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: 10. juni 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: 3

Kontrollert av:

Dato Sign

Oppgave 1

I forbindelse med blodgiving måles blant annet hemoglobinnivået i blodet. En blodgiver mente at hans hemoglobinmålinger på ulike tilfeldig valgte måletidspunkt kunne antas tilnærmet normalfordelt med gjennomsnitt 16,14 g/dl og standardavvik 0,63 (se figur 1).

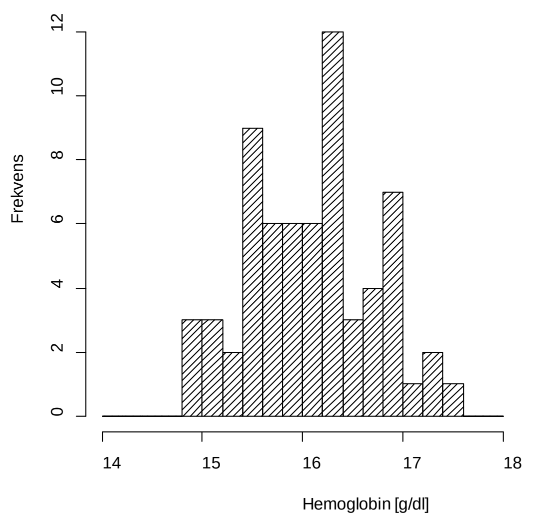
a) Virker normalfordelingsantagelsen rimelig?

Idrettsutøvere får ikke lov til å delta i konkurranser hvis hemoglobinnivået er over 17,5 g/dl (blodet blir for «tykt»). Skriv et R-uttrykk som beregner sannsynligheten for at denne blodgiveren tilfeldigvis skal få en måling over denne grenseverdien, gitt at normalfordelingsantagelsen aksepteres.

b) Skriv et R-uttrykk som finner medianen og øvre 95%-kvantil fra denne normalfordelingen.

Skisser så et R-uttrykk for hvordan du kan finne en tilnærming for den betingede sannsynligheten for å få en verdi over 17,5 g/dl, gitt at den er over 17,0 g/dl, ved å simulere fra den foreslåtte fordelingen.

c) Variasjonen i hemoglobinnivå er nok ikke helt tilfeldig; nivået kan påvirkes av væskeinntak, fysisk aktivitet, infeksjoner, stressnivå osv. Beskriv kort hvordan du ville ha gått fram for å modellere variasjonen i hemoglobinnivå.



Figur 1. Hemoglobinmålinger for en ikke-tilfeldig trøndersk blodgiver.

Oppgave 2

Et stort antall personer fra en større nasjon ble fulgt i et år, og antall dødsfall som følge av lungekreft ble registrert. Samtidig registrerte man om personene røyket eller ikke, fordelt på kategoriene «*Ikke-røker» (referansenivå)*, «*kun sigar eller pipe*», «*bare sigaretter*», «*sigaretter og pipe eller sigar*», og registrerte alderskategori (*40-44 (referansenivå), 45-49, …, 75-79, 80+*). Datasettet ser da ut som følger (kun de første radene vist – *Pers* angir antallet personer x 1000):

Alder	Røyking	Pers	Døde
40-44	Ikke	656	18
45-59	Ikke	359	22
50-54	Ikke	249	19
55-59	Ikke	632	55
60-64	Ikke	1067	117
65-69	Ikke	897	170
70-74	Ikke	668	179
75-79	Ikke	361	120
80+	Ikke	274	120
40-44	SigarPipe	145	2
45-59	SigarPipe	104	4
50-54	SigarPipe	98	3
…			

I undersøkelsen har vi dermed 656 000 personer i aldersgruppa 40-44 år som ikke røyker, hvorav 18 personer døde som følge av lungekreft i løpet av dette året. Merk at andelen personer i hver kategori ikke reflekterer andelen i hele populasjonen; røykere er overrepresentert i undersøkelsen.

De fleste kreftformer er aldersavhengig; forekomsten øker som regel med alder. For å ha en ide om hvor mye alder har å si for dødsraten for lungekreft tilpasser vi først en modell kun fra alder (dvs. ignorerer i første omgang røyking).

```
> Alder.mod<-glm(Døde~Alder, family=poisson, offset=log(Pers))
> summary(Alder.mod)

Call:
glm(formula = Døde ~ Alder, family = poisson, offset = log(Pers))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-5.1822  -1.7290  -0.7631   0.8167   4.9551

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.39572    0.05842 -58.125  < 2e-16 ***
Alder45-59   0.55603    0.07999   6.951 3.62e-12 ***
Alder50-54   0.98815    0.07681  12.864  < 2e-16 ***
Alder55-59   1.37145    0.06526  21.017  < 2e-16 ***
Alder60-64   1.62900    0.06254  26.049  < 2e-16 ***
Alder65-69   1.95715    0.06269  31.218  < 2e-16 ***
Alder70-74   2.20577    0.06410  34.409  < 2e-16 ***
Alder75-79   2.45779    0.06713  36.610  < 2e-16 ***
Alder80+     2.68749    0.07080  37.958  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)
```

```
    Null deviance: 4055.98  on 35  degrees of freedom
Residual deviance:  191.72  on 27  degrees of freedom
AIC: 449.75
```

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

b) Spiller det noen rolle for resultatet av modelleringen at antallet personer i hver kategori ikke reflekterer deres hyppighet i populasjonen?

   Hva er estimert dødsrate for lungekreft for en person i alderskategorien 40-44 år og for en person i alderskategorien 70-74 år?

c) Gi en kort forklaring av begrepet overdispersjon. Ser vi tegn til overdispersjon i dataene for vår modell, og hva kan i så fall ha forårsaket den?

   Hva kan gjøres for å forbedre modellen?

Forekomst av lungekreft har lenge vært koplet med røyking, så vi tilpasser en utvidet modell med røyking for å studere denne sammenhengen.

```
> Alder.Røyk.mod<-glm(Døde~Alder+Røyking, family=poisson, offset=log(Pers))
> summary(Alder.Røyk.mod)

Call:
glm(formula = Døde ~ Alder + Røyking, family = poisson, offset = log(Pers))

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.06055  -0.54773   0.06431   0.29963   1.48348

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)             -3.68002    0.06824 -53.929  < 2e-16 ***
Alder45-59               0.55388    0.07999   6.924 4.38e-12 ***
Alder50-54               0.98039    0.07682  12.762  < 2e-16 ***
Alder55-59               1.37946    0.06526  21.138  < 2e-16 ***
Alder60-64               1.65423    0.06257  26.439  < 2e-16 ***
Alder65-69               1.99817    0.06279  31.824  < 2e-16 ***
Alder70-74               2.27141    0.06435  35.296  < 2e-16 ***
Alder75-79               2.55858    0.06778  37.746  < 2e-16 ***
Alder80+                 2.84692    0.07242  39.310  < 2e-16 ***
RøykingSigaretter        0.41696    0.03991  10.447  < 2e-16 ***
RøykingSigaretterSigPip  0.21796    0.03869   5.633 1.77e-08 ***
RøykingSigarPipe         0.04781    0.04699   1.017    0.309
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 4055.984  on 35  degrees of freedom
Residual deviance:   21.487  on 24  degrees of freedom
AIC: 285.51
```

d) Skriv opp denne modellen i matematisk notasjon. Forklar kort hva dummy-variable er, og hvorfor de er nødvendige når vi tilpasser en modell med kategoriske forklaringsvariable.

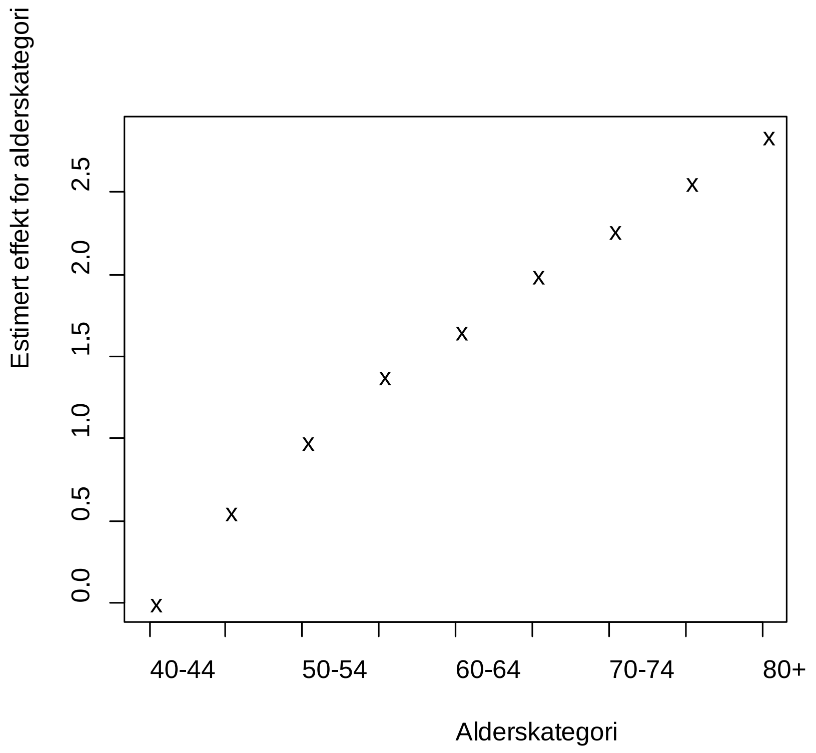
e) Ser det ut som om modellen med røyking er bedre enn modellen med kun alder som forklaringsvariabel? Skriv et R-uttrykk som beregner p-verdien for en test av om det er grunnlag for å hevde at røyking skal være med i modellen.

f) Hvordan ser modellen `Alder.Røyk.mod` ut med hensyn til overdispersjon?

Hvor mye større er raten for lungekreft for en person som kun røyker sigaretter enn for en ikke-røyker?

g) Residualanalyse er en viktig del av kvalitetsvurderingen for en modell. Beskriv kort hva du ville sett etter i en residualanalyse.

Du er fortsatt ikke helt fornøyd med modellen og ønsker å forbedre den ytterligere. Blant annet antyder figuren under (figur 2) en enkel sammenheng mellom dødelighet og alder. Kom med noen forslag til videre forbedringer av modellen, enten ved å redusere eller øke antall parametere.



Figur 2. Estimert koeffisient for hver alderskategori for modellen `Alder.Røyk.mod` fra punkt d).

The Normal Distribution

Description

Density, distribution function, quantile function and random generation for the normal distribution with mean equal to `mean` and standard deviation equal to `sd`.

Usage

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE,log.p =
FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p =
FALSE)
rnorm(n, mean = 0, sd = 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.
`mean`	vector of means.
`sd`	vector of standard deviations.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are *P[X ≤ x]* otherwise, *P[X > x]*.

Details

If `mean` or `sd` are not specified they assume the default values of `0` and `1`, respectively.

The normal distribution has density

f(x) = 1/(√(2 π) σ) e^-((x - μ)^2/(2 σ^2))

where *μ* is the mean of the distribution and *σ* the standard deviation.

Value

`dnorm` gives the density, `pnorm` gives the distribution function, `qnorm` gives the quantile function, and `rnorm` generates random deviates.

The length of the result is determined by `n` for `rnorm`, 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.

For `sd = 0` this gives the limit as `sd` decreases to 0, a point mass at `mu`. `sd < 0` is an error and returns `NaN`.

See Also

Distributions for other standard distributions, including dlnorm for the *Log*normal distribution.

Examples

```
require(graphics)

dnorm(0) == 1/sqrt(2*pi)
dnorm(1) == exp(-1/2)/sqrt(2*pi)
dnorm(1) == 1/sqrt(2*pi*exp(1))

## Using "log = TRUE" for an extended range :
par(mfrow = c(2,1))
plot(function(x) dnorm(x, log = TRUE), -60, 50,
     main = "log { Normal density }")
curve(log(dnorm(x)), add = TRUE, col = "red", lwd = 2)
mtext("dnorm(x, log=TRUE)", adj = 0)
mtext("log(dnorm(x))", col = "red", adj = 1)

plot(function(x) pnorm(x, log.p = TRUE), -50, 10,
     main = "log { Normal Cumulative }")
curve(log(pnorm(x)), add = TRUE, col = "red", lwd = 2)
mtext("pnorm(x, log=TRUE)", adj = 0)
mtext("log(pnorm(x))", col = "red", adj = 1)

## if you want the so-called 'error function'
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
## (see Abramowitz and Stegun 29.2.29)
## and the so-called 'complementary error function'
erfc <- function(x) 2 * pnorm(x * sqrt(2), lower = FALSE)
## and the inverses
erfinv <- function (x) qnorm((1 + x)/2)/sqrt(2)
erfcinv <- function (x) qnorm(x/2, lower = FALSE)/sqrt(2)
```

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 containNAs. 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)
```