Package: phenomix (via r-universe)

November 26, 2024

Type Package

Title Fit Density Curves to Peak Timing Data that Varies over Time

Version 1.0.4

Description The 'phenomix' package fits time-varying density curves to run timing type data commonly encountered in fisheries and ecology. Example applications include to peak run timing curves collected for juvenile or adult Pacific salmon, though could also be applied to other kinds of data such as hydrographs, plant phenology (flowering, leaf out).

License GPL (>=3)

URL `https://nwfsc-cb.github.io/phenomix`,
`https://github.com/nwfsc-cb/phenomix`

Depends R (>= 4.0.0)

Imports dplyr, ggplot2, gnorm, methods, stats, TMB (>= 1.7.20), nlme

Suggests testthat, knitr, rmarkdown

LinkingTo RcppEigen, TMB

VignetteBuilder knitr

ByteCompile true

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

SystemRequirements GNU make

Config/pak/sysreqs make

Repository https://nmfs-opensci.r-universe.dev

RemoteUrl https://github.com/nwfsc-cb/phenomix

RemoteRef HEAD

RemoteSha cda03751c76e124182eda2c0867026452984bdf5

Contents

`create_data` *Create data file for fitting time varying run timing distributions with TMB*

Description

Does minimal processing of data to use as argument to fitting function

Usage

```
create_data(
  data,
  min_number = 0,
  variable = "number",
  time = "year",
  date = "doy",
  asymmetric_model = TRUE,
  mu = ~1,
  sigma = ~1,
  covar_data = NULL,
  est_sigma_re = TRUE,
  est_mu_re = TRUE,
  tail_model = "student_t",
  family = "lognormal",
  max_theta = 10,
  share_shape = TRUE,
  nu_prior = c(2, 10),
  beta_prior = c(2, 1)
)
```

Arguments

`data` A data frame

`min_number` A minimum threshold to use, defaults to 0

`variable` A character string of the name of the variable in 'data' that contains the response (e.g. counts)

`time` A character string of the name of the variable in 'data' that contains the time variable (e.g. year)

`date` A character string of the name of the variable in 'data' that contains the response (e.g. day of year). The actual #' column should contain a numeric response – for example, the result from using lubridate::yday(x)

`asymmetric_model`
Boolean, whether or not to let model be asymmetric (e.g. run timing before peak has a different shape than run timing after peak)

`mu` An optional formula allowing the mean to be a function of covariates. Random effects are not included in the formula but specified with the `est_mu_re` argument

`sigma` An optional formula allowing the standard deviation to be a function of covariates. For asymmetric models, each side of the distribution is allowed a different set of covariates. Random effects are not included in the formula but specified with the `est_sigma_re` argument

`covar_data` a data frame containing covariates specific to each time step. These are used in the formulas `mu` and `sigma`

`est_sigma_re` Whether to estimate random effects by year in sigma parameter controlling tail of distribution. Defaults to TRUE

`est_mu_re` Whether to estimate random effects by year in mu parameter controlling location of distribution. Defaults to TRUE

`tail_model` Whether to fit Gaussian ("gaussian"), Student-t ("student_t") or generalized normal ("gnorm"). Defaults to Student-t

`family` Response for observation model, options are "gaussian", "poisson", "negbin", "binomial", "lognormal". The default ("lognormal") is not a true lognormal distribution, but a normal-log in that it assumes log(y) ~ Normal()

`max_theta` Maximum value of log(pred) when `limits=TRUE`. Defaults to 10

`share_shape` Boolean argument for whether asymmetric student-t and generalized normal distributions should share the shape parameter (nu for the student-t; beta for the generalized normal). Defaults to TRUE

`nu_prior` Two element vector (optional) for penalized prior on student t df, defaults to a Gamma(shape=2, scale=10) distribution

`beta_prior` Two element vector (optional) for penalized prior on generalized normal beta, defaults to a Normal(2, 1) distribution

Examples

```
data(fishdist)
datalist <- create_data(fishdist,
  min_number = 0, variable = "number", time = "year",
  date = "doy", asymmetric_model = TRUE, family = "gaussian"
)
```

---

extract_all *Output processing function to be called by user*

Description

This function extracts the means, sigmas, thetas, lower (25%) and upper (75%) quartiles, and respective sds

Usage

```
extract_all(fit)
```

Arguments

fit	A fitted object returned from fit()

---

extract_annual *Output processing function to be called by user*

Description

This function extracts the annual totals

Usage

```
extract_annual(fit, log = TRUE)
```

Arguments

fit	A fitted object returned from fit()
log	Whether to return estimates in log space, defaults to TRUE

`extract_lower` *Output processing function to be called by user*

Description

This function extracts the lower quartiles (25%) and respective sds

Usage

```
extract_lower(fit)
```

Arguments

`fit`	A fitted object returned from fit()

`extract_means` *Output processing function to be called by user*

Description

This function extracts the parameter means and respective sds

Usage

```
extract_means(fit)
```

Arguments

`fit`	A fitted object returned from fit()

`extract_sigma` *Output processing function to be called by user*

Description

This function extracts the parameter sigma and respective sds

Usage

```
extract_sigma(fit)
```

Arguments

`fit`	A fitted object returned from fit()

`extract_theta` *Output processing function to be called by user*

Description

This function extracts the parameter theta and respective sds

Usage

```
extract_theta(fit)
```

Arguments

`fit`	A fitted object returned from fit()

`extract_upper` *Output processing function to be called by user*

Description

This function extracts the upper quartiles (25%) and respective sds

Usage

```
extract_upper(fit)
```

Arguments

`fit`	A fitted object returned from fit()

`fit` *Fitting function to be called by user*

Description

This function creates a list of parameters, sets up TMB object and attempts to do fitting / estimation

Usage

```
fit(
  data_list,
  silent = FALSE,
  inits = NULL,
  control = list(eval.max = 2000, iter.max = 1000, rel.tol = 1e-10),
  limits = NULL,
  fit_model = TRUE
)
```

Arguments

`data_list` A list of data, as output from create_data

`silent` Boolean passed to TMB::MakeADFun, whether to be verbose or not (defaults to FALSE)

`inits` Optional named list of parameters for starting values, defaults to NULL

`control` Optional control list for stats::nlminb. For arguments see ?nlminb. Defaults to eval.max=2000, iter.max=1000, rel.tol=1e-10. For final model runs, the rel.tol should be even smaller

`limits` Whether to include limits for stats::nlminb. Can be a list of (lower, upper), or TRUE to use suggested hardcoded limits. Defaults to NULL, where no limits used.

`fit_model` Whether to fit the model. If not, returns a list including the data, parameters, and initial values. Defaults to TRUE

Examples

```
data(fishdist)

# example of fitting fixed effects, no trends, no random effects
set.seed(1)
datalist <- create_data(fishdist[which(fishdist$year > 1970), ],
  asymmetric_model = FALSE,
  est_mu_re = FALSE, est_sigma_re = FALSE
)
fit <- fit(datalist)
#
# # example of model with random effects in means only, and symmetric distribution
# set.seed(1)
# datalist <- create_data(fishdist[which(fishdist$year > 1970),], asymmetric_model = FALSE,
#                       est_sigma_re = FALSE)
# fit <- fit(datalist)
# # example of model with random effects in variances
# set.seed(1)
# datalist <- create_data(fishdist[which(fishdist$year > 1970),], asymmetric_model = TRUE,
#                           est_mu_re = TRUE)
# fit <- fit(datalist)
#
# # example of model with poisson response
# set.seed(1)
# datalist <- create_data(fishdist[which(fishdist$year > 1970),], asymmetric_model = FALSE,
#                          est_sigma_trend=FALSE, est_mu_trend=FALSE, est_mu_re = TRUE,
#                          family="poisson")
# fit <- fit(datalist)
```

fixef.phenomix *Get fixed effects parameters from model object, copying glmmTMB 'fast' implementation*

Description

Get fixed effects parameters from model object, copying glmmTMB 'fast' implementation

Usage

```
## S3 method for class 'phenomix'
fixef(object, ...)
```

Arguments

`object`	The fitted phenomix model
`...`	Additional arguments

limits *Internal function to assign upper and lower bounds to parameters, based on their names*

Description

Arguments can be adjusted as needed

Usage

```
limits(parnames, max_theta)
```

Arguments

`parnames`	A vector of character strings or names used for estimation
`max_theta`	A scalar (optional) giving the maximum value of theta, log(pred)

plot_diagnostics *Plotting function to be called by user*

Description

These functions make some basic plots for the user

Usage

```
plot_diagnostics(fitted, type = "timing", logspace = TRUE)
```

Arguments

`fitted`	A fitted model object
`type`	A plot type for ggplot, either "timing" or "scatter"
`logspace`	whether to plot the space in log space, defaults to TRUE

ranef.phenomix *Get random effects parameters from model object, copying glmmTMB ’fast’ implementation*

Description

Get random effects parameters from model object, copying glmmTMB ’fast’ implementation

Usage

```
## S3 method for class 'phenomix'
ranef(object, ...)
```

Arguments

`object`	The fitted phenomix model
`...`	Additional arguments

Index