Package ‘emax.glm’

July 4, 2019

**Title**  General Tools for Building GLM Expectation-Maximization Models

**Version**  0.1.2

**Maintainer**  Robert M. Cook <robert.cook@bcu.ac.uk>

**Description**  Implementation of Expectation Maximization (EM) regression of general linear models.

The package currently supports Poisson and Logistic regression with variable weights, with underlying theory included in the vignettes.

New users are recommended to look at the em(glm) and small(em) functions - the outputs of which are supported by AIC(), BIC(), and logLik() calls.

Several plot functions have been included for useful diagnostics and model exploration.


**License**  GPL-3

**Encoding**  UTF-8

**LazyData**  true

**RoxygenNote**  6.1.1

**Depends**  R (>= 2.10)

**Imports**  MASS, pracma, stats, pander, pROC, AER

**Suggests**  spelling, knitr, rmarkdown

**VignetteBuilder**  knitr

**Language**  en-US

**NeedsCompilation**  no

**Author**  Robert M. Cook [aut, cre] (<https://orcid.org/0000-0003-3343-8271>)

**Repository**  CRAN

**Date/Publication**  2019-07-04 08:30:03 UTC

**R topics documented:**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIC.em.glm</td>
<td>2</td>
</tr>
<tr>
<td>BIC.em.glm</td>
<td>3</td>
</tr>
</tbody>
</table>
AIC.em.glm

Calculate the AIC of the em.glm model

Description

Calculate the AIC of the em.glm model
Usage

```r
## S3 method for class 'em.glm'
AIC(object, ..., k = 2)
```

Arguments

- `object`: A `em.glm` class returned by the `em.glm` function.
- `...`: optionally more fitted model objects.
- `k`: numeric, the *penalty* per parameter to be used; the default `k = 2` is the classical AIC.

Value

The AIC score of the model.

Examples

```r
y <- c(AirPassengers)
n <- length(y)
x <- as.matrix(rep(1, n))
m <- em.glm(x = x, y = y, K = 2, b.init = "random")
AIC(m)
```

---

**BIC.em.glm**

*Calculate the BIC of the em.glm model*

Description

Calculate the BIC of the em.glm model

Usage

```r
## S3 method for class 'em.glm'
BIC(object, ...)
```

Arguments

- `object`: A `em.glm` class returned by the `em.glm` function.
- `...`: optionally more fitted model objects.

Value

The BIC score of the model.
Examples

```r
y <- c(Airpassengers)
n <- length(y)
x <- as.matrix(rep(1L, n))
m <- em.glm(x = x, y = y, K = 2, b.init = "random")
BIC(m)
```
dispersion

Pearson-based dispersion measurements of an 'em.glm' model.

Description
Pearson-based dispersion measurements of an 'em.glm' model.

Usage
dispersion(em.glm, x, y, weight)

Arguments
- em.glm: An 'em.glm' object.
- x: An n-by-p design matrix.
- y: A vector of observation of length n.
- weight: A n length vector of observation weight terms. This is currently designed to be either the exposure for a Poisson model or the number of trials for a Logistic model.

Value
A list of dispersion parameters for the model.

Examples
x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks
m <- em.glm(x = x, y = y, K = 2, b.init = "random")
dispersion(m, x, y, weight = c(1))

dprob.list
List of distribution functions accessed by family name ("poisson" or "binomial").

Description
List of distribution functions accessed by family name ("poisson" or "binomial").

Usage
dprob.list

Format
An object of class list of length 2.
em.fit_numeric  

Carry out the Newton-Raphson optimization of the parameters for given weights via numeric approximations.

Description

Carry out the Newton-Raphson optimization of the parameters for given weights via numeric approximations.

Usage

```
em.fit_numeric(b, x, y, class_probs, weight = c(1), tol = 1e-08,
               debug = FALSE, family = poisson(), maxiter = Inf)
```

Arguments

- **b**: The starting parameters.
- **x**: An \(n\)-by-\(p\) design matrix.
- **y**: A vector of observation of length \(n\).
- **class_probs**: An \(n\) length vector of probabilities for the proposed model.
- **weight**: A \(n\) length vector of observation weight terms. This is currently designed to be either the exposure for a Poisson model or the number of trials for a Logistic model.
- **tol**: The tolerance to repeat the Newton-Raphson optimization till.
- **debug**: Debugging flag - set to TRUE to output step-by-step change in parameter values.
- **family**: The GLM family being considered.
- **maxiter**: Maximum number of NR steps to take.

Value

The parameter values on convergence.

Examples

```
x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks
u <- make.dpois(x, y)
b <- c(1, 1, 1, 1)
class_probs <- rep(1, 54)
em.fit_numeric(b = b, x=x, y=y, class_probs = class_probs)
```
**em.fit_pracma**

*Carry our the Newton-Raphson optimization of the parameters for given weights via the `pracma` hessian.*

**Description**

Carry out the Newton-Raphson optimization of the parameters for given weights via the `pracma` hessian.

**Usage**

```r
em.fit_pracma(u, b, x, y, class_probs, weight, tol = 1e-08,
debug = FALSE, family = poisson(), maxiter = Inf)
```

**Arguments**

- **u**
  A `model.loglike` function.
- **b**
  The starting parameters.
- **x**
  An \(n\)-by-\(p\) design matrix.
- **y**
  A vector of observation of length \(n\).
- **class_probs**
  An \(n\) length vector of probabilities for the proposed model.
- **weight**
  A \(n\) length vector of observation weight terms. This is currently designed to be either the exposure for a Poisson model or the number of trials for a Logistic model.
- **tol**
  The tolerance to repeat the Newton-Raphson optimization till.
- **debug**
  Debugging flag - set to TRUE to output step-by-step change in parameter values.
- **family**
  The GLM family being considered.
- **maxiter**
  Maximum number of NR steps to take.

**Value**

The parameter values on convergence.

**Examples**

```r
x <- model.matrix(~ 1 + factor(wool) + factor(tension), data = warpbreaks)
y <- warpbreaks$breaks
class_probs = rep(1,54)
b <- c(1, 1, 1, 1)

u <- make.loglike(x, y, class_probs = class_probs)
em.fit_pracma(u, b, x, y, class_probs, weight = c(1))
```
em.glm  Expectation Maximization glm.

Description

Fit an Expectation Maximization glm using the glm family to define the link function. Two methods of optimization are included, using direct numeric approximations and using the pracma package to find the Hessian and Jacobian of the log-likelihood. The number of competing models to be fit is set by $K$.

Usage

```r
em.glm(x, y, b.init = "random", weight = c(1), K = 2,
family = poisson, method = "numeric", maxiter = 50,
maxiter.NR = Inf, tol.1 = 1e-08, tol.2 = 1e-08, noise = 0.2,
depbug = FALSE, param_errors = FALSE)
```

Arguments

- **x**: An $n$-by-$p$ design matrix.
- **y**: A vector of observation of length $n$.
- **b.init**: The method to initialize EM parameters. Built in methods are "random" and "fit" for pure white noise, and white noise around GLM estimates. Alternatively, pass a list of length $K$, each element consisting of a vector of length $p$. Users can also pass a zero-argument function to produce starting states.
- **weight**: A $n$ length vector of observation weight terms. This is currently designed to be either the exposure for a Poisson model or the number of trials for a Logistic model.
- **K**: Number of EM classes to be fit.
- **family**: GLM family to fit.
- **method**: Control string. Set to 'numeric' or 'pracma'.
- **maxiter**: Maximum number of re-weighting rounds to do in fitting the EM model. Primarily used to perform the 'small.em' warm-up routine.
- **maxiter.NR**: Maximum number of Newton-Raphson steps to take.
- **tol.1**: Escape tolerance of the Newton-Raphson step.
- **tol.2**: Escape tolerance of the re-weighting step.
- **noise**: Standard deviation of the white noise to be applied when generating random initial states.
- **debug**: Returns step-size in NR and re-weighting steps as a message if TRUE.
- **param_errors**: Bool flag - set to TRUE to calculate parameter errors.
Details

It is recommend users first call the `em.small` command to run small warm up trials to explore the parameter space.

Value

An `em.glm` object containing the class parameters, and class weights.

References

Dempster et al (1977) Maximum Likelihood from Incomplete Data via the EM Algorithm

Examples

```r
x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks
m <- em.glm(x = x, y = y, K = 2, b.init = "random")
summary(m)
```

em.glm_numeric_fit

Numeric approximation routine

Description

Numeric approximation routine

Usage

```r
em.glm_numeric_fit(x, y, b.list, class_probs, weight = c(1), K = 2,
tol.1 = 1e-08, debug = TRUE, family = poisson(), maxiter = Inf)
```

Arguments

- `x` An \(n\)-by-\(p\) design matrix.
- `y` A vector of observation of length \(n\).
- `b.list` List of \(K\)-classes each entry being a \(k\) length parameter vector,
- `class_probs` Matrix \((n \times K)\) of normalized class probabilities.
- `weight` A \(n\) length vector of observation weight terms. This is currently designed to be either the exposure for a Poisson model or the number of trials for a Logistic model.
em.glm_pracma_fit

K
Number of EM classes to be fit.
tol.1
Tolerance of the NR minimization.
debug
Boolean flag. Turn on to check optimization steps via messages.
family
GLM family to fit with.
maxiter
Maximum iterations of the NR methods for exiting before convergence.

Value
A list of parameter values on convergence for each of k-classes.

Examples

```r
x <- model.matrix(~ 1, data = warpbreaks)
y <- warpbreaks$breaks
b.list <- list(1, 1)
class_probs = matrix(rep(0.5, 54*2), ncol = 2)
em glm numeric fit(x = x, y = y, b.list = b.list, class_probs = class_probs)
```

Description

Hessian routine

Usage

`em.glm_pracma_fit(x, y, b.list, class_probs, weight = c(1), K = 2,
tol.1 = 1e-08, debug = FALSE, family = poisson(), maxiter = Inf)`

Arguments

- `x`
  An n-by-p design matrix.
- `y`
  A vector of observation of length n.
- `b.list`
  List of K-classes each entry being a k length parameter vector,
- `class_probs`
  Matrix (n x K) of normalized class probabilities.
- `weight`
  A n length vector of observation weight terms. This is currently designed to be either the exposure for a Poisson model or the number of trials for a Logistic model.
- `K`
  Number of EM classes to be fit.
- `tol.1`
  Tolerance of the NR minimization.
- `debug`
  Boolean flag. Turn on to check optimization steps via messages.
- `family`
  GLM family to fit with.
- `maxiter`
  Maximum iterations of the NR methods for exiting before convergence.
Value

A list of parameter values on convergence for each of k-classes.

Examples

```r
x <- model.matrix(~ 1, data = warpbreaks)
y <- warpbreaks$breaks
b.list <- list(1L, 1)
class_probs = matrix(rep(0.5, 54*2), ncol = 2)
em.glm_pracma_fit(x = x, y = y, b.list = b.list, class_probs = class_probs)
```

**Description**

Implementation of Expectation Maximization (EM) regression of general linear models. The package currently supports Poisson and Logistic regression with variable weights, with underlying theory included in the vignettes. New users are recommended to look at the em.glm() and small.em() functions - the outputs of which are supported by AIC(), BIC(), and logLik() calls. Several plot functions have been included for useful diagnostics and model exploration. Methods are based on the theory of Dempster et al (1977, ISBN:00359246), and follow the methods of Hastie et al. (2009) <doi:10.1007/978-0-387-21606-5_7> and A. Zeileis et al (2017) <doi:10.18637/jss.v027.i08>.

**Author(s)**

**Maintainer:** Robert M. Cook <robert.cook@bcu.ac.uk> (0000-0003-3343-8271)

**IC.em.glm**

General Information Criteria function

**Description**

General Information Criteria function

**Usage**

```r
IC.em.glm(em.glm, alpha)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>em.glm</td>
<td>An emax glm fit.</td>
</tr>
<tr>
<td>alpha</td>
<td>Scaling factor for information criteria (2 or ln(n) for AIC and BIC respectively).</td>
</tr>
</tbody>
</table>
**Value**

The IC value of the model for given value of \( k \).

**init.fit**

*Method to initialize EM parameters. Carries out a single GLM fit and applies random noise to form starting space.*

**Description**

Method to initialize EM parameters. Carries out a single GLM fit and applies random noise to form starting space.

**Usage**

\[
\text{init.fit}(y, x, K, \text{weight} = c(1), \text{family} = \text{poisson()}, \text{noise} = 1)
\]

**Arguments**

- \( y \)  
  A vector of observation of length \( n \).
- \( x \)  
  An \( n \)-by-\( p \) design matrix.
- \( K \)  
  Number of EM classes to be fit.
- \( \text{weight} \)  
  A \( n \) length vector of observation weight terms. This is currently designed to be either the exposure for a Poisson model or the number of trials for a Logistic model.
- \( \text{family} \)  
  GLM family to fit.
- \( \text{noise} \)  
  Standard deviation of the white noise to be applied when generating random initial states.

**Value**

A \( K \)-length list, each holding parameters.

**Examples**

\[
x <- \text{model.matrix}(~ 1 + \text{factor(wool)} + \text{factor(tension)}, \text{data} = \text{warpbreaks})
y <- \text{warpbreaks}$breaks
\]

\[
\text{init.fit}(y = y, x = x, K = 2)
\]
**init.random**

Method to initialize EM parameters. Purely standard normal noise.

**Description**

Method to initialize EM parameters. Purely standard normal noise.

**Usage**

```r
init.random(x, K, noise = 1)
```

**Arguments**

- `x`: An `n`-by-`p` design matrix.
- `K`: Number of EM classes to be fit.
- `noise`: Standard deviation of the white noise to be applied when generating random initial states.

**Value**

A `K`-length list, each holding parameters.

**Examples**

```r
x <- model.matrix(~ 1 + factor(wool) + factor(tension), data = warpbreaks)
init.random(x = x, K = 2)
```

**logLik.em.glm**

Calculate log-likelihood of the EM model.

**Description**

Calculate log-likelihood of the EM model.

**Usage**

```r
## S3 method for class 'em.glm'
logLik(object, x, y, weight = c(1), ...)
```
Arguments

object A \`em.glm\' class returned by the \em.glm\ function.
x An \(n\)-by-\(p\) design matrix.
y A vector of observation of length \(n\).
weight A \(n\) length vector of observation weight terms. This is currently designed to be either the exposure for a Poisson model or the number of trials for a Logistic model.

Value

Model log-likelihood.

Examples

\begin{verbatim}
x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks
m <- \em.glm\(x = x, y = y, K = 2, b.init = \"random\"
logLik(m, x, y)
\end{verbatim}

---

\textbf{make.dbinom}  \hspace{1cm} \textit{Build a Binomial log likelihood}

Description

Build a Binomial log likelihood

Usage

\texttt{make.dbinom(x, y, linkinv = binomial()}$linkinv, weight = 1,
\texttt{log = FALSE)}

Arguments

x An \(n\)-by-\(p\) design matrix.
y A vector of observation of length \(n\).
linkinv Inverse link function desired
weight A \(n\) length vector of observation weight terms. This is currently designed to be either the exposure for a Poisson model or the number of trials for a Logistic model.
log Boolean flag. If TRUE returns the log dist.

Value

A function to calculate (log) likelihood for a given set of parameters under a Binomial model.
Examples

```r
x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks
make.dbinom(x, y)
```

### Description

Build a Poisson log likelihood

### Usage

```r
make.dpois(x, y, linkinv = poisson()$linkinv, weight = c(1),
log = FALSE)
```

### Arguments

- **x**: An $n$-by-$p$ design matrix.
- **y**: A vector of observation of length $n$.
- **linkinv**: Inverse link function desired
- **weight**: A $n$ length vector of observation weight terms. This is currently designed to be either the exposure for a Poisson model or the number of trials for a Logistic model.
- **log**: Boolean flag. If TRUE returns the log dist.

### Value

A function to calculate (log) likelihood for a given set of parameters under a Poisson model.

### Examples

```r
x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks
make.dpois(x, y)
```
make_loglike

Construct a log-likelihood function in the parameters \( b \), for the given link family.

**Description**

Construct a log-likelihood function in the parameters \( b \), for the given link family.

**Usage**

```r
make.logLike(x, y, weight = c(1), class_probs = c(1),
family = poisson)
```

**Arguments**

- `x`: An \( n \times p \) design matrix.
- `y`: A vector of observation of length \( n \).
- `weight`: A \( n \) length vector of observation weight terms. This is currently designed to be either the exposure for a Poisson model or the number of trials for a Logistic model.
- `class_probs`: An \( n \) length vector of probabilities for the proposed model.
- `family`: The GLM family being considered.

**Value**

A model likelihood function. Expects one argument which is a \( p \) length vector of parameters.

**Examples**

```r
x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks
make.logLike(x, y)
```

make_param_errors

Calculate parameter errors via inversion of the Hessian matrix (either pracma or numeric approximations).

**Description**

Calculate parameter errors via inversion of the Hessian matrix (either pracma or numeric approximations).
Usage

```
make_param_errors(params, x, y, weight, family = poisson(),
                  method = "numeric", dispersion = 1)
```

Arguments

- **params**: Optimal parameters
- **x**: An $n$-by-$p$ design matrix.
- **y**: A vector of observation of length $n$.
- **weight**: A length vector of observation weight terms. This is currently designed to be either the exposure for a Poisson model or the number of trials for a Logistic model.
- **family**: GLM family to fit.
- **method**: Control string. Set to 'numeric' or 'pracma'.
- **dispersion**: Model dispersion parameter for over/under-dispersed models. Defaults to 1.

Value

Calculate the errors associated with each set of parameters.

Examples

```
x <- model.matrix(~ factor(wool) + factor(tension), warpbkreaks)
y <- warpbkreaks$breaks
m <- em.glm(x = x, y = y, K = 2, b.init = "random")
make_param_errors(m$params, x = x, y = y, weight = c(1))
```

---

**plot.em.glm**

*Plot fit-parameters and errors*

Description

Plot fit-parameters and errors

Usage

```
# S3 method for class 'em.glm'
plot(x, known_params = NULL, plot_type = lines,
      add = FALSE, ...)
```
Arguments

x An em.glm fit object.
known_params Prior estimates of fit parameters for comparison.
plot_type The plot type to display. Defaults to lines, alternative include points.
add Boolean flag to decide if the plot should be added to an existing displayed plot
object or create a new axes.
... Arguments to be passed to methods

Examples

```r
x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks
m <- em.glm(x = x, y = y, K = 2, b.init = "random")
plot(m)
```

Description

Error bar plot of coefficients and errors to inspect class overlap.

Usage

```r
# S3 method for class 'em.glm.summary'
plot(x, ...)
```

Arguments

x An EM summary object, from summary.em.glm
... Arguments to be passed to methods

Value

An R plot with error bars.

Examples

```r
x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks
m <- em.glm(x = x, y = y, K = 2, b.init = "random")
m.sum <- summary(m)
plot(m.sum)
```
plot_probabilities  

**Description**

Probability plots for the K classes fit

**Usage**

```r
plot_probabilities(...)  
```

**Arguments**

... Arguments to be passed to methods

---

**plot_probabilities.em.glm**  

*Test Plot em.glm*

**Description**

Test Plot em.glm

**Usage**

```r
## S3 method for class 'em.glm'
plot_probabilities(em.glm, ...)  
```

**Arguments**

- `em.glm` An `em.glm` object. From `em.fit`
- ... Associated arguments to be passed to `plot::par` function.

**Examples**

```r
x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)  
y <- warpbreaks$breaks  

m <- em.glm(x = x, y = y, K = 2, b.init = "random")  

plot_probabilities(m)  
```
plot_probabilities.matrix

Plot the class probabilities, both compared to data set index and as histogram.

Description

Plot the class probabilities, both compared to data set index and as histogram.

Usage

## S3 method for class 'matrix'
plot_probabilities(class_probabilities, ...)

Arguments

class_probabilities
    Matrix of n x K class probabilities
...
    Associated arguments to be passed to plot::par function.

predict.em.glm

Predict values from an 'em.glm' model.

Description

Predict values from an 'em.glm' model.

Usage

## S3 method for class 'em.glm'
predict(object, x, y, weight, type = "count", ...)

Arguments

object
    An em.glm fit object.

x
    An n-by-\( p \) design matrix.

y
    A vector of observation of length \( n \).

weight
    A \( n \) length vector of observation weight terms. This is currently designed to be either the exposure for a Poisson model or the number of trials for a Logistic model.

type
    Prediction type. Currently can be 'count' for the weighted prediction, 'rate' for the expected rate or 'rho' for the linear predictor.
...
    optionally more fitted model objects.
residuals.em.glm

Value

N-length vector of predicted terms.

Examples

```r
x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks
m <- em.glm(x = x, y = y, K = 2, b.init = "random")
predict(m, x = x, y = y, weight = c(1))
```

residuals.em.glm

Deviance residuals for an 'em.glm' object.

Description

Deviance residuals for an 'em.glm' object.

Usage

```r
## S3 method for class 'em.glm'
residuals(object, x, y, weight = c(1),
          type = "deviance", ...)
```

Arguments

- **object**: An 'em.glm' object.
- **x**: An n-by-p design matrix.
- **y**: A vector of observation of length n.
- **weight**: A n length vector of observation weight terms. This is currently designed to be either the exposure for a Poisson model or the number of trials for a Logistic model.
- **type**: Residual type - either deviance or Pearson's residuals.
- **...**: other arguments.

Value

An n length vector of residuals.
Examples

```r
x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks

m <- em.glm(x = x, y = y, K = 2, b.init = "random")
residuals(m, x = x, y = y)
```

---

**results_k25_n1000**  
*Simulated data set*

**Description**

Simulated data set

**Usage**

```r
results_k25_n1000
```

**Format**

An object of class `matrix` with 3 rows and 50 columns.

---

**results_k25_n1000_e05**  
*Simulated data set*

**Description**

Simulated data set

**Usage**

```r
results_k25_n1000_e05
```

**Format**

An object of class `matrix` with 3 rows and 50 columns.
results_simple

Simulated data set

Description

Simulated data set

Usage

results_simple

Format

An object of class matrix with 3 rows and 50 columns.

select_best

Select the best parameters from a set of results

Description

Select the best parameters from a set of results
Return the optimal model based on BIC scores

Usage

select_best(small.em)

Arguments

small.em A 'small.em' object

Value

The parameters of the best model, as judged by log-likelihood.

Examples

x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks

warm_up <- small.em(x = x, y = y, K = 2, b.init = "random", sample.size = 20)

select_best(warm_up)
Simulated data set

### sim.1

**Description**
Simulated data set

**Usage**
sim.1

**Format**
An object of class list of length 2.

### sim.2

**Description**
Simulated data set

**Usage**
sim.2

**Format**
An object of class list of length 3.

### sim.3

**Description**
Simulated data set

**Usage**
sim.3

**Format**
An object of class list of length 5.
Description

Carry out several short EM fits to test for optimal starting locations.

Usage

```r
small.em(x, y, b.init = "fit", weight = c(1), K = 2, maxiter = 5,
      tol.1 = 1e-04, tol.2 = 1e-04, noise = 0.2, sample.size = 500,
      repeats = 5, debug = FALSE, family = "poisson",
      method = "numeric", maxiter.NR = 20)
```

Arguments

- **x**: An n-by-p design matrix.
- **y**: A vector of observation of length n.
- **b.init**: The method to initialize EM parameters. Built-in methods are "random" and "fit" for pure white noise, and white noise around GLM estimates. Alternatively, pass a list of length K, each element consisting of a vector of length p. Users can also pass a zero-argument function to produce starting states.
- **weight**: A n length vector of observation weight terms. This is currently designed to be either the exposure for a Poisson model or the number of trials for a Logistic model.
- **K**: Number of EM classes to be fit.
- **maxiter**: Maximum number of re-weighting rounds to do in fitting the EM model. Primarily used to perform the ‘small.em’ warm-up routine.
- **tol.1**: Escape tolerance of the Newton-Raphson step.
- **tol.2**: Escape tolerance of the re-weighting step.
- **noise**: Standard deviation of the white noise to be applied when generating random initial states.
- **sample.size**: Number of cases to randomly select from the input data.
- **repeats**: Number of repetitions of the initialization to make.
- **debug**: Returns step-size in NR and re-weighting steps as a message if TRUE.
- **family**: GLM family to fit.
- **method**: Control string. Set to ’numeric’ or ’pracma’.
- **maxiter.NR**: Maximum number of Newton-Raphson steps to take.

Value

A ‘small.em’ list containing the parameters, weights, log likelihood and BIC values.
Examples

x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks

warm_up <- small.em(x = x, y = y, K = 2, b.init = "random", sample.size = 50)
summary(warm_up)

params <- select_best(warm_up)

m <- em.glm(x = x, y = y, K = 2, b.init = params)
summary(m)

summary.em.glm

Summarize EM glm coefficients.

Description

Summarize EM glm coefficients.

Usage

## S3 method for class 'em.glm'
summary(object, ...)

Arguments

object An em.glm object
...
additional arguments affecting the summary produced

Value

List of each classes coefficients and errors.

Examples

x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks

m <- em.glm(x = x, y = y, K = 2, b.init = "random")
summary(m)
summarize.small.em  

**Summarize a small.em class**

**Description**
Summarize a small.em class

**Usage**
```r
## S3 method for class 'small.em'
summary(object, ...) 
```

**Arguments**
- `object` A small.em class
- `...` additional arguments affecting the summary produced

**Value**
A data frame of log-likelihood, BIC and model index.

**Examples**
```r
x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks
warm_up <- small.em(x = x, y = y, K = 2, b.init = "random", sample.size = 50)
summary(warm_up)
```

update_probabilities  

**Construct normalized class properties for a given set of parameters**

**Description**
Construct normalized class properties for a given set of parameters

**Usage**
```r
update_probabilities(dprob, params)
```

**Arguments**
- `dprob` Probability distribution function to call. See 'dprob.list' for examples.
- `params` List of class parameters. Length of list is number of classes
Value

A n-by-k matrix of class probabilities (each row normalized to 1).

Examples

```r
x <- model.matrix(~ factor(wool) + factor(tension), warpbreaks)
y <- warpbreaks$breaks
dprob <- make.dpois(x = x, y = y)
params <- list(rep(1, 4))

update_probabilities(dprob, params)
```
Index

*Topic datasets
  data.1, 4
dprob.list, 5
results_k25_n1000, 22
results_k25_n1000_e05, 22
results_simple, 23
sim.1, 24
sim.2, 24
sim.3, 24
AIC.em.glm, 2
BIC.em.glm, 3
data.1, 4
deviance.em.glm, 4
dispersion, 5
dprob.list, 5
e.fit_numeric, 6
e.fit_pracma, 7
e.em, 8
e.em_numeric_fit, 9
e.em_pracma_fit, 10
e.max.em, 11
e.max.em_package (emax.glm), 11
IC.em.glm, 11
init.fit, 12
init.random, 13
logLik.em.glm, 13
make.dbinom, 14
make.dpois, 15
make.loglike, 16
make_param_errors, 16
plot.em.glm, 17
plot.em.glm.summary, 18
plot_probabilities, 19
plot_probabilities.em.glm, 19
plot_probabilities.matrix, 20
predict.em.glm, 20
residuals.em.glm, 21
results_k25_n1000, 22
results_k25_n1000_e05, 22
results_simple, 23
select_best, 23
sim.1, 24
sim.2, 24
sim.3, 24
small.em, 25
summary.em.glm, 26
summary.small.em, 27
update_probabilities, 27