.0678776359563926,
0.0316927071253303, 0, 4.44044781033881, 88.8524673428037
), .Dim = c(12L, 12L), .Dimnames = list(c("D1_2", "D2D1_2",
"D3D2_2", "A_2", "B_2", "C_2", "D1_1", "D2D1_1", "D3D2_1",
"A_1", "B_1", "C_1"), c("D1_2", "D2D1_2", "D3D2_2", "A_2",
"B_2", "C_2", "D1_1", "D2D1_1", "D3D2_1", "A_1", "B_1", "C_1"
))), names = c("par", "value", "counts", "convergence",
"message", "hessian"), class = c("list", "Tagloss"))
par(mar=c(4, 4, 1, 1))
plot(o, t=1:3000, model="2", scale=1000, ylim=c(0, 3),
col="red")
plot(o, t=1500:3000, model="1", scale=1000,
add=TRUE)
legend("topright", legend=c("2 -> 1", "1 -> 0"), col=c("red", "black"), lty=1)

plot(o, t=1:300, model="2", scale=1000, ylim=c(0, 3),
col="red", hessian=o$hessian)
plot(o, t=1:300, model="1", scale=1000,
add=TRUE, hessian=o$hessian)

```

```

legend("topright", legend=c("2 -> 1", "1 -> 0"), col=c("red", "black"), lty=1)

##### Example with fixed.parameters

data_f_21 <- Tagloss_format(outLR, model="21")
# Without the N20 the computing is much faster
data_f_21_fast <- subset(data_f_21, subset=(is.na(data_f_21$N20)))
par <- structure(c(49.5658922243074, 5.22150592456511, 8.00608716525864,
  50.612916258503, 6, 9),
  names = c("D1_2", "A_2", "B_2",
    "D1_1", "A_1", "B_1"))
pfixed <- c(D2D1_2=10000, D3D2_2=10000, C_2=0, D2D1_1=10000, D3D2_1=10000, C_1=0)
o <- Tagloss_fit(data=data_f_21_fast, fitted.parameters=par, fixed.parameters=pfixed)
# Here it is the result of the previous function
o <- structure(list(par = structure(c(55.2184044121564, 5.2630294044259,
8.13359029885985, 14269.9757684677, 21.8702023948044, 6.46586480967269
), names = c("D1_2", "A_2", "B_2", "D1_1", "A_1", "B_1")), value = 5853.64634357369,
  counts = structure(c(757L, NA), names = c("function", "gradient"
  )), convergence = 0L, message = NULL, hessian = structure(c(0.036636720324168,
-2.26385645873961, -1.2330608569755, -2.95585778076202e-06,
-2.27373675443232e-07, -0.0399197688238928, -2.26385645873961,
232.345637869003, 47.1904784262733, 0.000118689058581367,
7.50333128962666e-06, 1.69928603099834, -1.2330608569755,
47.1904784262733, 304.432723851278, 0.000196678229258396,
1.36424205265939e-06, 2.8553522497532, -2.95585778076202e-06,
0.000118689058581367, 0.000196678229258396, 4.54747350886464e-07,
0, 0.00741636085876962, -2.27373675443232e-07, 7.50333128962666e-06,
1.36424205265939e-06, 0, 4.00177668780088e-05, 8.79936123965308e-05,
-0.0399197688238928, 1.69928603099834, 2.8553522497532, 0.00741636085876962,
8.79936123965308e-05, 107.941018768543), .Dim = c(6L, 6L), .Dimnames = list(
  c("D1_2", "A_2", "B_2", "D1_1", "A_1", "B_1"), c("D1_2",
    "A_2", "B_2", "D1_1", "A_1", "B_1"))), names = c("par",
"value", "counts", "convergence", "message", "hessian"), class = c("list", "Tagloss"))
par(mar=c(4, 4, 1, 1))
plot(o, t=1:3000, model="2", scale=1000, ylim=c(0, 3),
  col="red")
plot(o, t=1500:3000, model="1", scale=1000,
  add=TRUE)
legend("topright", legend=c("2 -> 1", "1 -> 0"), col=c("red", "black"), lty=1)

plot(o, t=1:300, model="2", scale=1000, ylim=c(0, 3),
  col="red", hessian=o$hessian)
plot(o, t=1:300, model="1", scale=1000,
  add=TRUE, hessian=o$hessian)
legend("topright", legend=c("2 -> 1", "1 -> 0"), col=c("red", "black"), lty=1)

##### Example with delta

data_f_21 <- Tagloss_format(outLR, model="21")
# Without the N20 the computing is much faster
data_f_21_fast <- subset(data_f_21, subset=(is.na(data_f_21$N20)))
par <- structure(c(45.8764973711504, 5.22489974562498, 8.07602162728874,
-0.865444694177429), names = c("D1_2", "A_2", "B_2", "delta")

```

```

))
pfixed <- c(D2D1_2=10000, D3D2_2=10000, C_2=0)
o <- Tagloss_fit(data=data_f_21_fast, fitted.parameters=par, fixed.parameters=pfixed)
# Here it is the result of the previous function
o <- structure(list(par = structure(c(45.9035484983855, 5.22576211343279,
8.07585745169786, -0.865706100004634), names = c("D1_2", "A_2",
"B_2", "delta")), value = 5913.716964613, counts = structure(c(91L,
NA), names = c("function", "gradient")), convergence = 0L, message = NULL,
hessian = structure(c(0.0644593001197791, -2.88983483187621,
-1.49161280660337, -0.0875163550517755, -2.88983483187621,
221.02317802819, 45.3729608125286, 3.73816044429987, -1.49161280660337,
45.3729608125286, 440.129730122862, 30.4781699469459, -0.0875163550517755,
3.73816044429987, 30.4781699469459, 9.47964940678503), .Dim = c(4L,
4L), .Dimnames = list(c("D1_2", "A_2", "B_2", "delta"), c("D1_2",
"A_2", "B_2", "delta")))), names = c("par", "value", "counts",
"convergence", "message", "hessian"), class = c("list", "Tagloss"))
par(mar=c(4, 4, 1, 1))
plot(o, t=1:3000, model="2", scale=1000, ylim=c(0, 3),
col="red")
plot(o, t=1:3000, model="1", scale=1000, col="blue",
add=TRUE, hessian=o$hessian)
legend("topright", legend=c("2 -> 1", "1 -> 0"), col=c("red", "black"), lty=1)

##### Example with model_after
data_f_LR <- Tagloss_format(outLR, model="LR")
par <- structure(c(72.0399239978454, 58.1034231071992, 645.068735669251,
5.10791337470247, 3538.47220045768, 7.83358940767931),
names = c("D1_L2", "D2D1_L2", "D3D2_L2", "A_L2", "B_L2", "C_L2"))
pfixed <- NULL
# A progress bar can be shown when one core is used
system.time(
print(Tagloss_L(individuals=data_f_LR, par=par, days.maximum=Tagloss_daymax(data_f_LR),
fixed.parameters=pfixed, mc.cores=1, model_after="pR2=pL2;pR1=pL2;pL1=pL2",
progressbar = TRUE))
)
# When parallel computing is done, no progress bar can be shown
system.time(
print(Tagloss_L(individuals=data_f_LR, par=par, days.maximum=Tagloss_daymax(data_f_LR),
fixed.parameters=pfixed, model_after="pR2=pL2;pR1=pL2;pL1=pL2"))
)
# The NLR_00 are very long to calculate
data_f_LR_fast <- subset(data_f_LR, subset=(is.na(data_f_LR$NLR_00)))
system.time(
print(Tagloss_L(individuals=data_f_LR_fast, par=par, days.maximum=Tagloss_daymax(data_f_LR_fast),
fixed.parameters=pfixed, model_after="pR2=pL2;pR1=pL2;pL1=pL2"))
)
o <- Tagloss_fit(data=data_f_LR_fast,
fitted.parameters=par, fixed.parameters=pfixed,
model_after="pR2=pL2;pR1=pL2;pL1=pL2")

par(mar=c(4, 4, 1, 1))
plot(o, t=1:3000, model="2", scale=1000, ylim=c(0, 3),
col="red")

```

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

Tagloss_LengthObs	<i>Return a list with the number of days for different kinds of individuals are seen.</i>
-------------------	---

Description

Usefull to summarize data

Usage

```
Tagloss_LengthObs(data, progressbar = TRUE)
```

Arguments

data	Set of indivuals
progressbar	Is shown a progressbar?

Details

Tagloss_LengthObs returns a list with the number of days for different kinds of individuals are seen.

Value

Return a list with the number of days for different kinds of individuals are seen.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Model of Tag-loss: [Tagloss_L\(\)](#), [Tagloss_cumul\(\)](#), [Tagloss_daymax\(\)](#), [Tagloss_fit\(\)](#), [Tagloss_format\(\)](#), [Tagloss_mcmc\(\)](#), [Tagloss_mcmc_p\(\)](#), [Tagloss_model\(\)](#), [Tagloss_simulate\(\)](#), [logLik.Tagloss\(\)](#), [o_4p_p1p2](#), [plot.Tagloss\(\)](#), [plot.TaglossData\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
data_f_21 <- Tagloss_format(outLR, model="21")
Tagloss_LengthObs(data_f_21)

## End(Not run)
```

Tagloss_mcmc

Bayesian model of tag loss using a CMR database.

Description

This function fits a model of tag loss using a CMR database using Bayesian mcmc. The parameters must be stored in a data.frame with named rows for each parameter with the following columns:

- Density. The density function name, example dnorm, dlnorm, dunif
- Prior1. The first parameter to send to the Density function
- Prior2. The second parameter to send to the Density function
- SDProp. The standard error from new proposition value of this parameter
- Min. The minimum value for this parameter
- Max. The maximum value for this parameter
- Init. The initial value for this parameter

Usage

```
Tagloss_mcmc(
  data = stop("A database formatted using Tagloss_format() must be used"),
  parameters = stop("Priors must be supplied"),
  fixed.parameters = NULL,
  model_before = NULL,
  model_after = NULL,
  mc.cores = detectCores(all.tests = FALSE, logical = TRUE),
  groups = detectCores(all.tests = FALSE, logical = TRUE),
  n.iter = 10000,
  n.chains = 1,
  n.adapt = 100,
  thin = 30,
  trace = FALSE,
  traceML = FALSE,
  adaptive = FALSE,
  adaptive.lag = 500,
  adaptive.fun = function(x) {
    ifelse(x > 0.234, 1.3, 0.7)
  },
  intermediate = NULL,
  filename = "intermediate.Rdata",
  previous = NULL
)
```

Arguments

<code>data</code>	An object formatted using <code>Tagloss_format</code>
<code>parameters</code>	A data.frame with priors; see description and examples
<code>fixed.parameters</code>	Set of fixed parameters
<code>model_before</code>	Transformation of parameters before to use <code>Tagloss_model()</code>
<code>model_after</code>	Transformation of parameters after to use <code>Tagloss_model()</code>
<code>mc.cores</code>	Number of cores to use for parallel computing
<code>groups</code>	Number of groups for parallel computing
<code>n.iter</code>	Number of iterations for each chain
<code>n.chains</code>	Number of chains
<code>n.adapt</code>	Number of iteration to stabilize likelihood
<code>thin</code>	Interval for thinning Markov chain
<code>trace</code>	Or FALSE or period to show progress
<code>traceML</code>	TRUE or FALSE to show ML
<code>adaptive</code>	Should an adaptive process for SDProp be used
<code>adaptive.lag</code>	Lag to analyze the SDProp value in an adaptive context
<code>adaptive.fun</code>	Function used to change the SDProp
<code>intermediate</code>	Or NULL of period to save intermediate result
<code>filename</code>	Name of file in which intermediate results are saved
<code>previous</code>	The content of the file in which intermediate results are saved

Details

`Tagloss_mcmc` Bayesian model of tag loss using a CMR database.

Value

Return a list object with the Bayesian model describing tag loss.

Author(s)

Marc Girondot

See Also

Other Model of Tag-loss: `Tagloss_L()`, `Tagloss_LengthObs()`, `Tagloss_cumul()`, `Tagloss_daymax()`, `Tagloss_fit()`, `Tagloss_format()`, `Tagloss_mcmc_p()`, `Tagloss_model()`, `Tagloss_simulate()`, `logLik.Tagloss()`, `o_4p_p1p2`, `plot.Tagloss()`, `plot.TaglossData()`

Examples

```
## Not run:
library(phenology)
# Example
data_f_21 <- Tagloss_format(outLR, model="21")

# model fitted by Rivalan et al. 2005
par <- c(a0_2=-5.43E-2, a1_2=-103.52, a4_2=5.62E-4,
        delta_1=3.2E-4)
pfixed <- c(a2_2=0, a3_2=0, a2_1=0, a3_1=0)
model_before <- "par['a0_1']=par['a0_2'];par['a1_1']=par['a1_2'];par['a4_1']=par['a4_2']"
o <- Tagloss_fit(data=data_f_21, fitted.parameters=par, fixed.parameters=pfixed,
                model_before=model_before)
pMCMC <- Tagloss_mcmc_p(o, accept=TRUE)
o_MCMC <- Tagloss_mcmc(data=data_f_21, parameters=pMCMC, fixed.parameters=pfixed,
                      model_before=model_before,
                      n.iter=10000, n.chains = 1, n.adapt = 100, thin=30)

## End(Not run)
```

Tagloss_mcmc_p

*Generates set of parameters to be used with Tagloss_mcmc()***Description**

Interactive (or not!) script used to generate set of parameters to be used with Tagloss_mcmc().

Usage

```
Tagloss_mcmc_p(
  result = stop("An output from Tagloss_fit() must be provided"),
  default.density = "dunif",
  accept = FALSE
)
```

Arguments

result	An object obtained after a Tagloss_fit() fit
default.density	The default density, "dnorm" or "dunif"
accept	If TRUE, does not wait for user interaction

Details

Tagloss_mcmc_p generates set of parameters to be used with Tagloss_mcmc()

Value

A matrix with the parameters

Author(s)

Marc Girondot

See Also

Other Model of Tag-loss: [Tagloss_L\(\)](#), [Tagloss_LengthObs\(\)](#), [Tagloss_cumul\(\)](#), [Tagloss_daymax\(\)](#), [Tagloss_fit\(\)](#), [Tagloss_format\(\)](#), [Tagloss_mcmc\(\)](#), [Tagloss_model\(\)](#), [Tagloss_simulate\(\)](#), [logLik.Tagloss\(\)](#), [o_4p_p1p2](#), [plot.Tagloss\(\)](#), [plot.TaglossData\(\)](#)

Examples

```
## Not run:
library(phenology)
# Example
data_f_21 <- Tagloss_format(outLR, model="21")

# model fitted by Rivalan et al. 2005
par <- c(a0_2=-5.43E-2, a1_2=-103.52, a4_2=5.62E-4,
        delta_1=3.2E-4)
pfixed <- c(a2_2=0, a3_2=0, a2_1=0, a3_1=0)
model_before <- "par['a0_1']=par['a0_2'];par['a1_1']=par['a1_2'];par['a4_1']=par['a4_2']"
o <- Tagloss_fit(data=data_f_21, fitted.parameters=par, fixed.parameters=pfixed,
                model_before=model_before)
pMCMC <- Tagloss_mcmc_p(o, accept=TRUE)
o_MCMC <- Tagloss_mcmc(data=data_f_21, parameters=pMCMC, fixed.parameters=pfixed,
                    model_before=model_before,
                    n.iter=10000, n.chains = 1, n.adapt = 100, thin=30)

## End(Not run)
```

Tagloss_model

Return the daily rate of tag loss.

Description

This function compute a model of daily tag loss rate for days t based on a set of parameters, `par` or a fitted tag loss model in `x`.

Parameters are described in [Tagloss_fit](#).

Usage

```
Tagloss_model(
  t = NULL,
  par = NULL,
  Hessian = NULL,
  mcmc = NULL,
  model_before = NULL,
  model_after = NULL,
```

```

    model = stop("You must specify which tag loss rate you want."),
    method = NULL,
    replicates = NULL,
    x = NULL
  )

```

Arguments

t	Time for which values of model must be estimated
par	Parameters
Hessian	Hessian matrix of parameters
mcmc	A mcmc result
model_before	Function to be used before estimation of daily tagloss rate
model_after	Function to be used after estimation of daily tagloss rate
model	The model of parameter to be used, can be 1, 2, L1, L2, R1 or R2
method	Can be NULL, "delta", "SE", "Hessian", "MCMC", or "PseudoHessianFromMCMC"
replicates	Number of replicates to estimate se of output
x	A Tagloss fitted model

Details

Tagloss_model returns the daily rate of tag loss.

Value

Return the daily rate of tag loss if hessian is null or a data.frame with distribution of daily rate of tag loss if hessian is not null.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Model of Tag-loss: [Tagloss_L\(\)](#), [Tagloss_LengthObs\(\)](#), [Tagloss_cumul\(\)](#), [Tagloss_daymax\(\)](#), [Tagloss_fit\(\)](#), [Tagloss_format\(\)](#), [Tagloss_mcmc\(\)](#), [Tagloss_mcmc_p\(\)](#), [Tagloss_simulate\(\)](#), [logLik.Tagloss\(\)](#), [o_4p_p1p2](#), [plot.Tagloss\(\)](#), [plot.TaglossData\(\)](#)

Examples

```

## Not run:
library(phenology)

# Example
t <- 1:1000
par <- c(D1=200, D2D1=100, D3D2=200,
        A=-logit(0.02), B=-logit(0.05), C=-logit(0.07))

```

```

y <- Tagloss_model(t, par, model="1")
plot(x=t, y, type="l")
par <- c(D1_1=200, D2D1_1=100, D3D2_1=200,
        A_1=-logit(0.02), B_1=-logit(0.05), C_1=-logit(0.07))
y <- Tagloss_model(t, par, model="1")
phenology:::plot.Tagloss(x=list(), t=1:1000, fitted.parameters=par, model="1")

# Fig1A in Rivalan et al. 2005 (note an error for a0; a0 must be negative)
par <- c(a0=-1E5, a1=-2000, a2=0, a3=2*max(t), a4=0.1)
y <- Tagloss_model(t, par)
plot(x=t, y, type="l")

# Fig1B in Rivalan et al. 2005
par <- c(a0=-0.5, a1=-2000, a2=-0.001, a3=0, a4=0.1)
y <- Tagloss_model(t, par)
plot(x=t, y, type="l")

# Fig1C in Rivalan et al. 2005
par <- c(a0=-1, a1=-6, a2=0, a3=0, a4=0)
y <- Tagloss_model(t, par)
plot(x=t, y, type="l")

# Fig1D in Rivalan et al. 2005
par <- c(a0=-1, a1=-6, a2=0, a3=0, a4=0.1)
y <- Tagloss_model(t, par)
plot(x=t, y, type="l")

# Fig1E in Rivalan et al. 2005
par <- c(a0=-0.1, a1=-10, a2=-0.2, a3=60, a4=0.1)
y <- Tagloss_model(t, par)
plot(x=t, y, type="l")

# Fig1F in Rivalan et al. 2005
par <- c(a0=-0.1, a1=-10, a2=0.2, a3=60, a4=0.1)
y <- Tagloss_model(t, par)
plot(x=t, y, type="l")

# Example with fitted data
data_f_21 <- Tagloss_format(outLR, model="21")
# Without the N20 the computing is much faster
data_f_21_fast <- subset(data_f_21, subset=(is.na(data_f_21$N20)))
par <- c('D1_2' = 49.086835072129126,
        'D2D1_2' = 1065.0992647723231,
        'D3D2_2' = 6.15531475922079,
        'A_2' = 5.2179675647973758,
        'B_2' = 8.0045560376751386,
        'C_2' = 8.4082505219581876,
        'D1_1' = 177.23337287498103,
        'D2D1_1' = 615.42690323741033,
        'D3D2_1' = 2829.0806609455867,
        'A_1' = 28.500118091731551,
        'B_1' = 10.175426055942701,
        'C_1' = 6.9616630417169398)

```

```

o <- Tagloss_fit(data=data_f_21_fast, fitted.parameters=par)
t <- 1:10
y <- Tagloss_model(t, o$par, model="1")
y <- Tagloss_model(t, x=o, method=NULL, model="1")
y <- Tagloss_model(t, x=o, method="Hessian", model="1", replicates=1000)

## End(Not run)

```

Tagloss_simulate	<i>Return a list with the number of days different kinds of individuals are seen.</i>
------------------	---

Description

Generate data with known features.
model_before is applied to par parameter.
model_after is applied after par is separated in p1, p2, pL1, pL2, pR1 and pR2 parameters.
pobservation can be a vector of daily probabilities to be captured. The last value is repeated if necessary.
The maximum number of days of observation is exp(LengthObservation["max"]).
If model="12" then par must have _1 and _2 parameters.
if model="LR" then par must have _L2, _L1, _R2, R1 parameters.

Usage

```

Tagloss_simulate(
  n = 500,
  par,
  pobobservation = c(rep(0.05, 70), 0.01),
  LengthObservation = c(min = 0, max = 9),
  dailysurvival = 0.999,
  model = "12",
  model_before = NULL,
  model_after = NULL,
  progressbar = TRUE
)

```

Arguments

n	Number of individuals to simulate
par	Set of parameters
pobobservation	Probability of daily observation
LengthObservation	The log of number of days of observations is a random number between min and max
dailysurvival	Daily probability of survival

model	Must be "12" or "LR"
model_before	Transformation of parameters before to use Tagloss_model()
model_after	Transformation of parameters after to use Tagloss_model()
progressbar	Is a progressbar should be shown?

Details

Tagloss_simulate returns a list with the number of days different kinds of individuals are seen.

Value

Return a list with the number of days different kinds of individuals are seen.

Author(s)

Marc Girondot <marc.girondot@gmail.com>

See Also

Other Model of Tag-loss: [Tagloss_L\(\)](#), [Tagloss_LengthObs\(\)](#), [Tagloss_cumul\(\)](#), [Tagloss_daymax\(\)](#), [Tagloss_fit\(\)](#), [Tagloss_format\(\)](#), [Tagloss_mcmc\(\)](#), [Tagloss_mcmc_p\(\)](#), [Tagloss_model\(\)](#), [logLik.Tagloss\(\)](#), [o_4p_p1p2](#), [plot.Tagloss\(\)](#), [plot.TaglossData\(\)](#)

Examples

```
library(phenology)
## Not run:
# Example
par <- structure(c(49.5658922243074, 808.136085362158, 106.283783786853,
5.22150592456511, 8.00608716525864, 8.32718202233396, 150.612916258503,
715.865805125223, 2242.06574225966, 119.212383120678, 10.1860735529433,
7.14231725937626), names = c("D1_2", "D2D1_2", "D3D2_2", "A_2",
"B_2", "C_2", "D1_1", "D2D1_1", "D3D2_1", "A_1", "B_1", "C_1"))
cmr <- Tagloss_simulate(n=500,
                        par=par, model="12")
cmr_f <- Tagloss_format(cmr, model="12")

## End(Not run)
```

toggle_Min_PMin	<i>Transform a set of parameters from Min, MinB or MinE to PMin, PminB or PminE, or reverse</i>
-----------------	---

Description

This function is used to transform a set of parameters that uses Min, MinB or MinE to a set of parameters that uses PMin, PminB or PminE, or reverse.

Usage

```
toggle_Min_PMin(parameters = stop("A set of parameters must be indicated"))
```

Arguments

parameters Set of current parameters

Details

toggle_Min_PMin transforms a set of parameters from Min, MinB or MinE to PMin, PminB or PminE, or reverse

Value

Return a set of modified parameters

Author(s)

Marc Girondot <marc.girondot@gmail.com>

Examples

```
# Read a file with data
data(Gratiot)
# Generate a formatted list named data_Gratiot
refdate <- as.Date("2001-01-01")
data_Gratiot<-add_phenology(Gratiot, name="Complete", reference=refdate, format="%d/%m/%Y")
# Generate initial points for the optimisation
parg<-par_init(data_Gratiot)
# Change the parameters to PMinB and PMinE
parg1<-toggle_Min_PMin(parameters=parg)
# And change back to MinB and MinE
parg2<-toggle_Min_PMin(parameters=parg1)
```

Index

- * **Clutch**
 - phenology-package, 4
- * **ECF**
 - phenology-package, 4
- * **Ecology**
 - phenology-package, 4
- * **Exponential regression**
 - ExponentialRegression, 25
- * **Fill gaps for RMU**
 - phenology2fitRMU, 92
- * **Fill gaps in RMU**
 - CI.RMU, 17
 - fitRMU, 39
 - fitRMU_MHmcmc, 44
 - fitRMU_MHmcmc_p, 46
 - logLik.fitRMU, 76
 - plot.fitRMU, 101
- * **Model of Clutch Frequency**
 - ECFOCF_f, 22
 - ECFOCF_full, 23
 - fitCF, 29
 - fitCF_MHmcmc, 35
 - fitCF_MHmcmc_p, 38
 - generateCF, 55
 - lnLCF, 72
 - logLik.ECFOCF, 75
 - plot.ECFOCF, 98
 - plot.TableECFOCF, 115
 - TableECFOCF, 160
- * **Model of Interesting Period**
 - IPFit, 58
 - IPModel, 65
 - IPPredict, 67
 - plot.IP, 104
 - summary.IP, 154
- * **Model of Remigration Interval**
 - Bayesian.remigration, 14
 - LnRI_norm, 73
 - plot.Remigration, 113
- RI, 134
- RI2BP, 136
- * **Model of Tag-loss**
 - logLik.Tagloss, 78
 - o_4p_p1p2, 87
 - plot.Tagloss, 117
 - plot.TaglossData, 121
 - Tagloss_cumul, 161
 - Tagloss_daymax, 165
 - Tagloss_fit, 166
 - Tagloss_format, 169
 - Tagloss_L, 171
 - Tagloss_LengthObs, 176
 - Tagloss_mcmc, 177
 - Tagloss_mcmc_p, 179
 - Tagloss_model, 180
 - Tagloss_simulate, 183
- * **OCF**
 - phenology-package, 4
- * **Phenology model**
 - adapt_parameters, 6
 - add_phenology, 7
 - add_SE, 12
 - AutoFitPhenology, 13
 - BE_to_LBLE, 16
 - extract_result, 28
 - fit_phenology, 47
 - Gratiot, 57
 - L_to_LBLE, 79
 - LBLE_to_BE, 68
 - LBLE_to_L, 69
 - likelihood_phenology, 70
 - logLik.phenology, 77
 - map_Gratiot, 80
 - map_phenology, 81
 - MarineTurtles_2002, 84
 - MinBMinE_to_Min, 85
 - par_init, 89
 - phenology, 91

- phenology2fitRMU, 92
- phenology_MHmcmc, 94
- phenology_MHmcmc_p, 97
- plot.phenology, 107
- plot.phenologydata, 110
- plot.phenologymap, 112
- plot_delta, 122
- plot_phi, 123
- print.phenology, 125
- print.phenologymap, 126
- print.phenologyout, 128
- remove_site, 129
- result_Gratiot, 130
- result_Gratiot1, 131
- result_Gratiot2, 132
- result_Gratiot_Flat, 133
- summary.phenology, 155
- summary.phenologymap, 157
- summary.phenologyout, 158
- * **Phenology**
 - phenology-package, 4
- * **Seasonality**
 - phenology-package, 4
- * **datasets**
 - Gratiot, 57
 - map_Gratiot, 80
 - MarineTurtles_2002, 84
 - o_4p_p1p2, 87
 - outLR, 86
 - result_Gratiot, 130
 - result_Gratiot1, 131
 - result_Gratiot2, 132
 - result_Gratiot_Flat, 133
- * **tagloss**
 - phenology-package, 4
- adapt_parameters, 6
- adapt_parameters(), 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159
- add_phenology, 7
- add_phenology(), 7, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159
- add_SE, 11
- add_SE(), 7, 10, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159
- AutoFitPhenology, 13
- AutoFitPhenology(), 7, 10, 12, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159
- Bayesian.remigration, 14
- Bayesian.remigration(), 74, 114, 135, 137
- BE_to_LBLE, 16
- BE_to_LBLE(), 7, 10, 12, 13, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159
- CI.RMU, 17
- CI.RMU(), 41, 45, 47, 76, 102
- ECFOCF_f, 22
- ECFOCF_f(), 25, 31, 37, 38, 56, 73, 75, 100, 116, 161
- ECFOCF_full, 23
- ECFOCF_full(), 23, 31, 37, 38, 56, 73, 75, 100, 116, 161
- ExponentialRegression, 25
- extract_result, 28
- extract_result(), 7, 10, 12, 13, 17, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159
- fit_phenology, 47
- fit_phenology(), 7, 10, 12, 13, 17, 28, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159
- fitCF, 29
- fitCF(), 23, 25, 37, 38, 56, 73, 75, 100, 116, 161
- fitCF_MHmcmc, 35
- fitCF_MHmcmc(), 23, 25, 31, 38, 56, 73, 75, 100, 116, 161
- fitCF_MHmcmc_p, 38

- fitCF_MHmcmc_p(), *23, 25, 31, 37, 56, 73, 75, 100, 116, 161*
 fitRMU, *39*
 fitRMU(), *19, 45, 47, 76, 102*
 fitRMU_MHmcmc, *44*
 fitRMU_MHmcmc(), *19, 41, 47, 76, 102*
 fitRMU_MHmcmc_p, *46*
 fitRMU_MHmcmc_p(), *19, 41, 45, 76, 102*
 fixed.parameters0, *54*

 generateCF, *55*
 generateCF(), *23, 25, 31, 37, 38, 73, 75, 100, 116, 161*
 Gratiot, *7, 10, 12, 13, 17, 28, 49, 57, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159*

 IPFit, *58*
 IPFit(), *66, 67, 105, 155*
 IPModel, *65*
 IPModel(), *60, 67, 105, 155*
 IPPredict, *67*
 IPPredict(), *60, 66, 105, 155*

 L_to_LBLE, *79*
 L_to_LBLE(), *7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159*
 LBLE_to_BE, *68*
 LBLE_to_BE(), *7, 10, 12, 13, 17, 28, 49, 58, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159*
 LBLE_to_L, *69*
 LBLE_to_L(), *7, 10, 12, 13, 17, 28, 49, 58, 68, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159*
 likelihood_phenology, *70*
 likelihood_phenology(), *7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159*
 lnLCF, *72*
 lnLCF(), *23, 25, 31, 37, 38, 56, 75, 100, 116, 161*

 LnRI_norm, *73*
 LnRI_norm(), *15, 114, 135, 137*
 logLik.ECFOCF, *75*
 logLik.ECFOCF(), *23, 25, 31, 37, 38, 56, 73, 100, 116, 161*
 logLik.fitRMU, *76*
 logLik.fitRMU(), *19, 41, 45, 47, 102*
 logLik.phenology, *77*
 logLik.phenology(), *7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159*
 logLik.Tagloss, *78*
 logLik.Tagloss(), *87, 119, 122, 162, 165, 168, 170, 172, 176, 178, 180, 181, 184*

 map_Gratiot, *7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159*
 map_phenology, *81*
 map_phenology(), *7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159*
 MarineTurtles_2002, *7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 84, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159*
 MinBMinE_to_Min, *85*
 MinBMinE_to_Min(), *7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159*

 o_4p_p1p2, *79, 87, 119, 122, 162, 165, 168, 170, 172, 176, 178, 180, 181, 184*
 outLR, *86*

 par_init, *89*
 par_init(), *7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159*

- Parameter_Global_Year, 88
- phenology, 91
- phenology(), 7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159
- phenology-package, 4
- phenology2fitRMU, 92
- phenology2fitRMU(), 7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159
- phenology_MHmcmc, 94
- phenology_MHmcmc(), 7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159
- phenology_MHmcmc_p, 97
- phenology_MHmcmc_p(), 7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159
- plot.ECFOCF, 98
- plot.ECFOCF(), 23, 25, 31, 37, 38, 56, 73, 75, 116, 161
- plot.fitRMU, 101
- plot.fitRMU(), 19, 41, 45, 47, 76
- plot.IP, 104
- plot.IP(), 60, 66, 67, 155
- plot.phenology, 107
- plot.phenology(), 7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 111, 112, 123, 124, 126–128, 130–134, 156, 157, 159
- plot.phenologydata, 110
- plot.phenologydata(), 7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 112, 123, 124, 126–128, 130–134, 156, 157, 159
- plot.phenologymap, 112
- plot.phenologymap(), 7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 123, 124, 126–128, 130–134, 156, 157, 159
- plot.Remigration, 113
- plot.Remigration(), 15, 74, 135, 137
- plot.TableECFOCF, 115
- plot.TableECFOCF(), 23, 25, 31, 37, 38, 56, 73, 75, 100, 161
- plot.Tagloss, 117
- plot.Tagloss(), 79, 87, 122, 162, 165, 168, 170, 172, 176, 178, 180, 181, 184
- plot.TaglossData, 121
- plot.TaglossData(), 79, 87, 119, 162, 165, 168, 170, 172, 176, 178, 180, 181, 184
- plot_delta, 122
- plot_delta(), 7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 124, 126–128, 130–134, 156, 157, 159
- plot_phi, 123
- plot_phi(), 7, 10, 12, 13, 17, 28, 49, 58, 68, 69, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 126–128, 130–134, 156, 157, 159
- print.phenology, 125
- print.phenology(), 7, 10, 12, 13, 17, 28, 49, 58, 68, 70, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 127, 128, 130–134, 156, 157, 159
- print.phenologymap, 126
- print.phenologymap(), 7, 10, 12, 13, 17, 28, 49, 58, 68, 70, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126, 128, 130–134, 156, 157, 159
- print.phenologyout, 128
- print.phenologyout(), 7, 10, 12, 13, 17, 28, 49, 58, 68, 70, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126, 127, 130–134, 156, 157, 159
- remove_site, 129
- remove_site(), 7, 10, 12, 13, 17, 28, 49, 58, 68, 70, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 131–134, 156, 157, 159

- result_Gratiot, [7, 10, 12, 13, 17, 28, 49, 58, 68, 70, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130, 130, 132–134, 156, 157, 159](#)
- result_Gratiot1, [7, 10, 12, 13, 17, 28, 49, 58, 68, 70, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130, 131, 131, 133, 134, 156, 157, 159](#)
- result_Gratiot2, [7, 10, 12, 13, 17, 28, 49, 58, 68, 70, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–132, 132, 134, 156, 157, 159](#)
- result_Gratiot_Flat, [7, 10, 12, 13, 17, 28, 49, 58, 68, 70, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–133, 133, 156, 157, 159](#)
- RI, [134](#)
- RI(), [15, 74, 114, 137](#)
- RI2BP, [136](#)
- RI2BP(), [15, 74, 114, 135](#)
- Sea_Turtles_Bayesian, [138](#)
- shift_sinusoid, [152](#)
- summary.IP, [154](#)
- summary.IP(), [60, 66, 67, 105](#)
- summary.phenology, [155](#)
- summary.phenology(), [7, 10, 12, 13, 17, 28, 49, 58, 68, 70, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 157, 159](#)
- summary.phenologymap, [157](#)
- summary.phenologymap(), [7, 10, 12, 13, 17, 28, 49, 58, 68, 70, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 159](#)
- summary.phenologyout, [158](#)
- summary.phenologyout(), [7, 10, 12, 13, 17, 28, 49, 58, 68, 70, 71, 78, 80, 81, 83, 85, 86, 90, 92, 93, 95, 97, 109, 111, 112, 123, 124, 126–128, 130–134, 156, 157](#)
- TableECFOCF, [160](#)
- TableECFOCF(), [23, 25, 31, 37, 38, 56, 73, 75, 100, 116](#)
- Tagloss_cumul, [161](#)
- Tagloss_cumul(), [79, 87, 119, 122, 165, 168, 170, 172, 176, 178, 180, 181, 184](#)
- Tagloss_daymax, [165](#)
- Tagloss_daymax(), [79, 87, 119, 122, 162, 168, 170, 172, 176, 178, 180, 181, 184](#)
- Tagloss_fit, [161, 166, 180](#)
- Tagloss_fit(), [79, 87, 119, 122, 162, 165, 170, 172, 176, 178, 180, 181, 184](#)
- Tagloss_format, [169](#)
- Tagloss_format(), [79, 87, 119, 122, 162, 165, 168, 172, 176, 178, 180, 181, 184](#)
- Tagloss_L, [171](#)
- Tagloss_L(), [79, 87, 119, 122, 162, 165, 168, 170, 176, 178, 180, 181, 184](#)
- Tagloss_LengthObs, [176](#)
- Tagloss_LengthObs(), [79, 87, 119, 122, 162, 165, 168, 170, 172, 178, 180, 181, 184](#)
- Tagloss_mcmc, [177](#)
- Tagloss_mcmc(), [79, 87, 119, 122, 162, 165, 168, 170, 172, 176, 180, 181, 184](#)
- Tagloss_mcmc_p, [179](#)
- Tagloss_mcmc_p(), [79, 87, 119, 122, 162, 165, 168, 170, 172, 176, 178, 181, 184](#)
- Tagloss_model, [180](#)
- Tagloss_model(), [79, 87, 119, 122, 162, 165, 168, 170, 172, 176, 178, 180, 184](#)
- Tagloss_simulate, [183](#)
- Tagloss_simulate(), [79, 87, 119, 122, 162, 165, 168, 170, 172, 176, 178, 180, 181](#)
- toggle_Min_PMin, [184](#)