

Package: ef2 (via r-universe)

October 28, 2026

Title Modelling Framework for the Estimation of Salmonid Abundance

Version 2.0.0

Date 2026-10-09

Description Estimates capture probabilities and densities from multipass removal (electrofishing) data, following Millar et al. (2016) <[doi:10.1016/j.fishres.2016.01.001](https://doi.org/10.1016/j.fishres.2016.01.001)>. Capture probability is modelled on the logit scale using fixed effects, penalised smooth terms and random effects, for example for between visit and between sample variation. The likelihood is computed using 'RTMB', with random effects integrated out using the Laplace approximation.

URL <https://faskally.github.io/ef2/>, <https://github.com/Faskally/ef2>

BugReports <https://github.com/Faskally/ef2/issues>

Depends R (>= 4.1.0)

Imports Matrix, RTMB, methods, mgcv

Suggests testthat (>= 3.0.0)

License MIT + file LICENSE

LazyData TRUE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Repository <https://faskally.r-universe.dev>

Date/Publication 2026-10-28 20:35:42 UTC

RemoteUrl <https://github.com/faskally/ef2>

RemoteRef HEAD

RemoteSha 4c92a7bd619253f1f05071d4e23c99c6d740ff14

Contents

ef2-package	2
abundance	3
as.gam2	5
capture_prob	6
ef2_data	7
efp2	8
fitted.efp2	11
predict.efp2	13
rw_penalty	14
Index	16

ef2-package	<i>Modelling Framework for the Estimation of Salmonid Abundance</i>
-------------	---

Description

A set of functions to estimate capture probabilities and densities from multipass removal (electrofishing) data. The main function is [efp2](#), which fits a model for capture probability, and predictions of capture probability are available from [predict.efp2](#).

Author(s)

Maintainer: Colin Millar <colin.millar@ices.dk>

Authors:

- Rob Fryer
- Iain Malcolm
- Ross Glover

Other contributors:

- Marine Scotland Science [copyright holder]

References

Millar, C. P., Fryer, R. J., Millidine, K. J. and Malcolm, I. A. (2016) Modelling capture probability of Atlantic salmon (*Salmo salar*) from a diverse national electrofishing dataset: Implications for the estimation of abundance. *Fisheries Research*, 177, 1-12. [doi:10.1016/j.fishres.2016.01.001](#)

Glover, R. S., Fryer, R. J., Soulsby, C., Bacon, P. J. and Malcolm, I. A. (2018) Incorporating estimates of capture probability and river network covariance in novel habitat - abundance models: Assessing the effects of conservation stocking on catchment-scale production of juvenile Atlantic salmon (*Salmo salar*) from a long-term electrofishing dataset. *Ecological Indicators*, 93, 302-315. [doi:10.1016/j.ecolind.2018.05.013](#)

See Also

Useful links:

- <https://faskally.github.io/ef2/>
- <https://github.com/Faskally/ef2>
- Report bugs at <https://github.com/Faskally/ef2/issues>

abundance

Estimate abundance and density from removal samples

Description

Abundance is estimated as $\hat{N} = T/P$, where T is the total catch over the passes of a sample and P is the cumulative capture probability, see [capture_prob](#).

Usage

```
abundance(
  object,
  newdata = NULL,
  area = NULL,
  by = NULL,
  target = c("realised", "expected"),
  level = 0.95
)
```

Arguments

object	an efp2 fitted model
newdata	an optional data.frame as for capture_prob , which must also contain the counts used as the response in the model
area	an optional name of a column in the data giving the area fished, or a vector of areas with one value per row of the data. If given, density (abundance per unit area) is also returned. With by, the area must be the same for all samples in a group.
by	an optional character vector of the names of variables in the data, which must be the same for all passes of a sample. Abundance is summed over the samples with each combination of their values.
target	the abundance to estimate, "realised" (the default) or "expected", see Details
level	the confidence level

Details

Two targets are available, which have the same estimate but different standard errors:

"realised" the number of fish N present when the sample was taken. Given N , the catch is binomial, $T \sim \text{Bin}(N, P)$, and (Huggins, 1989)

$$\text{var}(\hat{N}) \approx \hat{N}(1 - P)/P + \hat{N}^2(1 - P)^2\text{var}(\text{logit}(P)).$$

The lower confidence limit is not less than the catch.

"expected" the expected number of fish λ , for example density times area, when the number present is $N \sim \text{Poisson}(\lambda)$. The catch is then $T \sim \text{Poisson}(\lambda P)$, and the first term of the variance is $\hat{\lambda}/P$, which also includes the natural variation of N about λ . The standard error is larger by a factor of about $1/\sqrt{1 - P}$ when P is well estimated.

Use "realised" to say how many fish were present, and "expected" to describe the underlying abundance, for example when comparing sites or years. A Poisson model for the counts with offset $\log(AP)$, where A is the area, is a model for the expected abundance.

Confidence intervals are computed on the log scale. When the total catch is zero the estimated abundance is zero and the confidence limits are NA.

With by, abundance is summed over the samples in each group, for example over the lifestages caught at a site in a year. The variation in the catches is independent between samples, but the estimates of P are correlated, as they depend on the same model coefficients, and this is allowed for:

$$\text{var}\left(\sum_s \hat{N}_s\right) \approx \sum_s v_s + w^\top \Sigma w,$$

where v_s is $\hat{N}_s(1 - P_s)/P_s$ for the realised abundance and $\hat{N}_s/P_s$ for the expected abundance, $w_s = \hat{N}_s(1 - P_s)$ and Σ is the covariance matrix of $\text{logit}(P_s)$.

Value

a data.frame with one row per sample, in the order the samples first appear in the data, with columns id, catch the total catch, P, N the estimated abundance, its standard error se, and confidence limits lower and upper. If area is given there are also columns area, density, density_lower and density_upper. With by there is one row per group, sorted by the by variables, with the by variables and nsample, the number of samples, in place of id and P.

References

Huggins, R. M. (1989) On the statistical analysis of capture experiments. *Biometrika*, 76, 133-140.
doi:10.1093/biomet/76.1.133

See Also

[capture_prob](#)

Examples

```
ef2_data$pass12 <- factor(pmin(ef2_data$pass, 2))
ef2_data$sampleID <- as.numeric(factor(paste(ef2_data$date,
  ef2_data$siteID, ef2_data$lifestage)))
ef2_data <- ef2_data[order(ef2_data$sampleID, ef2_data$pass), ]
m1 <- efp2(count ~ lifestage + pass12, data = ef2_data,
  pass = pass, id = sampleID)
head(abundance(m1, area = "area"))

# summed over lifestages for each site and year
head(abundance(m1, area = "area", by = c("siteID", "year")))

# expected abundance, for comparing years
abundance(m1, by = "year", target = "expected")
```

as.gam2

*Conversion of an efp2 fit to a gam fit for plotting***Description**

Converts an efp2 fit to a gam object, so that the plotting and prediction tools in the mgcv package can be used. The coefficients and their variances are those estimated by efp2, but the smoothing parameters and effective degrees of freedom in the gam object are not.

Usage

```
as.gam2(object)
```

Arguments

object an efp2 fitted model

Details

The conversion fits a model using mgcv, which can take a very long time if the model has many random effects. Predictions do not need the conversion, see [predict.efp2](#).

Value

an object of class gam, see [gamObject](#)

Examples

```
data(ef2_data)
ef2_data$sampleID <- factor(paste(ef2_data$date, ef2_data$siteID,
  ef2_data$lifestage, ef2_data$species))
m1 <- efp2(count ~ lifestage + s(doy, k = 3),
  data = ef2_data, pass = pass, id = sampleID)
```

```
m1_gam <- as.gam2(m1)
plot(m1_gam)
```

capture_prob	<i>Cumulative capture probability of each sample</i>
--------------	--

Description

The probability that a fish is caught on one of the passes of a sample, $P = 1 - \prod_i (1 - p_i)$, where p_i is the capture probability on pass i .

Usage

```
capture_prob(object, newdata = NULL, se.fit = FALSE, level = 0.95)
```

Arguments

object	an efp2 fitted model
newdata	an optional data.frame with one row per pass, containing the covariates in the model and the variables used for pass and id in the call to efp2 . If omitted the data used to fit the model are used. An offset is only included for the data used to fit the model.
se.fit	if TRUE standard errors and confidence intervals are returned
level	the confidence level

Details

Standard errors are computed on the logit scale by the delta method, using the covariance of the fixed and random effects. The derivative of $\text{logit}(P)$ with respect to the linear predictor of pass i is p_i/P . Confidence intervals are computed on the logit scale and back transformed.

Value

a data.frame with one row per sample, in the order the samples first appear in the data, with columns id, npass the number of passes and P. If se.fit = TRUE there are also columns se, the standard error of $\text{logit}(P)$, and lower and upper, the confidence limits for P.

References

Glover, R. S., Fryer, R. J., Soulsby, C., Bacon, P. J. and Malcolm, I. A. (2018) Incorporating estimates of capture probability and river network covariance in novel habitat - abundance models. Ecological Indicators, 93, 302-315. [doi:10.1016/j.ecolind.2018.05.013](https://doi.org/10.1016/j.ecolind.2018.05.013)

See Also

[abundance](#)

Examples

```
ef2_data$pass12 <- factor(pmin(ef2_data$pass, 2))
ef2_data$sampleID <- as.numeric(factor(paste(ef2_data$date,
  ef2_data$siteID, ef2_data$lifestage)))
ef2_data <- ef2_data[order(ef2_data$sampleID, ef2_data$pass), ]
m1 <- efp2(count ~ lifestage + pass12, data = ef2_data,
  pass = pass, id = sampleID)
head(capture_prob(m1, se.fit = TRUE))
```

ef2_data

Counts of salmon fry and parr from electrofishing.

Description

A dataset containing counts from multipass electrofishing samples over three sites and 5 years.

Usage

```
ef2_data
```

Format

A data frame with 90 rows and 10 variables:

siteID Unique identifier for each electrofishing site

year year of data collection

date date of data collections

doy day of the year of data collection, a continuous covariate which can be used to fit smooth terms

EFpasscount the total number of electrofishing passes

pass the electrofishing pass on which the fish were caught

species fish species, Salmon

lifestage fish lifestage, Fry or Parr

count the number of fish caught per pass for each site visit and species, etc.

area the area of river fished

Source

<https://www.gov.scot/publications/scotland-river-temperature-monitoring-network-srtmn/>

efp2

*Estimate capture probabilities from electrofishing data***Description**

This function uses the marginal likelihood of capture probabilities to estimate model parameters

Usage

```
efp2(
  formula,
  data = NULL,
  pass = pass,
  id = id,
  offset = NULL,
  verbose = FALSE,
  hessian = TRUE,
  fit = TRUE,
  control = list(),
  sparse = TRUE
)
```

Arguments

formula	a formula object. Random effects are specified using <code>s(f, bs = "re")</code> terms, where <code>f</code> is a factor, and smooths using other <code>s()</code> terms, see Details.
data	a <code>data.frame</code> containing all relevant info
pass	a vector of integers giving the pass number of the observation
id	a vector of integers identifying an observation (a set of electrofishing passes)
offset	an optional offset for the linear predictor of capture probability
verbose	if TRUE optimiser messages are printed to the screen
hessian	if TRUE the hessian is computed and the covariance matrix of the parameters is returned via <code>Vb</code>
fit	if TRUE model is fitted if FALSE the data that would be passed to the optimiser is returned
control	a list of control commands to be passed to nlminb
sparse	if TRUE (the default) penalised smooths are set up so that sparse bases and penalties stay sparse, see Details. If FALSE the constrained basis and penalties from <code>mgcv</code> are used, which are dense and so slower to fit.

Details

Random effects on the logit capture probability are included by adding `s(f, bs = "re")` terms to the formula, where `f` is a factor in data. For example, `s(visitID, bs = "re")` adds between visit variation and `s(sampleID, bs = "re")` adds between sample variation. Each term has its own standard deviation, and the random effects are integrated out of the likelihood using the Laplace approximation, so that `logLik` and `AIC` are based on the marginal likelihood. Fitted values are conditional on the estimated random effects.

Smooth terms can also be included in the formula. These are penalised, and the amount of smoothing is estimated: a smooth is fitted as a random effect, in which the part of the smooth that is not penalised (for example a linear trend) is a fixed effect, and the standard deviation of the random part controls how smooth the term is. A standard deviation close to zero means that only the unpenalised part of the smooth is needed. As for other random effects, `logLik` and `AIC` are based on the marginal likelihood.

The recommended smooths are first and second order random walks, using the Markov random field basis of `mgcv` with a penalty from `rw_penalty`: `s(f, bs = "mrf", xt = list(penalty = rw_penalty(levels(f))))`, where `f` is a factor whose levels are the steps of the walk. The basis and the penalty are sparse, and are kept sparse when fitting, so these models are quick to fit even when there are many levels. The standard deviation is that of the steps of the random walk, and the starting values of the walk (the first level for a first order walk, the first two for a second order walk) are fixed effects: this means that the intercept is the logit capture probability at the first level of the walk, and that a second order walk has an additional fixed effect for its linear trend, which is on an arbitrary scale.

Any other penalised smooth from `mgcv` can be used, such as `s(x)` or `te(x, z)`, but their penalties are dense so they will take longer to fit, particularly if the basis dimension `k` is large. To fit an unpenalised regression spline, whose flexibility is set entirely by `k`, use `fx = TRUE`, for example `s(x, k = 3, fx = TRUE)`.

The fixed effects are estimated by maximum likelihood, so the likelihood depends slightly on how the unpenalised part of a smooth is separated from the penalised part. With `sparse = TRUE` one coefficient of the smooth is fixed for each function that is not penalised, as described above for random walks. With `sparse = FALSE`, and for smooths with by variables or models without an intercept, the identifiability constraints of `mgcv` are applied to the smooth first, which makes the basis and the penalty dense. The two approaches give very similar, but not identical, fits.

Value

If `fit = TRUE`, an object of class `efp2`, a list which inherits from `glm`, containing among other things the fixed effect estimates (`coefficients`), the random effect estimates (`random`), the standard deviations of the random effects and smooths (`sigma`), the fitted capture probabilities (`fitted`), the log likelihood (`llik`) and the variance matrix of the fixed effects (`Vb`). See [efp2-methods](#) for the methods available. If `fit = FALSE`, a list containing the data that would be passed to the optimiser.

References

Millar, C. P., Fryer, R. J., Millidine, K. J. and Malcolm, I. A. (2016) Modelling capture probability of Atlantic salmon (*Salmo salar*) from a diverse national electrofishing dataset: Implications for the estimation of abundance. *Fisheries Research*, 177, 1-12. doi:10.1016/j.fishres.2016.01.001

Examples

```
# create two electrofishing site visits with 3 and 4 passes and 2 lifestages
ef2_data <- data.frame(n      = c(100, 53, 24, 50, 26, 12,
                                100, 53, 24, 50, 26, 12),
                      pass   = c( 1,  2,  3,  1,  2,  3,
                                1,  2,  3,  1,  2,  3),
                      stage  = c( 1,  1,  1,  2,  2,  2,
                                1,  1,  1,  2,  2,  2),
                      sample = c( 1,  1,  1,  2,  2,  2,
                                3,  3,  3,  4,  4,  4))

ef_data2 <- data.frame(n      = c(100, 53, 24, 50, 26, 12,
                                100, 53, 24, 12, 50, 26, 12, 6),
                      pass   = c( 1,  2,  3,  1,  2,  3,
                                1,  2,  3,  4,  1,  2,  3,  4),
                      stage  = c( 1,  1,  1,  2,  2,  2,
                                1,  1,  1,  1,  2,  2,  2,  2),
                      sample = c( 1,  1,  1,  2,  2,  2,
                                3,  3,  3,  3,  4,  4,  4,  4))

ef_data3 <- data.frame(n      = c(100, 53, 24, 50, 26, 12, 40,
                                100, 53, 24, 12, 50, 26, 12, 6, 40),
                      pass   = c( 1,  2,  3,  1,  2,  3,  1,
                                1,  2,  3,  4,  1,  2,  3,  4,  1),
                      stage  = c( 1,  1,  1,  2,  2,  2,  1,
                                1,  1,  1,  1,  2,  2,  2,  2,  2),
                      sample = c( 1,  1,  1,  2,  2,  2,  5,
                                3,  3,  3,  3,  4,  4,  4,  4,  6))

# Fit a simple model
m2 <- efp2(n ~ 1 + factor(stage), data = ef2_data, pass = pass, id = sample)
cbind(ef2_data, fit = fitted(m2))
m3 <- efp2(n ~ 1 + factor(stage), data = ef_data2, pass = pass, id = sample)
cbind(ef_data2, fit = fitted(m3))
m4 <- efp2(n ~ 1 + factor(stage), data = ef_data3, pass = pass, id = sample)
cbind(ef_data3, fit = fitted(m4))

# create two electrofishing site visits with 3 and 4 passes and 2 lifestages
ef2_data <- data.frame(n      = c(200, 53, 24, 100, 26, 12,
                                200, 53, 24, 100, 26, 12),
                      pass   = c( 1,  2,  3,  1,  2,  3,
                                1,  2,  3,  1,  2,  3),
                      stage  = c( 1,  1,  1,  2,  2,  2,
                                1,  1,  1,  2,  2,  2),
                      sample = c( 1,  1,  1,  2,  2,  2,
                                3,  3,  3,  4,  4,  4))

# Fit a simple model
m2 <- efp2(n ~ 1 + factor(stage) + factor(replace(pass, pass > 2, 2)),
          data = ef2_data, pass = pass, id = sample)
out <- cbind(ef2_data, p = fitted(m2, type = "p"))
out
```

```

# a smooth effect of a continuous covariate, day of the year, using a
# second order random walk
data(ef2_data)
ef2_data$sampleID <- factor(paste(ef2_data$date, ef2_data$siteID,
                                ef2_data$lifestage, ef2_data$species))
ef2_data$pass12 <- factor(replace(ef2_data$pass, ef2_data$pass > 2, 2))
days <- min(ef2_data$doy):max(ef2_data$doy)
ef2_data$fday <- factor(ef2_data$doy, levels = days)
rw2 <- rw_penalty(days, order = 2)
m5 <- efp2(count ~ siteID + pass12 + lifestage +
           s(fday, bs = "mrf", xt = list(penalty = rw2)),
           data = ef2_data, pass = pass, id = sampleID)
m5

# predicted capture probability over the smooth, with standard errors
newdata <- data.frame(siteID = "Site1", pass12 = "1", lifestage = "Fry",
                      fday = factor(days, levels = days))
pred <- predict(m5, newdata = newdata, type = "response", se.fit = TRUE)
cbind(newdata, fit = pred$fit, se = pred$se.fit)

# an unpenalised regression spline
m6 <- efp2(count ~ siteID + pass12 + lifestage + s(doy, k = 3, fx = TRUE),
           data = ef2_data, pass = pass, id = sampleID)
m6

```

fitted.efp2

Methods for efp2 fits

Description

Methods to summarise and extract information from a model fitted using [efp2](#).

Usage

```

## S3 method for class 'efp2'
fitted(object, type = "p", ...)

## S3 method for class 'efp2'
vcov(object, ...)

## S3 method for class 'efp2'
confint(object, parm, level = 0.95, ...)

## S3 method for class 'efp2'
logLik(object, ...)

## S3 method for class 'efp2'
formula(x, ...)

```

```
## S3 method for class 'efp2'
summary(object, ...)

## S3 method for class 'summary.efp2'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'efp2'
residuals(object, ...)

## S3 method for class 'efp2'
anova(object, ...)

## S3 method for class 'efp2'
plot(x, ...)

## S3 method for class 'efp2'
model.matrix(object, ...)

## S3 method for class 'efp2'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

<code>object, x</code>	an efp2 fitted model
<code>type</code>	the type of fitted value: "p" for the capture probability in each pass, "pi" for the probability that a fish is first caught in each pass, "both" for a list containing both and "lpmatrix" for the design matrix.
<code>...</code>	further arguments, ignored unless stated
<code>parm</code>	the parameters to compute confidence intervals for, either a vector of numbers or names. The default is all fixed effects.
<code>level</code>	the confidence level required
<code>digits</code>	the number of significant digits to print

Details

The variance matrix returned by `vcov` is for the fixed effects only, the joint variance matrix of the fixed and random effects is available as `object$Vp`. Confidence intervals are Wald intervals based on this variance matrix.

The log likelihood is the marginal log likelihood when the model contains random effects or penalised smooths, and the degrees of freedom are the number of fixed effects plus the number of standard deviations.

Residuals, analysis of deviance tables and diagnostic plots are not available for efp2 fits, and the corresponding methods return an error.

Value

fitted returns a vector, list or matrix depending on type; model.matrix a matrix; vcov a matrix; confint a matrix with columns giving the lower and upper confidence limits; logLik an object of class logLik; formula the model formula; summary an object of class summary.efp2, a list containing the coefficient table (coefficients) and the table of random effect standard deviations (sigma). The print methods return their argument invisibly.

Examples

```
data(ef2_data)
ef2_data$sampleID <- factor(paste(ef2_data$date, ef2_data$siteID,
                                ef2_data$lifestage, ef2_data$species))
m1 <- efp2(count ~ lifestage, data = ef2_data, pass = pass, id = sampleID)

summary(m1)
vcov(m1)
confint(m1)
logLik(m1)
head(fitted(m1))
```

predict.efp2

Predict capture probabilities from an efp2 fit

Description

Predictions of capture probability from a fitted efp2 model, for the data used to fit the model or for new data.

Usage

```
## S3 method for class 'efp2'
predict(
  object,
  newdata = NULL,
  type = c("link", "response", "lpmatrix"),
  se.fit = FALSE,
  exclude = NULL,
  ...
)
```

Arguments

object	an efp2 fitted model
newdata	a data.frame containing the covariates in the model formula. If NULL, predictions are for the data used to fit the model.

type	the type of prediction: "link" for the logit capture probability, "response" for the capture probability and "lpmatrix" for the design matrix which gives the logit capture probability when multiplied by the model parameters.
se.fit	if TRUE standard errors of the predictions are also returned
exclude	a character vector of terms to leave out of the prediction, named as in the model output, for example "s(visitID)". This is the way to predict for levels of a random effect that were not in the data used to fit the model. Variables that are only used by excluded smooth or random effect terms need not be supplied in newdata.
...	further arguments, ignored

Details

Predictions are conditional on the estimated random effects, unless these are removed using `exclude`. For a random walk smooth, predictions can be made for any level in the penalty, including levels with no data, which are interpolated. Standard errors on the response scale are calculated using the delta method.

The offset supplied to `efp2` is included in the predictions only when `newdata` is NULL.

Value

a vector of predictions, or if `se.fit = TRUE` a list with elements `fit` and `se.fit`. If `type = "lpmatrix"` a matrix.

Examples

```
data(ef2_data)
ef2_data$sampleID <- factor(paste(ef2_data$date, ef2_data$siteID,
                                ef2_data$lifestage, ef2_data$species))
m1 <- efp2(count ~ lifestage, data = ef2_data, pass = pass, id = sampleID)

newdata <- data.frame(lifestage = c("Fry", "Parr"))
predict(m1, newdata = newdata, type = "response", se.fit = TRUE)
```

rw_penalty

Random walk penalties for smooth terms

Description

Creates a first or second order random walk penalty matrix for the levels of an ordered covariate, for use with the Markov random field (mrf) basis of `mgcv`. This is the recommended way to include smooth terms in `efp2`: the basis and the penalty are both sparse, so models are quick to fit even when there are many levels.

Usage

```
rw_penalty(levels, order = 1)
```

Arguments

levels	the levels of the covariate, in order, and assumed to be equally spaced. A level is needed for every step of the random walk, including any for which there are no data.
order	the order of the random walk, 1 or 2

Details

A first order random walk (order = 1) penalises the differences between neighbouring levels, and shrinks towards a constant. A second order random walk (order = 2) penalises the second differences, and shrinks towards a linear trend. In both cases the standard deviation estimated by [efp2](#) is the standard deviation of the steps of the random walk, on the logit scale.

To use the penalty, the covariate must be a factor with the same levels as the penalty, see the examples.

Value

a square matrix with one row and column for each level, with the levels as dimnames. The matrix has an attribute, `null.space`, which is a basis for the functions which are not penalised.

Examples

```
rw_penalty(1:5)
rw_penalty(1:5, order = 2)

# a first order random walk over day of the year
data(ef2_data)
ef2_data$sampleID <- factor(paste(ef2_data$date, ef2_data$siteID,
                                ef2_data$lifestage, ef2_data$species))
days <- min(ef2_data$doy):max(ef2_data$doy)
ef2_data$fdoy <- factor(ef2_data$doy, levels = days)
rw1 <- rw_penalty(days)

m1 <- efp2(count ~ lifestage + s(fdoy, bs = "mrf", xt = list(penalty = rw1)),
           data = ef2_data, pass = pass, id = sampleID)
m1
```

Index

* datasets

ef2_data, [7](#)

abundance, [3](#), [6](#)

anova.efp2 (fitted.efp2), [11](#)

as.gam2, [5](#)

capture_prob, [3](#), [4](#), [6](#)

confint.efp2 (fitted.efp2), [11](#)

ef2 (ef2-package), [2](#)

ef2-package, [2](#)

ef2_data, [7](#)

efp2, [2](#), [6](#), [8](#), [11](#), [14](#), [15](#)

efp2-methods (fitted.efp2), [11](#)

fitted.efp2, [11](#)

formula.efp2 (fitted.efp2), [11](#)

gamObject, [5](#)

logLik.efp2 (fitted.efp2), [11](#)

model.matrix.efp2 (fitted.efp2), [11](#)

nlminb, [8](#)

plot.efp2 (fitted.efp2), [11](#)

predict.efp2, [2](#), [5](#), [13](#)

print.efp2 (fitted.efp2), [11](#)

print.summary.efp2 (fitted.efp2), [11](#)

residuals.efp2 (fitted.efp2), [11](#)

rw_penalty, [9](#), [14](#)

summary.efp2 (fitted.efp2), [11](#)

vcov.efp2 (fitted.efp2), [11](#)