Topics in Statistical Sciences 2 – Exam exercise 1

Søren Højsgaard and Torben Tvedebrink

Version: 28/09/2017 11:02

This exercise is about the generalized linear models as discussed in lectures 1–3 of Topics in Statistical Sciences 2. During the oral exam you will have 20 min to present the exercise. You decide what topics to cover and how to present them, however, we will ask questions to any part of the covered curricula, exercise and presentation.

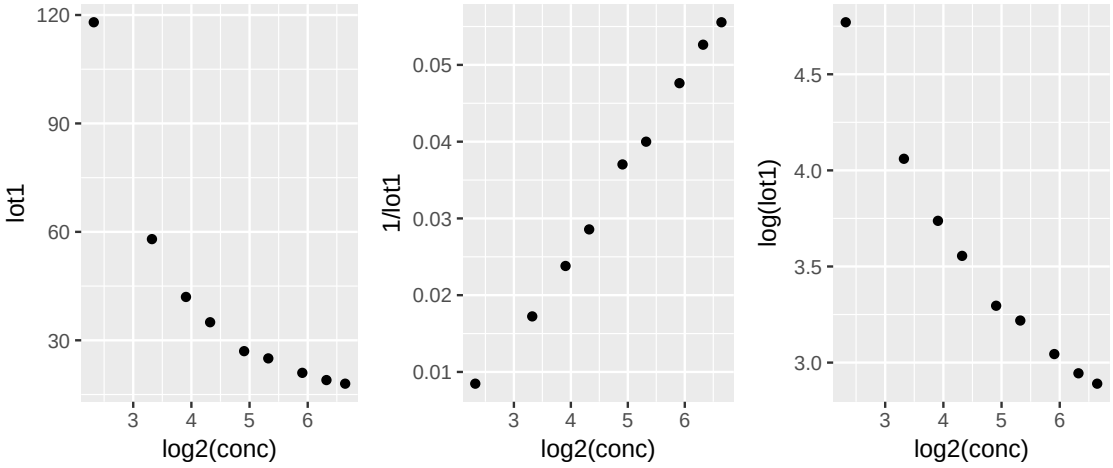
1 Iteratively reweighted least squares for fitting a gamma model

Remember we looked at these data

```
ct <- read.table("../data/clottingTime.txt", header=T)
ct
```

```
##   conc lot1 lot2
## 1    5  118   69
## 2   10   58   35
## 3   15   42   26
## 4   20   35   21
## 5   30   27   18
## 6   40   25   16
## 7   60   21   13
## 8   80   19   12
## 9  100   18    1
```

```
library(ggplot2)
source("multiplot.R")
multiplot(qplot(log2(conc), lot1, data=ct),
        qplot(log2(conc), 1/lot1, data=ct),
        qplot(log2(conc), log(lot1), data=ct), cols=3)
```



We considered these models:

```
g1a <- glm(lot1 ~ log2(conc), family=Gamma("inverse"), data=ct)
g1a
```

```
##
## Call:  glm(formula = lot1 ~ log2(conc), family = Gamma("inverse"), data = ct)
##
## Coefficients:
## (Intercept)   log2(conc)
##    -0.01655      0.01064
##
## Degrees of Freedom: 8 Total (i.e. Null);  7 Residual
## Null Deviance:     3.513
## Residual Deviance: 0.01673  AIC: 37.99
```

```
g1b <- glm(lot1 ~ log2(conc), family=Gamma("log"), data=ct)
g1b
```

```
##
## Call:  glm(formula = lot1 ~ log2(conc), family = Gamma("log"), data = ct)
##
## Coefficients:
## (Intercept)   log2(conc)
##      5.5032       -0.4172
##
## Degrees of Freedom: 8 Total (i.e. Null);  7 Residual
## Null Deviance:     3.513
## Residual Deviance: 0.1626  AIC: 58.48
```

```
g1c <- glm(lot1 ~ log2(conc), family=Gamma("identity"), data=ct)
g1c
```

```
##
## Call:  glm(formula = lot1 ~ log2(conc), family = Gamma("identity"),
##     data = ct)
##
## Coefficients:
## (Intercept)   log2(conc)
##       99.25       -12.74
##
## Degrees of Freedom: 8 Total (i.e. Null);  7 Residual
## Null Deviance:     3.513
## Residual Deviance: 0.6085  AIC: 70.43
```

This exam exercise consists of the following components

1. Implement in R a function called `fit_gamma` which has the following form:

```
fit_gamma <- function(formula, link="inverse", phi=NULL, w=NULL){
    ## Your code goes here...
    ## More hints follow below...
}
```

   The code should run at the time of exam in case we provide you with an extra dataset.

2. Explain the steps in the code alongside with the corresponding theory as explained in the lecture notes.
3. Fit the clotting time data `glm` and `fit_gamma` using different links: "inverse", "identity" and "log". Dou you get the same result?
4. Do the models fit well to data? You may want to consider, among other things, the residuals and the fitted values.
5. Using both `glm` and `fit_gamma`, create a larger model with log2(dose) and log2(dose) squared as predictors. Test the smaller model with only log2(dose) against the larger model with log2(dose) and log2(dose) squared as predictors

1.1 Requirements to your implementation

Regarding `fit_gamma`, the minimal requirements are

Input:

- formula: a model formula
- link: the link function; default is the canonical link, but other link functions should be allowed as well.
- phi: the dispersion parameter. If NULL, phi is estimated from data; if phi is a single positive number then this value is used; otherwise and error is signaled.
- weights: If NULL, weights is taken to be a vector of 1's; otherwise the weights are used as given.

Output: A list with the following components:

- coef: regression coefficients
- vcov: the variance-covariance matrix of the regression coefficients
- p: the number of regression coefficients
- resid: the pearson residuals½
- fit: the fitted values

1.2 Hints to your implementation

Much work is done by existing R functions that you are welcome to use:

```
lm.fit()   ## least squares fit
lm.wfit()  ## weighted least squares fit
```

The `Gamma` function provides much of what you need:

```
f <- Gamma("inverse")
f$link     # name of link
```

```
## [1] "inverse"
```

```
f$linkfun  # g(mu); eta = g(mu)
```

```
## function (mu)
## 1/mu
## <environment: namespace:stats>
```

```
f$linkinv  # h(eta) = g^{-1}(eta); mu=h(eta)
```

```
## function (eta)
## 1/eta
## <environment: namespace:stats>
```

```
f$mu.eta   # h'(eta); "deta" / "dmu"
```

```
## function (eta)
## -1/(eta^2)
## <environment: namespace:stats>
```

```
f$variance # V(mu)
```

```
## function (mu)
## mu^2
## <bytecode: 0x58e3750>
## <environment: 0x4664888>
```

You will need `g'(mu)`; an easy option is to use `Deriv()` from the `Deriv` package:

```
library(Deriv)
Deriv(f$linkfun, "mu")
```

```
## function (mu)
## -(1/mu^2)
## <environment: namespace:stats>
```

A vector y of responses and the model matrix X can be obtained from a formula e.g. as:

```
formula <- lot1 ~ log2(conc)
y <- model.response(model.frame(formula, data=ct))
X <- model.matrix(formula, data=ct)
y
```

```
##   1   2   3   4   5   6   7   8   9
## 118  58  42  35  27  25  21  19  18
```

```
X
```

```
##   (Intercept) log2(conc)
## 1           1   2.321928
## 2           1   3.321928
## 3           1   3.906891
## 4           1   4.321928
## 5           1   4.906891
## 6           1   5.321928
## 7           1   5.906891
## 8           1   6.321928
## 9           1   6.643856
## attr(,"assign")
## [1] 0 1
```

1.3 A skeleton

For your help we provide the following skeleton of the function:

```r
fit_gamma <- function(formula, data, link="inverse", phi=NULL, w=NULL){

    y <- model.response(model.frame(formula, data=data))
    X <- model.matrix(formula, data=data)

    f <- Gamma(link)
    g <- f$linkfun
    g.inv <- f$linkinv
    g.der <- Deriv::Deriv(g, "mu")
    V <- f$variance

    if (is.null(w))
        w <- rep(1, length(y))

    conv.eps <- 1e-12

    ## Get started; intitial value of z and beta
    z <- NULL
    beta <- NULL

    ## Iterate until beta no longer changes
    repeat{

        eta <- NULL
        mu <- NULL
        z <- NULL
        v <- NULL
        beta <- NULL

        # break the loop when beta stops changing between succesive
        # iterations.

        break

    }

    ## Compute mu and working weights v after final iteration

    ## Estimate phi if necessary

    ## Compute estimated covariance matrix of regression parameters

    ## Compute Pearson residuals

    out <- list( ## Fill in your values
        coef = NULL,
        vcov = NULL,
        phi  = NULL,
        resid= NULL,
        fit  = NULL,
        p    = NULL)

    out
}
```