NTNU

Kunnskap for en bedre verden

Institutt for matematiske fag

Eksamensoppgave i
ST2304 Statistisk modellering for biologer og bioteknologer

Faglig kontakt under eksamen: Jarle Tufto

Tlf: 99 70 55 19

Eksamensdato: 28. mai, 2018

Eksamenstid (fra–til): 9–13

Hjelpemiddelkode/Tillatte hjelpemidler: Tabeller og formler i statistikk, Tapir Forlag, K. Rottmann: Matematisk formelsamling, spesifisert kalkulator, ett gult A4-ark med stempel og egne håndskrevne notater.

Annen informasjon:
Hjelpesider for noen R funksjoner som du kan få bruk for følger i vedlegget. Alle svar skal begrunnes og besvarelsen skal inneholde naturlig mellomregning.

Målform/språk: bokmål

Antall sider: 6

Antall sider vedlegg: 3

Kontrollert av:

Informasjon om trykking av eksamensoppgave	
Originalen er:	
1-sidig ☐	**2-sidig** ☒
sort/hvit ☒	**farger** ☐
skal ha flervalgskjema ☐	

Dato Sign

Oppgave 1

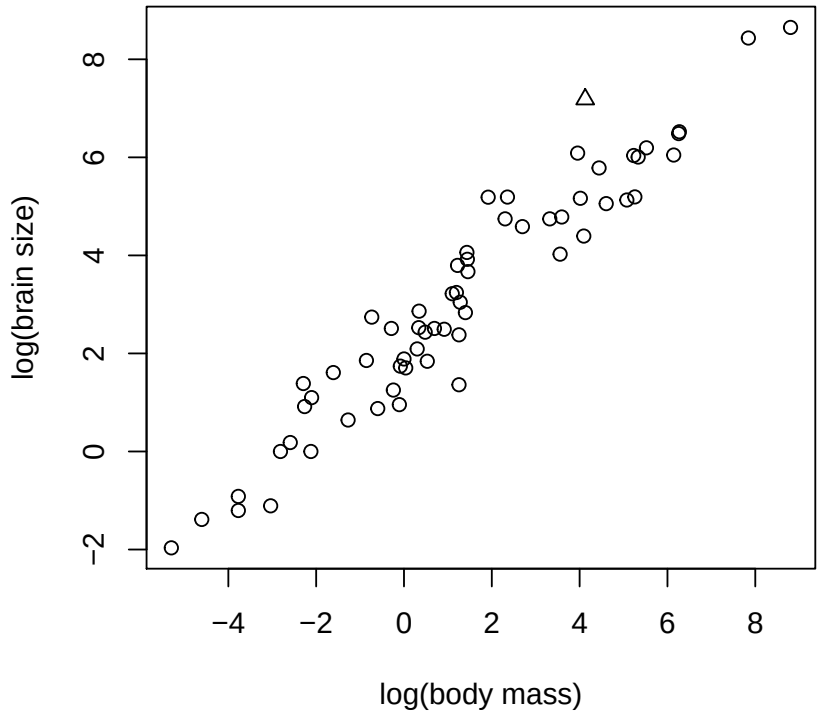
Vi ønsker å undersøke sammenhengen mellom kroppsmasse x (målt i kg) og hjernestørrelse y (målt i gram) hos ulike patterdyrarter. Vi log transformerer først begge variabler (ved å bruke naturlig logaritme) (se Figur 1) og tilpasser så en lineær regresjonsmodel til dataene (med arten menneske ekskludert) på følgende måte.

```
> logx <- log(x)  # log gives natural log in R
> logy <- log(y)
> model0 <- lm(logy ~ logx)
> summary(model0)

Call:
lm(formula = logy ~ logx)

Residuals:
     Min       1Q   Median       3Q      Max
-1.68392 -0.47707 -0.02668  0.47305  1.64949

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.11500    0.09030   23.42   <2e-16 ***
logx         0.74228    0.02687   27.62   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6511 on 59 degrees of freedom
Multiple R-squared:  0.9282,Adjusted R-squared:  0.927
F-statistic:   763 on 1 and 59 DF,  p-value: < 2.2e-16
```

a) Skriv opp modellen vi har tilpasset i matematisk notasjon. Hvilke antakelser gjør vi når vi bruker modellen? Hvilke ukjente parametere inneholder modellen? Hva er estimatene av disse parametrene?

b) Anta at vi ønsker å undersøke om hjernestørrelse til ulike patterdyrarter avhenger av deres kroppsmasse. Hva er da nullhypotese og alternativ hypotese, gitt modellen over? Gitt et signifikansnivå $\alpha = 0,05$ og output fra R over, hva er testens konklusjon?

c) Den ene patterdyrarten i datasettet, menneske, har kroppsmasse $x = 62$ kg (i gjennomsnitt) (triangelet i Figur 1). Naturlig logaritme av kroppsmasse for menneske er dermed $\ln(62) = 4.127$. Basert på modellen vi har tilpasset, hva blir forventet log hjernestørrelse til menneske? Hvor mange gram svarer dette til? Diskuter kort forskjellen mellom predikert verdi og faktisk hjernestørrelse lik 1320 gram for menneske.



Figur 1: Hjerne og kroppsstørrelse til ulike pattedyrarter etter log-transformasjon. Triangelet representerer arten mennneske.

Vi ønsker å teste om sammenhengen mellom $\ln y$ og $\ln x$ er ikke-lineær og gjør dette ved å inkludere kvadratet av log kroppsmasse, $(\ln x)^2$, som en ekstra forklaringsvariabel i modellen som følger:

```
> logx2 <- logx^2
> model1 <- lm(logy ~ logx + logx2)
> summary(model1)

Call:
lm(formula = logy ~ logx + logx2)

Residuals:
     Min       1Q   Median       3Q      Max
-1.74543 -0.48045  0.01852  0.41621  1.58807

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.161272   0.103471   20.89   <2e-16 ***
logx         0.762467   0.034724   21.96   <2e-16 ***
logx2       -0.006407   0.006965   -0.92    0.361
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.652 on 58 degrees of freedom
Multiple R-squared:  0.9293,Adjusted R-squared:  0.9268
F-statistic: 380.9 on 2 and 58 DF,  p-value: < 2.2e-16
```

d) Gir denne modelltilpasningen oss noen grunn til å tro at sammenhengen mellom $\ln y$ og $\ln x$ er ikke-linear?

e) Vis at den lineære sammenhengen mellom $\ln y$ og $\ln x$ som vi har antatt i punkt a) medfører at sammenhengen mellom y og x blir på formen

$$y = cx^b.$$

Hva er estimatet av parameteren b? Hva ville b vært dersom hjernestørrelse var direkte proposjonal med kroppsmasse? Utfør en tosidig test av hypotesen at hjernestørrelse er direkte proposjonal med kroppsmasse ved å bruke det som er gitt i begynnelsen av oppgaven. Bruk signifikansnivå $\alpha = 0.05$.

Hint:

$$\ln(a \cdot b) = \ln a + \ln b, \qquad \ln(a^b) = b \ln a, \qquad e^{a+b} = e^a \cdot e^b, \qquad e^{a \ln b} = b^a.$$

Oppgave 2

På grunn av mistanke om forhøyet forekomst av lungekreft som følge av petrokjemisk industrivirksomhet i den danske byen Fredricia ble følgende datasett på forekomst av lungekreft i 4 ulike danske byer og i 6 ulike aldergrupper samlet inn på slutten av 1970-tallet.

```
> eba1977
         city   age    n  y petro
1  Fredericia 40-54 3059 11   yes
2     Horsens 40-54 2879 13    no
3     Kolding 40-54 3142  4    no
4       Vejle 40-54 2520  5    no
5  Fredericia 55-59  800 11   yes
6     Horsens 55-59 1083  6    no
7     Kolding 55-59 1050  8    no
8       Vejle 55-59  878  7    no
9  Fredericia 60-64  710 11   yes
10    Horsens 60-64  923 15    no
11    Kolding 60-64  895  7    no
12      Vejle 60-64  839 10    no
13 Fredericia 65-69  581 10   yes
14    Horsens 65-69  834 10    no
15    Kolding 65-69  702 11    no
16      Vejle 65-69  631 14    no
17 Fredericia 70-74  509 11   yes
18    Horsens 70-74  634 12    no
19    Kolding 70-74  535  9    no
20      Vejle 70-74  539  8    no
21 Fredericia   75+  605 10   yes
22    Horsens   75+  782  2    no
23    Kolding   75+  659 12    no
24      Vejle   75+  619  7    no
> levels(city)
[1] "Horsens"    "Kolding"    "Vejle"      "Fredericia"
> levels(age)
[1] "40-54" "55-59" "60-64" "65-69" "70-74" "75+"
```

Merk at første nivå av `age` og `city` er henholdsvis 40-54 år og Horsens. Variabelen `petro` diskuteres først i punkt e). Variablene `y` og `n` representerer henholdsvis antall tilfeller av lungekreft ut av totalt antall personer innenfor hver by og alderskategori. Vi tilpasser følgende generalisert lineære modell (`glm2`).

```
> glm2 <- glm(cbind(y, n - y) ~ city + age, binomial(link = "logit"))
> summary(glm2)

Call:
```

```
glm(formula = cbind(y, n - y) ~ city + age, family = binomial(link = "logit"))

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.64532  -0.67472  -0.03449   0.37480   1.85912

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    -5.96072    0.21122 -28.220  < 2e-16 ***
cityKolding    -0.04189    0.19313  -0.217   0.8283
cityVejle       0.05843    0.19315   0.303   0.7622
cityFredericia  0.33447    0.18273   1.830   0.0672 .
age55-59        1.10699    0.24902   4.445 8.77e-06 ***
age60-64        1.52908    0.23250   6.577 4.81e-11 ***
age65-69        1.78192    0.23047   7.732 1.06e-14 ***
age70-74        1.87272    0.23652   7.918 2.42e-15 ***
age75+          1.42888    0.25123   5.688 1.29e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 130.999  on 23  degrees of freedom
Residual deviance:  23.638  on 15  degrees of freedom
AIC: 137.74

Number of Fisher Scoring iterations: 5
```

a) La $i = 1, 2, \ldots, 4$ og $j = 1, 2, \ldots, 6$ representere de to kategoriske forklaringsvariablene `city` og `age`. For hver observasjon antar modellen vi har tilpasset da at responsvariabelen $Y \sim \text{bin}(n, p)$, og at

$$\text{logit}\, p = \mu + \alpha_i + \beta_j.$$

Hvilke ukjente parametere inneholder modellen og hva representerer disse parametrene? Hvorfor kan det være en rimelig antakelse at antall forekomster av lungekreft er binomisk fordelt? Hvorfor trenger vi generellt å bruke en link funksjon når vi modellerer sannsynligheten for en hendelse?

b) Gitt modellen vi har tilpasset over, beregn sannsynligheten p for lungekreft i Horsens innenfor alderskategori 40-54 år. Beregn tilsvarende sannsynlighet i samme alderskategori men i byen Fredericia.

c) Hvor mye større er oddsen $p/(1-p)$ for lungekreft i Fredericia sammenlignet med Horsens innenfor en gitt alderskategori? Uttrykk gjerne svaret i prosent.

d) Utfør en test av om det er overdispersjon i dataene. Bruk $\alpha = 0,05$ som signifikansnivå. Diskuter kort mulige mekanismer som kan tenkes å generere overdispersjon i dette konkrete datasettet.

e) For å teste om det er en signifikant forskjell mellom byene tilpasser vi en enklere modell `glm0` og sammenligner denne med `glm2` som følger.

```
> glm0 <- glm(cbind(y, n - y) ~ age, binomial(link = "logit"))
> anova(glm0, glm2, test = "Chisq")
Analysis of Deviance Table

Model 1: cbind(y, n - y) ~ age
Model 2: cbind(y, n - y) ~ city + age
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        18     28.559
2        15     23.638  3   4.9215   0.1776
```

Hvilken nullhypotese og alternative hypotese testes her? Hva er testens konklusjon?

Som nok en alternativ modell `glm1` tar vi `city` ut av modellen men inkluderer i stedet forklaringsvariabelen `petro` som har to nivå (`yes` for Fredericia og `no` for andre byer). Vi sammenligner så denne modellen med `glm0` og `glm2` som følger.

```
> glm1 <- glm(cbind(y, n - y) ~ petro + age, binomial(link = "logit"))
> anova(glm0, glm1, test = "Chisq")
Analysis of Deviance Table

Model 1: cbind(y, n - y) ~ age
Model 2: cbind(y, n - y) ~ petro + age
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        18     28.559
2        17     23.893  1   4.6662  0.03076 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> anova(glm1, glm2, test = "Chisq")
Analysis of Deviance Table

Model 1: cbind(y, n - y) ~ petro + age
Model 2: cbind(y, n - y) ~ city + age
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        17     23.893
2        15     23.638  2  0.25532   0.8802
```

På bakgrunn av spørsmålet om det er en forhøyet risiko for lungekreft i Fredericia knyttet til den petrokjemiske industrien, er `glm1` en rimelig modell? Gir dataene støtte for mistanken om en forhøyet risiko for lungekreft i Fredericia på grunn av den petrokjemiske industrien? Bruk signifikansnivå $\alpha = 0.05$

`lm` *Fitting Linear Models*

Description

`lm` is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although `aov` may provide a more convenient interface for these).

Usage

```
lm(formula, data, subset, weights, na.action,
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
   singular.ok = TRUE, contrasts = NULL, offset, ...)
```

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'.
`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 `lm` is called.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If non-NULL, weighted least squares is used with weights `weights` (that is, minimizing `sum(w*e^2)`); otherwise ordinary least squares is used. See also 'Details',
`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.
`method`	the method to be used; for fitting, currently only `method = "qr"` is supported; `method = "model.frame"` returns the model frame (the same as with `model = TR` see below).
`model, x, y, qr`	logicals. If `TRUE` the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.
`singular.ok`	logical. If `FALSE` (the default in S but not in R) a singular fit is an error.
`contrasts`	an optional list. See the `contrasts.arg` of `model.matrix.default`.
`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 are specified their sum is used. See `model.offset`.
`...`	additional arguments to be passed to the low level regression fitting functions (see below).

Details

Models for `lm` are specified symbolically. A typical model 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`. A terms specification of the form `first + second` indicates all the terms in `first` together with all the terms in `second` with 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`.

If the formula includes an `offset`, this is evaluated and subtracted from the response.

If `response` is a matrix a linear model is fitted separately by least-squares to each column of the matrix.

See `model.matrix` for some further details. 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 (see `aov` and `demo(glm.vr)` for an example).

A formula has an implied intercept term. To remove this use either `y ~ x - 1` or `y ~ 0 + x`. See `formula` for more details of allowed formulae.

Non-NULL `weights` can be used to indicate that different observations have different variances (with the values in `weights` being inversely proportional to the variances); 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 (including the case that there are w_i observations equal to y_i and the data have been summarized).

`lm` calls the lower level functions `lm.fit`, etc, see below, for the actual numerical computations. For programming only, you may consider doing likewise.

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

Value

`lm` returns an object of `class` `"lm"` or for multiple responses of class `c("mlm", "lm")`.

The functions `summary` and `anova` are used to obtain and print a summary and analysis of variance table of the results. The generic accessor functions `coefficients`, `effects`, `fitted.values` and `residuals` extract various useful features of the value returned by `lm`.

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

`coefficients`	a named vector of coefficients
`residuals`	the residuals, that is response minus fitted values.
`fitted.values`	the fitted mean values.
`rank`	the numeric rank of the fitted linear model.
`weights`	(only for weighted fits) the specified weights.
`df.residual`	the residual degrees of freedom.
`call`	the matched call.
`terms`	the `terms` object used.
`contrasts`	(only where relevant) the contrasts used.
`xlevels`	(only where relevant) a record of the levels of the factors used in fitting.
`offset`	the offset used (missing if none were used).
`y`	if requested, the response used.
`x`	if requested, the model matrix used.
`model`	if requested (the default), the model frame used.
`na.action`	(where relevant) information returned by `model.frame` on the special handling of `NA`s.

In addition, non-null fits will have components `assign`, `effects` and (unless not requested) `qr` relating to the linear fit, for use by extractor functions such as `summary` and `effects`.

Using time series

Considerable care is needed when using `lm` with time series.

Unless `na.action = NULL`, the time series attributes are stripped from the variables before the regression is done. (This is necessary as omitting `NA`s would invalidate the time series attributes, and if `NA`s are omitted in the middle of the series the result would no longer be a regular time series.)

Even if the time series attributes are retained, they are not used to line up series, so that the time shift of a lagged or differenced regressor would be ignored. It is good practice to prepare a `data` argument by `ts.intersect(..., dframe = TRUE)`, then apply a suitable `na.action` to that data frame and call `lm` with `na.action = NULL` so that residuals and fitted values are time series.

Note

Offsets specified by `offset` will not be included in predictions by `predict.lm`, whereas those specified by an offset term in the formula will be.

Author(s)

The design was inspired by the S function of the same name described in Chambers (1992). The implementation of model formula by Ross Ihaka was based on Wilkinson & Rogers (1973).

References

Chambers, J. M. (1992) *Linear models.* Chapter 4 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

Wilkinson, G. N. and Rogers, C. E. (1973) Symbolic descriptions of factorial models for analysis of variance. *Applied Statistics*, **22**, 392–9.

See Also

`summary.lm` for summaries and `anova.lm` for the ANOVA table; `aov` for a different interface.

The generic functions `coef`, `effects`, `residuals`, `fitted`, `vcov`.

`predict.lm` (via `predict`) for prediction, including confidence and prediction intervals; `confint` for confidence intervals of *parameters*.

`lm.influence` for regression diagnostics, and `glm` for **generalized** linear models.

The underlying low level functions, `lm.fit` for plain, and `lm.wfit` for weighted regression fitting.

More `lm()` examples are available e.g., in `anscombe`, `attitude`, `freeny`, `LifeCycleSavings`, `longley`, `stackloss`, `swiss`.

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

Examples

```
require(graphics)

## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
lm.D90 <- lm(weight ~ group - 1) # omitting intercept

anova(lm.D9)
summary(lm.D90)

opar <- par(mfrow = c(2,2), oma = c(0, 0, 1.1, 0))
plot(lm.D9, las = 1)      # Residuals, Fitted, ...
par(opar)

### less simple examples in "See Also" above
```

`glm` *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`.

Author(s)

The original R implementation of `glm` was written by Simon Davies working for Ross Ihaka at the University of Auckland, but has since been extensively re-written by members of the R Core team.

The design was inspired by the S function of the same name described in Hastie & Pregibon (1992).

References

Dobson, A. J. (1990) *An Introduction to Generalized Linear Models.* London: Chapman and Hall.

Hastie, T. J. and Pregibon, D. (1992) *Generalized linear models.* Chapter 6 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

McCullagh P. and Nelder, J. A. (1989) *Generalized Linear Models.* London: Chapman and Hall.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* New York: Springer.

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)

## End(Not run)
```