# Package 'ef'

May 8, 2026

**Title** Modelling Framework for the Estimation of Salmonid Abundance

**Version** 1.2.0

**Date** 2020-10-18

**Description** A set of functions to estimate capture probabilities and densities from multipass pass removal data.

**Depends** R (>= 3.0), TMB

**Imports** Matrix, dplyr, methods, mgcv

**LinkingTo** TMB, RcppEigen

**Suggests** testthat

**License** MIT + file LICENSE

**LazyData** TRUE

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**NeedsCompilation** yes

**Author** Colin Millar [aut, cre],
Rob Fryer [aut],
Iain Malcolm [aut],
Ross Glover [aut],
Marine Scotland Science [cph]

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

**Repository** CRAN

**Date/Publication** 2020-11-03 12:10:02 UTC

## Contents

---

`ef-package` *Modelling Framework for the Estimation of Salmonid Abundance in Scottish Rivers*

---

## Description

A set of functions to estimate capture probabilities and densities from multipass pass removal data.

---

`as.gam` *Coversion of an efp fit to a gam fit for plotting*

---

## Description

Coversion of an efp fit to a gam fit for plotting

## Usage

```
as.gam(object)
```

## Arguments

| | |
|---|---|
| `object` | an efp fitted model |

## Value

gam type object

`BICadj` *Adjusted BIC function*

## Description

Complete function for returning overdispersion estimates

## Usage

```
BICadj(model, data, overdispersion.output)
```

## Arguments

| | |
|---|---|
| `model` | a fitted ef model |
| `data` | the data used to fit the model |
| `overdispersion.output` | output form the function overdispersion |

## Value

the adjusted BIC

## Note

When calling the function, you need to specify the data source for the model so that the number of site visits can be determined. You also need to specify the output from the overdispersion model to get the measure

`efp` *Estimate capture probabilites from electrofishing data*

## Description

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

## Usage

```
efp(
  formula,
  data = NULL,
  pass = pass,
  id = id,
  offset = NULL,
  verbose = FALSE,
  init = "0",
```

```
  hessian = TRUE,
  fit = TRUE,
  sample_re = FALSE
)
```

## Arguments

| | |
|---|---|
| formula | a formula object |
| data | a data.frame containing all relavent 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 possible offset for the linear predictor of capture probability |
| verbose | if TRUE stan optimiser messages are printed to the screen |
| init | should initialisatiom be random? |
| hessian | if TRUE the hessian is computed and the covariance matrix of the parameters is returned via Vb |
| fit | if TRUE model is fitted if FALSE the data that would be passed to the optimiser is returned |
| sample_re | should sample random effects be included |

## Value

glm type object

## Examples

```
# create two electrofishing site visits with 3 and 4 passes and 2 lifestages
ef_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 <- efp(n ~ 1 + factor(stage), data = ef_data, pass = pass, id = sample)
cbind(ef_data, fit = fitted(m2))
m3 <- efp(n ~ 1 + factor(stage), data = ef_data2, pass = pass, id = sample)
cbind(ef_data2, fit = fitted(m3))
m4 <- efp(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
ef_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 <- efp(n ~ 1 + factor(stage) + factor(replace(pass, pass> 2, 2)),
          data = ef_data, pass = pass, id = sample)
out <- cbind(ef_data, p = fitted(m2, type = "p"))
out
```

---

`ef_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

```
ef_data
```

## Format

A data frame with 90 rows and 9 variables:

**siteID** Unique identifier for each elecrofishing site

**year** year of data collection

**date** date of data collections

**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://www2.gov.scot/Topics/marine/Salmon-Trout-Coarse/Freshwater/Monitoring/temperature

---

## invlogit *Inverse logistic transformation*

---

### Description

This function transforms values on the logistic scale to values on the probability scale.

### Usage

```
invlogit(x)
```

### Arguments

| | |
|---|---|
| x | a numeric vector |

### Value

vector of values between 0 and 1

---

## logit *Logistic transformation*

---

### Description

This function transforms values on the probability scale to values on the logistic scale.

### Usage

```
logit(p)
```

### Arguments

| | |
|---|---|
| p | a numeric vector with values between 0 and 1 |

### Value

vector of values between -Inf and Inf

`overdispersion` *Estimating overdispersion*

## Description

Complete function for returning overdispersion estimates

## Usage

```
overdispersion(
  data,
  siteID,
  visitID,
  count = "count",
  pass = "pass",
  lifestage = "lifestage",
  pass12 = "pass12",
  id,
  largemodel
)
```

## Arguments

| | |
|---|---|
| `data` | dataframe containing EF data |
| `siteID` | site name or unique ID |
| `visitID` | a number identifying each unique visit |
| `count` | count of fish (defaults to "count") |
| `pass` | the EF pass number (defaults to "pass") |
| `lifestage` | the lifestage (defaults to "lifestage") |
| `pass12` | categorical variable with 2 levels where the 1st pass and subsequent passes are treated separately (defaults to "pass12") |
| `id` | sample ID |
| `largemodel` | the large model that captures most of the systematic variation in the data - this is specified before running the overdispersion function |

## Value

a data.frame summarising overdispersion

## Note

ensure column names in function call are in inverted commas

`transpar` *Utility function to convert parameters to probabilities*

## Description

The matrix G should be of dimension n x p, and the parameter vector should be length p

## Usage

```
transpar(par, G)
```

## Arguments

| | |
|---|---|
| `par` | fitted model parameters |
| `G` | The design matrix for a model |

## Value

a data frame

# Index