NTNU

Kunnskap for en bedre verden

Institutt for matematiske fag

Eksamensoppgave i ST2304 Statistisk modellering for biologer og bioteknologer

Faglig kontakt under eksamen: Ola H. Diserud

Tlf.: 93 21 88 23

Eksamensdato: 11. august 2017

Eksamenstid (fra-til): 9-13

Hjelpemiddelkode/Tillatte hjelpemidler: Et gult A4-ark med egne håndskrevne notater, godkjent kalkulator, *Tabeller og formler i statistikk* (Tapir forlag), *Matematisk formelsamling* (K. Rottmann)

Annen informasjon: Hjelpesider for noen R funksjoner du kan få bruk for finnes i vedlegget. Alle svar skal begrunnes, og besvarelsen skal inneholde tilstrekkelig mellomregning.

Målform/språk: Bokmål
Antall sider: 4
Antall sider vedlegg: 4

Kontrollert av:

Dato Sign

Informasjon om trykking av eksamensoppgave

Originalen er:

1-sidig ☐ **2-sidig** X

sort/hvit X **farger** ☐

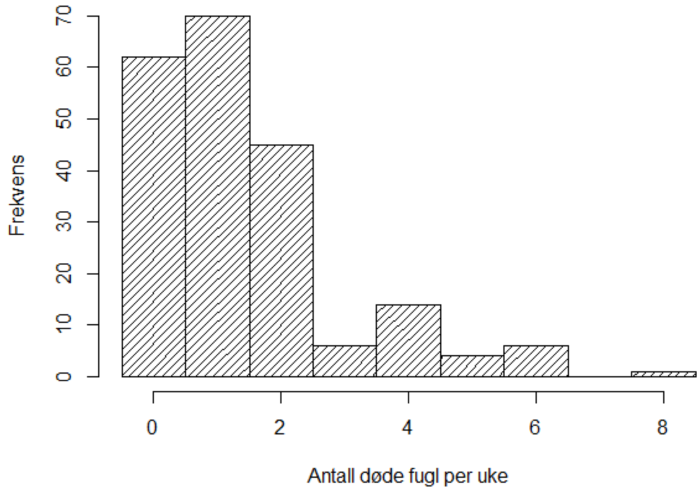
skal ha flervalgskjema ☐

Oppgave 1

Et problem med vindmøller til kraftproduksjon er at fugler kan kollidere med rotorbladene og bli alvorlig skadet. Som et prøveprosjekt monteres ei mølle et sted langs norskekysten, og det registreres ukentlig hvor mange fugler som blir funnet døde fra kollisjonsskader rundt mølla. Erfaring fra Danmark med ei lik mølle gir et forventet antall døde fugler per uke på 1,1.

a) Kan vi anta at antallet fugl som dør av kollisjon ved mølla per uke er en Poisson-fordelt tilfeldig variabel med konstant forventningsverdi (λ)? Begrunn svaret.

I løpet av fire år blir det funnet til sammen 298 kollisjonsdøde fugl rundt den norske mølla. Ukentlig antall døde fugl vises i figur 1. Estimer forventet antall kollisjonsdøde fugl per uke (λ) for den norske vindmølla.



Figur 1. Ukentlig antall fugl døde av kollisjonsskader funnet rundt ei norsk vindmølle.

b) Skriv et R-uttrykk som beregner sannsynligheten for å finne 7 eller flere kollisjonsdøde fugler rundt mølla ei uke i november. Skriv også et R-uttrykk som beregner sannsynligheten for å finne mer enn 100 døde fugler i løpet av et år.

Sannsynligheten for å observere totalt 298 kollisjonsdøde fugl i løpet av 4 år med en ukentlig forventning som i Danmark på 1,1, er svært lav (<0.00001). Hva forteller dette oss om å bruke data fra ei mølle på møller plassert andre steder? Spekuler kort rundt hvilke lokale forhold som kan påvirke forventet antall kollisjonsdøde fugl.

c) La *T* være tid mellom to etterfølgende kollisjoner. Dere kan her benytte at *T* er eksponensialfordelt. Skriv et R-uttrykk som beregner sannsynligheten for at det går mer enn 1 uke mellom to etterfølgende kollisjoner.

Forventningsverdien til en tilfeldig variabel kan tenkes på som grenseverdien for gjennomsnittet hvis vi kan gjenta eksperimentet mange ganger. Skriv et R-uttrykk for hvordan du kan estimere forventningsverdien, 5%-kvantilen og 95%-kvantilen til eksponensialfordelingen med rate = 1 / λ.

Oppgave 2

Likestillingsombudet påstår at kvinner har mindre sjanse for å få jobb hos et bestemt firma enn menn med samme bakgrunn og erfaring. Data på utdanning (antall år), erfaring (antall år) og kjønn (mann=1, kvinne=0) ble samlet inn for 28 tidligere søkere. I tillegg ble det registrert om søkeren ble ansatt i jobben eller ikke (ansatt=1, ikke ansatt=0). Datasettet ser da slik ut (viser kun første 10 søkere):

```
   Ansatt Utdanning Erfaring Kjønn
1       0         6        2     0
2       0         4        0     1
3       1         6        6     1
4       1         6        3     1
5       0         4        1     0
6       1         8        3     0
7       0         4        2     1
8       0         4        4     0
9       0         6        1     0
10      1         8       10     0
…
```

Vi tilpasser så en regresjonsmodell (`mod.3var`) til datasettet for å undersøke om variablene *Utdanning (antall år)*, *Erfaring (antall år)*, og *Kjønn (1=mann, 0=kvinne)* har effekt på sannsynligheten for å bli ansatt.

```
> summary(mod.3var)

Call:
glm(formula = Ansatt ~ Utdanning + Erfaring + Kjønn, family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max  
-1.4380  -0.4573  -0.1009   0.1294   2.1804  

Coefficients:
            Estimate Std. Error z value Pr(>|z|)  
(Intercept) -14.2483     6.0805  -2.343   0.0191 *
Utdanning     1.1549     0.6023   1.917   0.0552 .
Erfaring      0.9098     0.4293   2.119   0.0341 *
Kjønn         5.6037     2.6028   2.153   0.0313 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 35.165  on 27  degrees of freedom
Residual deviance: 14.735  on 24  degrees of freedom
AIC: 22.735

Number of Fisher Scoring iterations: 7
```

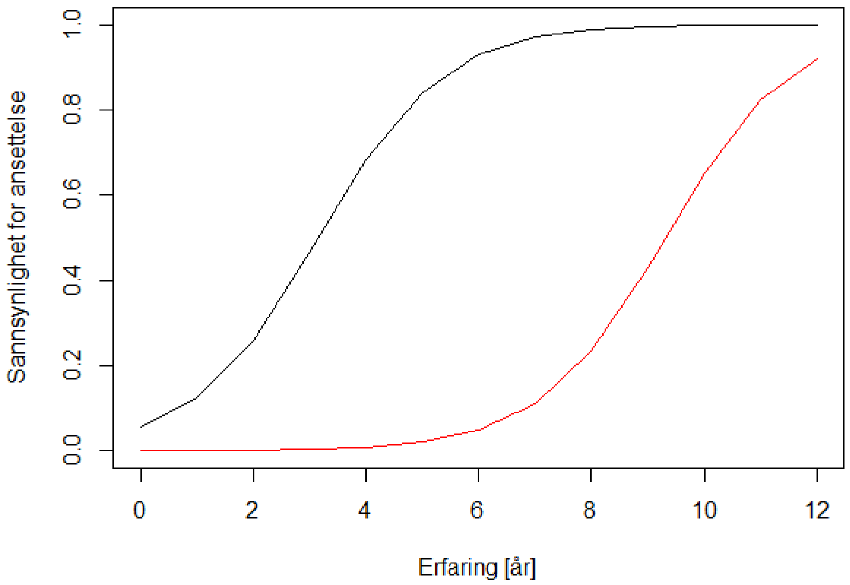
a) Diskuter om forutsetningene for modellen er oppfylt i dette tilfellet.

b) Er det her tilstrekkelig bevis for å påstå at kjønn har betydning for sannsynligheten for å bli ansatt i dette firmaet?

Er de estimerte parameterne i modellen som forventet? Begrunn svaret ved f.eks. å se på fortegnet til estimatet.

c) Sett opp likningen for å regne ut sannsynligheten for at en person skal bli ansatt.

Regn ut sannsynligheten for at ei kvinne med fem års utdanning og to års erfaring skal bli ansatt. Og hva blir tilsvarende sannsynlighet for en mann (dvs. med fem års utdanning og to års erfaring)?



Figur 1. Sannsynlighet for ansettelse for kvinner (rød kurve) og menn (svart kurve) som funksjon av antall års erfaring i arbeidslivet. Antall år med utdanning er her satt til fem år for begge kjønn.

Hvis vi ikke spesifiserer noe annet velges `logit`-funksjonen som link-funksjon for denne typen generalisert lineær modell. Alternative link-funksjoner for en binomisk fordelt responsvariabel er `probit` og `cloglog`. R-kode for å tilpasse modeller med forskjellige link-funksjoner og forskjellig antall forklaringsvariable er vist under. I tillegg presenteres de tilhørende AIC (Akaike's «An Information Criterion») verdiene.

```
mod.0<-glm(Ansatt~1,family=binomial(link="logit"))
mod.3v.logit<-glm(Ansatt~Utdanning+Erfaring+Kjønn,family=binomial(link="logit"))
mod.3v.probit<-glm(Ansatt~Utdanning+Erfaring+Kjønn,family=binomial(link="probit"))
mod.3v.cll<-glm(Ansatt~Utdanning+Erfaring+Kjønn,family=binomial(link="cloglog"))
```

```
> AIC(mod.0,mod.3v.logit,mod.3v.probit,mod.3v.cll)
              df      AIC
mod.0          1 37.16473
mod.3v.logit   4 22.73484
mod.3v.probit  4 22.57725
mod.3v.cll     4 22.29216
```

d) Forklar kort hvorfor vi trenger å spesifisere en link-funksjon for en generalisert lineær modell når responsvariabelen er binomisk fordelt.

Hva måles ved en modells AIC verdi?

Hvilken av modellene over ville du valgt? Svaret skal begrunnes.

Videre fortsetter vi å bruke `logit` som link-funksjon i modellen. I et forsøk på å forenkle modellen `mod.3var` (presentert i summary'et på foregående side) prøver vi å tilpasse en modell hvor *Utdanning* fjernes som forklaringsvariabel for sannsynligheten for å bli ansatt. Denne modellen (`mod.2var`) er presentert under

```
> summary(mod.2var)

Call:
glm(formula = Ansatt ~ Erfaring + Kjønn, family = binomial(link = "logit"),
    data = ansatt)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.2702  -0.6106  -0.2371   0.4715   2.7134

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -5.5914     2.1248  -2.632   0.0085 **
Erfaring      0.6452     0.2562   2.518   0.0118 *
Kjønn         3.8303     1.7829   2.148   0.0317 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 35.165  on 27  degrees of freedom
Residual deviance: 20.847  on 25  degrees of freedom
AIC: 26.847

Number of Fisher Scoring iterations: 6
```

e) Hvilken av disse modellene (`mod.3var` og `mod.2var`) foretrekker du? Begrunn svaret.

   Skriv et R-uttrykk som sammenligner disse to modellene.

f) Sammenhengen mellom utdanning eller erfaring og sannsynligheten for ansettelse kan ha flere former, avhengig av arbeidsgivers preferanser og arbeidsoppgaver. Det kan f.eks. være slik at noen arbeidsgivere vurderer at lengre relevant erfaring kan kompensere for kortere utdanning, eller at en viss lengde på utdanningen er «tilstrekkelig» for arbeidsoppgavene. Forklar kort hvordan du kan undersøke om vi har slike sammenhenger i vår situasjon.

   Kom med noen forslag til andre mulige forbedringer av modellen. Er det f.eks. noen modellforutsetninger som bør sjekkes?

The Poisson Distribution

Description

Density, distribution function, quantile function and random generation for the Poisson distribution with parameter `lambda`.

Usage

```
dpois(x, lambda, log = FALSE)
ppois(q, lambda, lower.tail = TRUE, log.p = FALSE)
qpois(p, lambda, lower.tail = TRUE, log.p = FALSE)
rpois(n, lambda)
```

Arguments

`x`	vector of (non-negative integer) quantiles.
`q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of random values to return.
`lambda`	vector of (non-negative) means.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are $P[X \le x]$, otherwise, $P[X > x]$.

Details

The Poisson distribution has density

$$p(x) = \lambda^x \exp(-\lambda)/x!$$

for $x = 0, 1, 2, \ldots$. The mean and variance are $E(X) = Var(X) = \lambda$.
If an element of `x` is not integer, the result of `dpois` is zero, with a warning. $p(x)$ is computed using Loader's algorithm, see the reference in `dbinom`.
The quantile is right continuous: `qpois(p, lambda)` is the smallest integer x such that $P(X \le x) \ge p$.
Setting `lower.tail = FALSE` allows to get much more precise results when the default, `lower.tail = TRUE` would return 1, see the example below.

Value

`dpois` gives the (log) density, `ppois` gives the (log) distribution function, `qpois` gives the quantile function, and `rpois` generates random deviates.

Invalid `lambda` will result in return value `NaN`, with a warning.
The length of the result is determined by `n` for `rpois`, and is the maximum of the lengths of the numerical arguments for the other functions.
The numerical arguments other than `n` are recycled to the length of the result. Only the first elements of the logical arguments are used.

Source

`dpois` uses C code contributed by Catherine Loader (see `dbinom`).

`ppois` uses `pgamma`.
`qpois` uses the Cornish–Fisher Expansion to include a skewness correction to a normal approximation, followed by a search.
`rpois` uses
Ahrens, J. H. and Dieter, U. (1982). Computer generation of Poisson deviates from modified normal distributions. *ACM Transactions on Mathematical Software*, **8**, 163–179.

See Also

Distributions for other standard distributions, including `dbinom` for the binomial and `dnbinom` for the negative binomial distribution.
`poisson.test`.

Examples

```
require(graphics)

-log(dpois(0:7, lambda = 1) * gamma(1+ 0:7)) # == 1
Ni <- rpois(50, lambda = 4); table(factor(Ni, 0:max(Ni)))

1 - ppois(10*(15:25), lambda = 100)  # becomes 0
(cancellation)
    ppois(10*(15:25), lambda = 100, lower.tail = FALSE)  #
no cancellation

par(mfrow = c(2, 1))
x <- seq(-0.01, 5, 0.01)
plot(x, ppois(x, 1), type = "s", ylab = "F(x)", main =
"Poisson(1) CDF")
plot(x, pbinom(x, 100, 0.01), type = "s", ylab = "F(x)",
     main = "Binomial(100, 0.01) CDF")
```

The Exponential Distribution

Description

Density, distribution function, quantile function and random generation for the exponential distribution with rate `rate` (i.e., mean `1/rate`).

Usage

```
dexp(x, rate = 1, log = FALSE)
pexp(q, rate = 1, lower.tail = TRUE, log.p = FALSE)
qexp(p, rate = 1, lower.tail = TRUE, log.p = FALSE)
rexp(n, rate = 1)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`rate`	vector of rates.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are $P[X \le x]$, otherwise, $P[X > x]$.

Details

If `rate` is not specified, it assumes the default value of `1`.

The exponential distribution with rate λ has density

$f(x) = \lambda {e}^{- \lambda x}$

for $x \ge 0$.

Value

`dexp` gives the density, `pexp` gives the distribution function, `qexp` gives the quantile function, and `rexp`generates random deviates.

The length of the result is determined by `n` for `rexp`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than `n` are recycled to the length of the result. Only the first elements of the logical arguments are used.

Note

The cumulative hazard $H(t) = - \log(1 - F(t))$ is `-pexp(t, r, lower = FALSE, log = TRUE)`.

Source

`dexp`, `pexp` and `qexp` are all calculated from numerically stable versions of the definitions.

`rexp` uses

Ahrens, J. H. and Dieter, U. (1972). Computer methods for sampling from the exponential and normal distributions. *Communications of the ACM*, **15**, 873–882.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Johnson, N. L., Kotz, S. and Balakrishnan, N. (1995) *Continuous Univariate Distributions*, volume 1, chapter 19. Wiley, New York.

See Also

exp for the exponential function.

Distributions for other standard distributions, including dgamma for the gamma distribution anddweibull for the Weibull distribution, both of which generalize the exponential.

Examples

```
dexp(1) - exp(-1) #-> 0

## a fast way to generate *sorted*  U[0,1]  random
numbers:
rsunif <- function(n) { n1 <- n+1
   cE <- cumsum(rexp(n1)); cE[seq_len(n)]/cE[n1] }
plot(rsunif(1000), ylim=0:1, pch=".")
abline(0,1/(1000+1), col=adjustcolor(1, 0.5))
```

Fitting Generalized Linear Models

Description

`glm` is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution.

Usage

```
glm(formula, family = gaussian, data, weights, subset,
    na.action, start = NULL, etastart, mustart, offset,
    control = list(...), model = TRUE, method = "glm.fit",
    x = FALSE, y = TRUE, contrasts = NULL, ...)

glm.fit(x, y, weights = rep(1, nobs),
        start = NULL, etastart = NULL, mustart = NULL,
        offset = rep(0, nobs), family = gaussian(),
        control = list(), intercept = TRUE)

## S3 method for class 'glm'
weights(object, type = c("prior", "working"), ...)
```

Arguments

`formula`	an object of class "`formula`" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
`family`	a description of the error distribution and link function to be used in the model. For `glm` this can be a character string naming a family function, a family function or the result of a call to a family function. For `glm.fit`only the third option is supported. (See `family` for details of family functions.)
`data`	an optional data frame, list or environment (or object coercible by`as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from`environment(formula)`, typically the environment from which `glm` is called.
`weights`	an optional vector of 'prior weights' to be used in the fitting process. Should be `NULL` or a numeric vector.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when the data contain`NA`s. The default is set by the `na.action` setting of `options`, and is`na.fail` if that is unset. The 'factory-fresh' default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful.
`start`	starting values for the parameters in the linear predictor.
`etastart`	starting values for the linear predictor.
`mustart`	starting values for the vector of means.
`offset`	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases. One or more `offset`terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See `model.offset`.
`control`	a list of parameters for controlling the fitting process. For `glm.fit` this is passed to `glm.control`.
`model`	a logical value indicating whether *model frame* should be included as a component of the returned value.
`method`	the method to be used in fitting the model. The default method`"glm.fit"` uses iteratively reweighted least squares (IWLS): the alternative `"model.frame"` returns the model frame and does no fitting. User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as `glm.fit`. If specified as a character string it is looked up from within the **stats** namespace.
`x, y`	For `glm`: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For `glm.fit`: `x` is a design matrix of dimension `n * p`, and `y` is a vector of observations of length `n`.
`contrasts`	an optional list. See the `contrasts.arg` of `model.matrix.default`.
`intercept`	logical. Should an intercept be included in the *null* model?
`object`	an object inheriting from class `"glm"`.
`type`	character, partial matching allowed. Type of weights to extract from the fitted model object. Can be abbreviated.
`...`	For `glm`: arguments to be used to form the default `control` argument if it is not supplied directly. For `weights`: further arguments passed to or from other methods.

Details

A typical predictor has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for`response`. For `binomial` and `quasibinomial` families the response can also be specified as a `factor` (when the first level denotes failure and all others success) or as a two-column matrix with the columns giving the numbers of successes and failures. A terms specification of the form `first + second` indicates all the terms in `first`together with all the terms in `second` with any duplicates removed.

A specification of the form `first:second` indicates the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification`first*second` indicates the *cross* of `first` and `second`. This is the same as `first + second + first:second`.

The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a `terms`object as the formula.

Non-`NULL` `weights` can be used to indicate that different observations have different dispersions (with the values in `weights` being inversely proportional to the dispersions); or equivalently, when the elements of `weights` are positive integers *w_i*, that each response *y_i* is the mean of *w_i* unit-weight observations. For a binomial GLM prior weights are used to give the number of trials when the response is the proportion of successes: they would rarely be used for a Poisson GLM.

`glm.fit` is the workhorse function: it is not normally called directly but can be more efficient where the response vector, design matrix and family have already been calculated.

If more than one of `etastart`, `start` and `mustart` is specified, the first in the list will be used. It is often advisable to supply starting values for a `quasi` family, and also for families with unusual links such as `gaussian("log")`.

All of `weights`, `subset`, `offset`, `etastart` and `mustart` are evaluated in the same way as variables in `formula`, that is first in `data` and then in the environment of`formula`.

For the background to warning messages about 'fitted probabilities numerically 0 or 1 occurred' for binomial GLMs, see Venables & Ripley (2002, pp. 197–8).

Value

`glm` returns an object of class inheriting from `"glm"` which inherits from the class `"lm"`. See later in this section. If a non-standard `method` is used, the object will also inherit from the class (if any) returned by that function.

The function `summary` (i.e., `summary.glm`) can be used to obtain or print a summary of the results and the function `anova` (i.e., `anova.glm`) to produce an analysis of variance table.

The generic accessor functions `coefficients`, `effects`, `fitted.values` and`residuals` can be used to extract various useful features of the value returned by `glm`.

`weights` extracts a vector of weights, one for each case in the fit (after subsetting and`na.action`).

An object of class `"glm"` is a list containing at least the following components:

`coefficients`	a named vector of coefficients
`residuals`	the *working* residuals, that is the residuals in the final iteration of the IWLS fit. Since cases with zero weights are omitted, their working residuals are `NA`.
`fitted.values`	the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
`rank`	the numeric rank of the fitted linear model.
`family`	the `family` object used.
`linear.predictors`	the linear fit on link scale.
`deviance`	up to a constant, minus twice the maximized log-likelihood. Where sensible, the constant is chosen so that a saturated model has deviance zero.
`aic`	A version of Akaike's *An Information Criterion*, minus twice the maximized log-likelihood plus twice the number of parameters, computed by the `aic` component of the family. For binomial and Poison families the dispersion is fixed at one and the number of parameters is the number of coefficients. For gaussian, Gamma and inverse gaussian families

	the dispersion is estimated from the residual deviance, and the number of parameters is the number of coefficients plus one. For a gaussian family the MLE of the dispersion is used so this is a valid value of AIC, but for Gamma and inverse gaussian families it is not. For families fitted by quasi-likelihood the value is `NA`.
`null.deviance`	The deviance for the null model, comparable with `deviance`. The null model will include the offset, and an intercept if there is one in the model. Note that this will be incorrect if the link function depends on the data other than through the fitted mean: specify a zero offset to force a correct calculation.
`iter`	the number of iterations of IWLS used.
`weights`	the *working* weights, that is the weights in the final iteration of the IWLS fit.
`prior.weights`	the weights initially supplied, a vector of `1`s if none were.
`df.residual`	the residual degrees of freedom.
`df.null`	the residual degrees of freedom for the null model.
`y`	if requested (the default) the `y` vector used. (It is a vector even for a binomial model.)
`x`	if requested, the model matrix.
`model`	if requested (the default), the model frame.
`converged`	logical. Was the IWLS algorithm judged to have converged?
`boundary`	logical. Is the fitted value on the boundary of the attainable values?
`call`	the matched call.
`formula`	the formula supplied.
`terms`	the `terms` object used.
`data`	the `data argument`.
`offset`	the offset vector used.
`control`	the value of the `control` argument used.
`method`	the name of the fitter function used, currently always`"glm.fit"`.
`contrasts`	(where relevant) the contrasts used.
`xlevels`	(where relevant) a record of the levels of the factors used in fitting.
`na.action`	(where relevant) information returned by `model.frame` on the special handling of `NA`s.

In addition, non-empty fits will have components `qr`, `R` and `effects` relating to the final weighted linear fit.

Objects of class `"glm"` are normally of class `c("glm", "lm")`, that is inherit from class `"lm"`, and well-designed methods for class `"lm"` will be applied to the weighted linear model at the final iteration of IWLS. However, care is needed, as extractor functions for class `"glm"` such as `residuals` and `weights` do **not** just pick out the component of the fit with the same name.

If a `binomial` `glm` model was specified by giving a two-column response, the weights returned by `prior.weights` are the total numbers of cases (factored by the supplied case weights) and the component `y` of the result is the proportion of successes.

Fitting functions

The argument `method` serves two purposes. One is to allow the model frame to be recreated with no fitting. The other is to allow the default fitting function `glm.fit` to be replaced by a function which takes the same arguments and uses a different fitting algorithm. If `glm.fit` is supplied as a character string it is used to search for a function of that name, starting in the **stats** namespace.

The class of the object return by the fitter (if any) will be prepended to the class returned by `glm`.

See Also

`anova.glm`, `summary.glm`, etc. for `glm` methods, and the generic functions `anova`,`summary`, `effects`, `fitted.values`, and `residuals`.

`lm` for non-generalized *linear* models (which SAS calls GLMs, for 'general' linear models).

`loglin` and `loglm` (package **MASS**) for fitting log-linear models (which binomial and Poisson GLMs are) to contingency tables.

`bigglm` in package **biglm** for an alternative way to fit GLMs to large datasets (especially those with many cases).

`esoph`, `infert` and `predict.glm` have examples of fitting binomial glms.

Examples

```
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))
glm.D93 <- glm(counts ~ outcome + treatment, family =
poisson())
anova(glm.D93)
summary(glm.D93)

## an example with offsets from Venables & Ripley (2002,
p.189)
utils::data(anorexia, package = "MASS")

anorex.1 <- glm(Postwt ~ Prewt + Treat + offset(Prewt),
                family = gaussian, data = anorexia)
summary(anorex.1)


# A Gamma example, from McCullagh & Nelder (1989, pp. 300-
2)
clotting <- data.frame(
    u = c(5,10,15,20,30,40,60,80,100),
    lot1 = c(118,58,42,35,27,25,21,19,18),
    lot2 = c(69,35,26,21,18,16,13,12,12))
summary(glm(lot1 ~ log(u), data = clotting, family =
Gamma))
summary(glm(lot2 ~ log(u), data = clotting, family =
Gamma))

## Not run:
## for an example of the use of a terms object as a
formula
demo(glm.vr)
```

Akaike's An Information Criterion

Description

Generic function calculating Akaike's 'An Information Criterion' for one or several fitted model objects for which a log-likelihood value can be obtained, according to the formula *-2*log-likelihood + k*npar*, where*npar* represents the number of parameters in the fitted model, and *k = 2* for the usual AIC, or *k = log(n)*(*n* being the number of observations) for the so-called BIC or SBC (Schwarz's Bayesian criterion).

Usage

```
AIC(object, ..., k = 2)

BIC(object, ...)
```

Arguments

`object`	a fitted model object for which there exists a logLik method to extract the corresponding log-likelihood, or an object inheriting from class logLik.
`...`	optionally more fitted model objects.
`k`	numeric, the *penalty* per parameter to be used; the default k = 2 is the classical AIC.

Details

When comparing models fitted by maximum likelihood to the same data, the smaller the AIC or BIC, the better the fit.

The theory of AIC requires that the log-likelihood has been maximized: whereas AIC can be computed for models not fitted by maximum likelihood, their AIC values should not be compared.

Examples of models not ‘fitted to the same data’ are where the response is transformed (accelerated-life models are fitted to log-times) and where contingency tables have been used to summarize data.

These are generic functions (with S4 generics defined in package **stats4**): however methods should be defined for the log-likelihood function logLik rather than these functions: the action of their default methods is to call logLik on all the supplied objects and assemble the results. Note that in several common cases logLik does not return the value at the MLE: see its help page.

The log-likelihood and hence the AIC/BIC is only defined up to an additive constant. Different constants have conventionally been used for different purposes and so extractAIC and AIC may give different values (and do for models of class "lm": see the help for extractAIC). Particular care is needed when comparing fits of different classes (with, for example, a comparison of a Poisson and gamma GLM being meaningless since one has a discrete response, the other continuous).

BIC is defined as AIC(object, ..., k = log(nobs(object))). This needs the number of observations to be known: the default method looks first for a "nobs" attribute on the return value from the logLik method, then tries the nobs generic, and if neither succeed returns BIC as NA.

Value

If just one object is provided, a numeric value with the corresponding AIC (or BIC, or ..., depending onk).

If multiple objects are provided, a data.frame with rows corresponding to the objects and columns representing the number of parameters in the model (df) and the AIC or BIC.

Author(s)

Originally by José Pinheiro and Douglas Bates, more recent revisions by R-core.

References

Sakamoto, Y., Ishiguro, M., and Kitagawa G. (1986). *Akaike Information Criterion Statistics*. D. Reidel Publishing Company.

See Also

extractAIC, logLik, nobs.

Examples

```
lm1 <- lm(Fertility ~ . , data = swiss)
AIC(lm1)
stopifnot(all.equal(AIC(lm1),
                    AIC(logLik(lm1))))
BIC(lm1)

lm2 <- update(lm1, . ~ . -Examination)
AIC(lm1, lm2)
BIC(lm1, lm2)
```