NTNU
Norwegian University of
Science and Technology

Department of Mathematical Sciences

Examination paper for **ST2304 Statistical modelling for biologists and biotechologists SOLUTION**

Academic contact during examination:

Phone:

Examination date: SOLUTION

Examination time (from–to):

Permitted examination support material:

Other information:

Language: English

Number of pages: 10

Number of pages enclosed: 3

Checked by:

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

Date Signature

Problem 1 Beetles

Solutions in blue.

One problem in toxicology is to work out how toxic a compound is to an organism. This is estimated by calculating the LD_{50} ("lethal dose"). Animals are treated with different concentrations of a toxin, and from this the concentration that would kill half of the animals is estimated: this is the LD_{50}. Here we want to estimate the LD_{50} for beetles exposed to carbon disulphide at 8 different concentrations. The data are plotted in Figure 1.

All log values are the natural log.

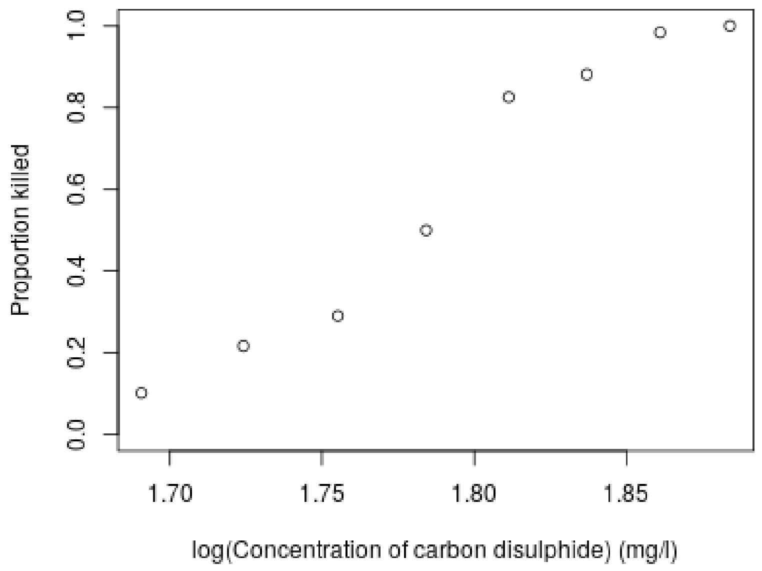

Figure 1: Proportion of beetles killed by exposure to different concentrations of carbon disulphide.

The LD_{50} can be estimated by fitting a GLM, assuming a binomial distribution for each concentration, and modelling how the proportion of beetles that are killed (which we call p) varies with the log(concentration). For the analysis here we assume a logit link function, and a linear effect of log(concentration).

a) What are the parameters of the binomial distribution? N, p (2 marks). But also give marks if they say intercept & slope instead of p.

b) What assumptions do we make when using a GLM with a binomial distribution? p varies with dose: linear on log scale, independence, binomial distribution (2 marks).

c) What other link functions could be used in this analysis? probit, cloglog (2 marks).

The model was fitted, with the summary shown in Figure 2.

d) Write down the model for the effect of log(concentration) on logit(p), in mathematical notation. logit(p) = -59.8 + 33.8 x (3 marks: 1 for a +bx).

e) Does the analysis support the assumption that there is an effect of log(concentration) on the proportion of beetles killed? Explain your answer. Yes. The z-test for Concentration, the deviance test, and the confidence interval all suggest it's horribly significant. (4 marks: one for yes, one for "it's significant" (by whatever measure), 2 for giving statistics).

The LD_{50} can be calulated from the model as $-\frac{intercept}{slope}$.

f) What is the estimate of LD_{50} in this experiment? 59.869/33.784 = 1.77 (1 mark).

Because the estimate is a ratio, the confidence interval for it is not trivial to work out. Therefore replicate data was simulated from the model, and the LD_{50} was estimated from the simulations. This gave the density in Figure 3.

g) Using just the plot, suggest an approximate 95% confidence interval. (Your estimate will be rough, because from the plot you can only give a guess) Exact is (1.764, 1.780), i.e. (1.76, 1.78). Allow some latitude (2 marks)

The residuals from the model in Figure 2 are plotted in Figure 4.

h) Does the residual plot (Figure 4) suggest that the model assumptions seem reasonable? If not, what could you do to improve it?

evidence of curvature (2 point). Could transform Concentration, or use a quadratic (could use a different link function) (2 points).

```
 beetle <- glm(cbind(NKilled, NSurv) ~ log(concentration),
+                family=binomial("logit"))
> summary(beetle)

Call:
glm(formula = cbind(NKilled, NSurv) ~ log(concentration),
    family = binomial("logit"))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5213  -0.6270   0.8705   1.2575   1.6486

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)         -59.869      5.108  -11.72   <2e-16 ***
log(concentration)   33.784      2.870   11.77   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 280.866  on 7  degrees of freedom
Residual deviance:  11.474  on 6  degrees of freedom
AIC: 41.803

Number of Fisher Scoring iterations: 4
```

Figure 2: R code and results from fitting a GLM for the proportion of beetles killed as a response to log(concentration)

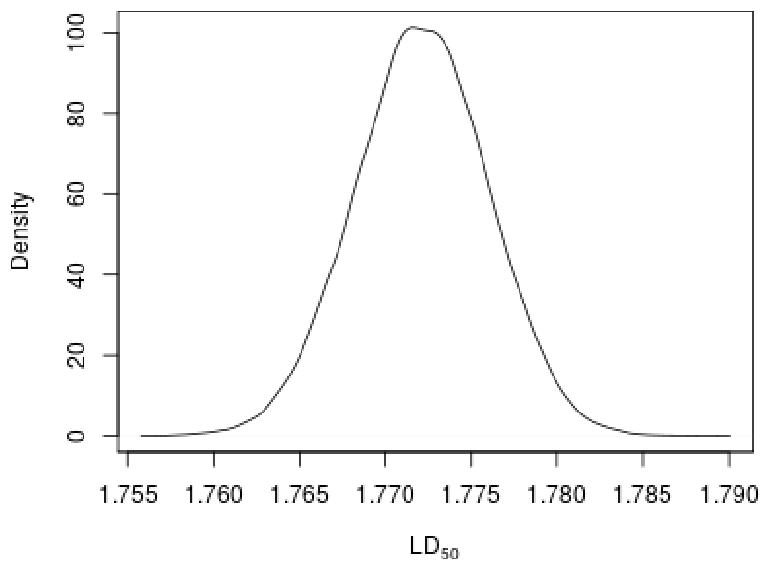

Figure 3: Density curve for LD_{50} for carbon disulphide in beetles.

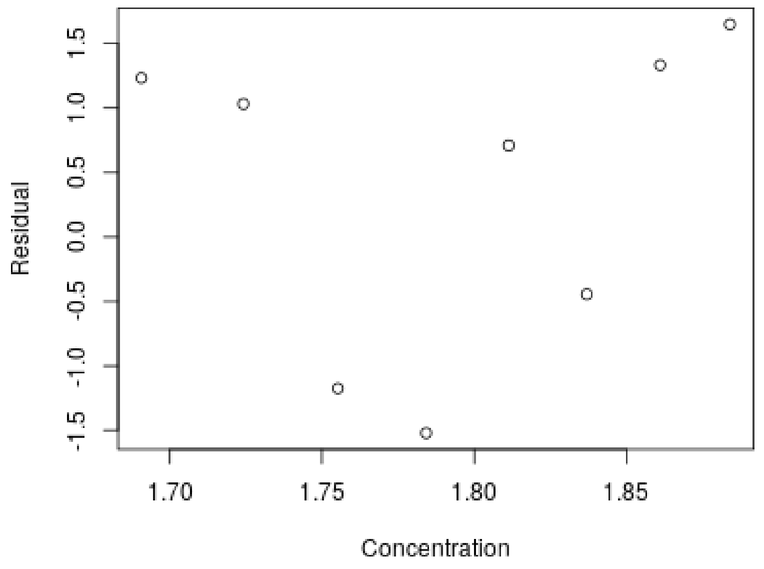

Figure 4: Residuals of the model in Figure 2

Problem 2 Seed Retention Time (SRT)

A new piece of research recently looked at seed retention time (SRT) in different animal species, i.e. the time from when an animal eats a seed to when it excretes the seed in its faeces. Data on this, along with the body size of the species, has been collected. The study asked how long it would take a seed to go through any animal: this can be linked to movement to predict how far an animal can disperse seeds. The study looked at two taxa, birds and mammals. As an example, we can ask how long it would take a seed to go through a *Tyrannosaurus rex* (which we can consider to be a bird). The data are plotted in Figure 5.

All log values are the natural log.

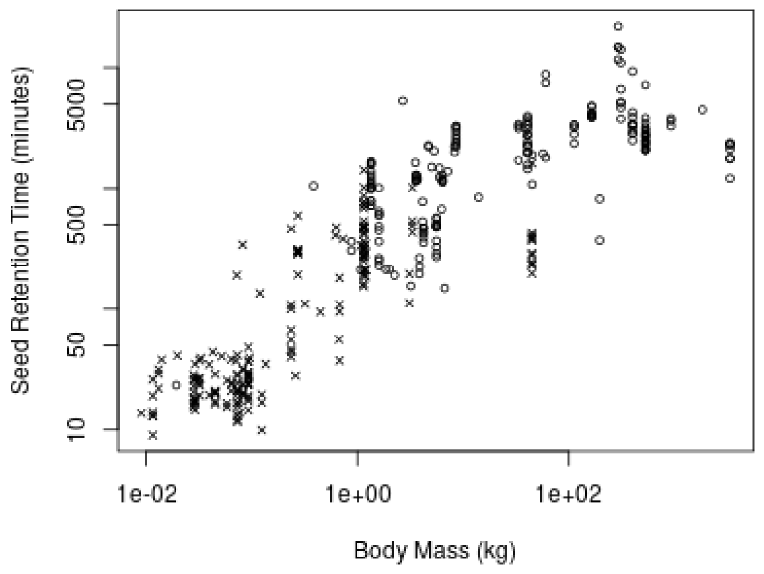

Figure 5: Seed retention time and body mass for mammals and birds, both plotted on the log scale.

We can fit a regression model and use that to make this prediction. The model and its summary are in Figure 6.

a) Write down the model for the effect of log(body mass) on log(SRT) in mathematical notation. Answer below.

$$y_i = alpha + beta * x_i + e(4marks).$$

```
> model1 <- lm(log(SRT) ~ log(Body.mass), data=Data)
> summary(model1)

Call:
lm(formula = log(SRT) ~ log(Body.mass), data = Data)

Residuals:
     Min       1Q   Median       3Q      Max
-2.52500 -0.70189 -0.04724  0.75092  2.72079

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     1.69101    0.12377   13.66   <2e-16 ***
log(Body.mass)  0.52801    0.01454   36.32   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9425 on 384 degrees of freedom
  (4 observations deleted due to missingness)
Multiple R-squared:  0.7745,       Adjusted R-squared:  0.7739
F-statistic:  1319 on 1 and 384 DF,  p-value: < 2.2e-16
```

Figure 6: R code and summary from a linear regression model to predict log(seed retention time (SRT)) by log(body mass) of a species.

b) What assumptions are we making when using a linear regression model? Linear response, gaussian residuals, independence (4 marks), homoscedasticity.

c) Which unknown parameters does the model contain? Intercept, slope, residual variance (3 marks).

d) What is the effect of body mass on SRT? SRT increases with body masss (1 mark), coefficient of 0.53 and R^2 of 0.77 (2 marks if they use one or both of these to say effect is reasonably large). 2 marks if notice that effect is not linear, but decreasing, on original scale.

The body mass of *T. rex* has been estimated as about 7700 kg, and we can assume it behaved like a bird.

e) Predict the SRT for a *T. rex* using the output of model 1 (Figure 6). (log(7700) = 8.95)

log(y.trex) <- 1.69 + 0.53 * 8.95 = 6.43 (1 mark). y.trex = exp(6.43) = 620 minutes, = 10 1/3 hours (1 mark).

The model fit of model 1 was checked graphically: see Figure 7.

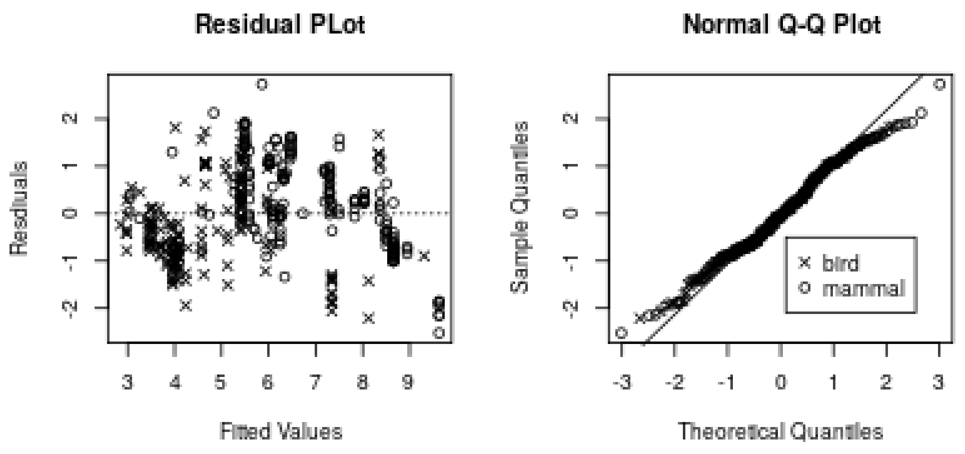

Figure 7: Residuals for a model of the effect of body mass for mammals and birds on seed retention time.

f) How well does model 1 fit? Are there any deviations from the assumptions of the model? QQ plot suggests normal. No signs of outliers. No sign of heteroscedasticity (2 marks). Residual plot looks curved: either effect of taxon, or quadratic, or missed variable.

g) How could you improve the model in the light of the conclusions in qu 2f? (Other than carrying out the analyses described below)? (2 points): Curvature is the main problem. So could add quadtratic term, or use a Box-Cox transformation.

To follow up this initial model, two models of increasing complexity were fitted: see Figure 8 for the code.

h) Describe the differences between how model 1 and model 2 assume body mass and taxon (bird or mammal) affect SRT. Model 2 has an effect of Taxon, i.e. the intercept is different for the two taxa (2 marks).

```
> model2 <- lm(log(SRT) ~ log(Body.mass) + Taxon, data=Data)
> model3 <- lm(log(SRT) ~ log(Body.mass) * Taxon, data=Data)
```

Figure 8: R code to fit two multiple regression models to predict log(seed retention time (SRT)) against log(body mass) of a species, and its taxon (whether it is a bird or a mammal).

i) Describe the differences between how model 2 and model 3 assume body mass and taxon (bird or mammal) affect SRT. Model 3 has a Taxon:log(Body.mass) interaction, so the slopes of the effects differ (2 marks).

We have a summary of model 3 in Figure 9.

j) Draw, by hand, approximate fitted lines for the two taxa (birds and mammals), from model 3 (Figure 9). You may want to exaggerate the differences to illustrate the patterns in the fitted model. 8 marks. 2 for plotting and labelling axes properly. 2 for having positive slopes. 2 for mammals having a higher intercept. 2 for mammals having a lower slope.

k) Predict the SRT for a *T. rex* from model 3 (assuming it is a bird). 1.3222+0.54183*9 = 6.20 (1 mark). exp(1.3222+0.54183*9)/60 = 8.2 hours (1 mark).

The three models were compared in Figure 10.

l) On the basis of the comparison in Figure 10, and also the summaries of model 1 and model 3, which model do you prefer? What statistics did you take into consideration?

Either Model 1 or Model 3 (1 mark, if with justification). ANOVA shows that the data are unlikely to have arisen if either Models 1 and 2 were true, suggesting the extra complexity is due to more than chance. (2 marks: probably expressed more cleanly than this!). BUT R^2 suggest that Model 3 only explains a couple of percent more, so extra complexity doesn't make a lot of difference (2 marks).

m) Compare the predictions for SRT in *T. rex* from model 1 and model 3 (the standard error for the prediction of log(SRT) is around 0.9 in both models): what are the differences between the predictions, and how important are

```
> summary(model3)

Call:
lm(formula = log(SRT) ~ log(Body.mass) * Taxon, data = Data)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.13565 -0.53808 -0.00752  0.49367  2.12382 

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)    
(Intercept)                 1.32220    0.16185   8.169 4.59e-15 ***
log(Body.mass)              0.54183    0.02806  19.310  < 2e-16 ***
Taxonmammal                 3.11416    0.29379  10.600  < 2e-16 ***
log(Body.mass):Taxonmammal -0.25509    0.03652  -6.986 1.26e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7951 on 382 degrees of freedom
  (4 observations deleted due to missingness)
Multiple R-squared:  0.8404,	Adjusted R-squared:  0.8391 
F-statistic: 670.3 on 3 and 382 DF,  p-value: < 2.2e-16
```

Figure 9: R code and summary from a multiple regression model to predict log(seed retention time (SRT)) by log(body mass) of a species, and its taxon (bird or mammal).

they? (4 marks) Times are Model 1: 10 1/3 hours, Model 2: 8 1/5 hours. So this is an appreciable difference (if you're a T. rex!). But the uncertainty in the prediction is large, so we cannot really say whether one is closer to the truth. 2 marks for Model 1 predicting an appreciably longer time. 2 marks for seeing that the difference is small compared to the uncertainty.

```
> anova(model1, model2, model3)

Analysis of Variance Table

Model 1: log(SRT) ~ log(Body.mass)
Model 2: log(SRT) ~ log(Body.mass) + Taxon
Model 3: log(SRT) ~ log(Body.mass) * Taxon
  Res.Df    RSS Df Sum of Sq       F           Pr(>F)
1    384 341.08
2    383 272.32  1    68.765 108.785        < 2.2e-16 ***
3    382 241.47  1    30.849  48.803 0.00000000001265 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 10: R code and output from a comparison of models 1 to 3, to predict log(seed retention time (SRT)) by log(body mass) of a species, and its taxon.

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 NAs. 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 = TRUE, 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 NAs.

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 NAs would invalidate the time series attributes, and if NAs 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.fram` 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)
```