GLM computational details

Ben Bolker

September 26, 2018

Computational details of IRLS

Coding IRLS

A `family` object in R is coded as a list of useful components:

```
pfamily <- poisson()
names(pfamily)

##  [1] "family"     "link"       "linkfun"    "linkinv"    "variance"
##  [6] "dev.resids" "aic"        "mu.eta"     "initialize" "validmu"
## [11] "valideta"   "simulate"

pfamily$variance

## function (mu)
## mu
## <bytecode: 0x5f96450>
## <environment: 0x5f9bc98>

pfamily$linkinv

## function (eta)
## pmax(exp(eta), .Machine$double.eps)
## <environment: namespace:stats>
```

It's not *too* hard to write your own GLM function: the hard parts are figuring out what to do about special situations (tricky starting values, poor convergence, etc..)

```
myglmfit <- function(y,X,family,tol=1e-8,maxit=50) {
    mu <- y  ## set initial values
    ## set up 'oldbeta' and 'beta' so they're not identical
    oldbeta <- rep(0,ncol(X))
    beta    <- rep(1,ncol(X))
    it <- 1  ## number of iterations
    while (it < maxit && max(abs((1-beta/oldbeta)))>tol) {
        oldbeta <- beta
```

```
        eta <- family$linkfun(mu)    ## calc. linear predictor
        mm <- family$mu.eta(eta)     ## calc. d(mu)/d(eta)
        adjdev <- eta + (y-mu)/mm    ## adjusted response
        W <- c(1/(mm^2*family$variance(mu)))  ## weights
        beta <- lm.wfit(X,adjdev,W)$coefficients  ## weighted least-squares
        mu <- family$linkinv(X %*% beta)          ## compute new mu
        it <- it+1                                ## update
    }
    beta
}
X <- model.matrix(~wool*tension,data=warpbreaks)
y <- warpbreaks$breaks
myglmfit(y,X,poisson())

##    (Intercept)          woolB       tensionM       tensionH woolB:tensionM
##      3.7967368     -0.4566272     -0.6186830     -0.5957987      0.6381768
## woolB:tensionH
##      0.1883632

coef(glm(breaks~wool*tension,data=warpbreaks,family=poisson))

##    (Intercept)          woolB       tensionM       tensionH woolB:tensionM
##      3.7967368     -0.4566272     -0.6186830     -0.5957987      0.6381768
## woolB:tensionH
##      0.1883632
```

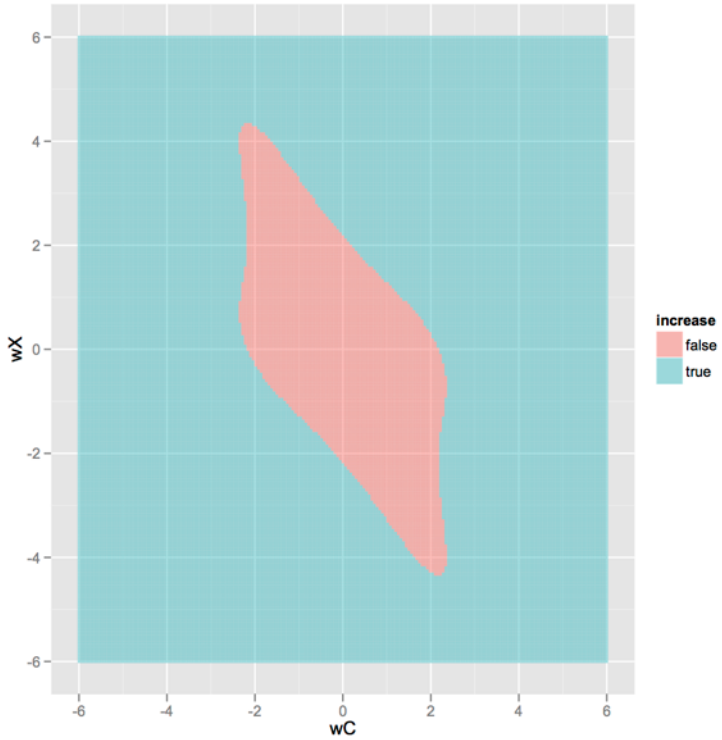
A bad example

GLM likelihood is *log-concave* with a unique solution, so in principle we shouldn't have a problem. But the IRLS algorithm doesn't always get us there, if the data are bad enough (a more common problem is when the MLEs are infinite . . . we'll discuss this situation later).

John Mount shows the results of

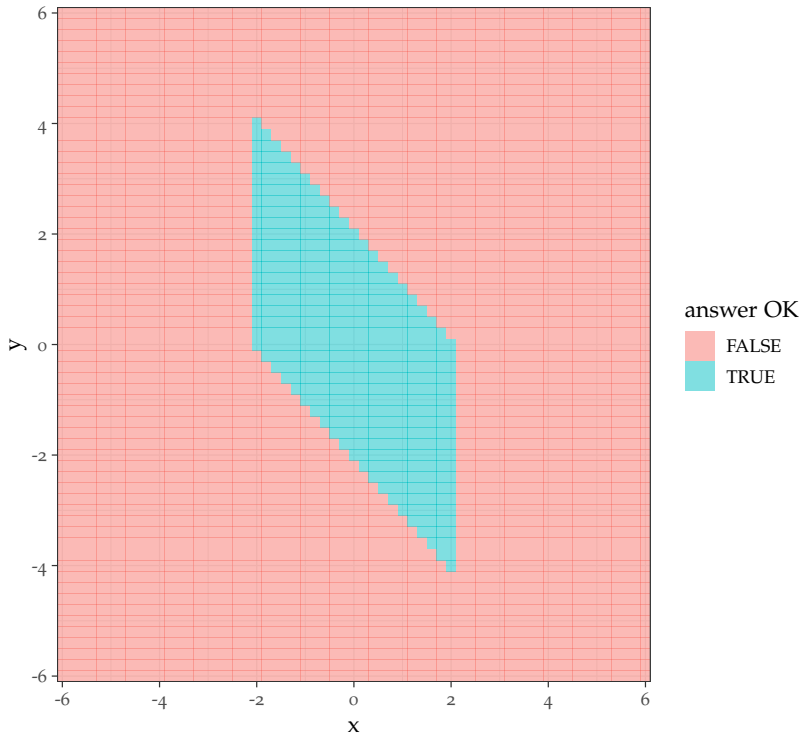
```
p <- data.frame(x=c(1,0,1,0),y=c(TRUE,TRUE,FALSE,FALSE))
coef(glm(y~x,data=p,family=binomial,start=c(0,0)))

## (Intercept)           x
##           0           0

badstartfun <- function(start) {
    cc <- coef(glm(y~x,data=p,family=binomial,start=start))
    sum(cc^2)>1e-12
}
badstartfun(c(0,0))
```

```
## [1] FALSE
```



I did it by brute force, using ‘emdbook::curve3d()‘ and the ‘glm()‘ function:

```
library(ggplot2)
theme_set(theme_bw())
brkvec <- seq(-6,6,by=2) ## for compatibility with previous pllot
ggplot(ccm,aes(x,y,fill=!value))+geom_tile(alpha=0.5)+
    scale_fill_discrete(name="answer OK")+
    scale_x_continuous(expand=c(0,0),breaks=brkvec)+
    scale_y_continuous(expand=c(0,0),breaks=brkvec)
```

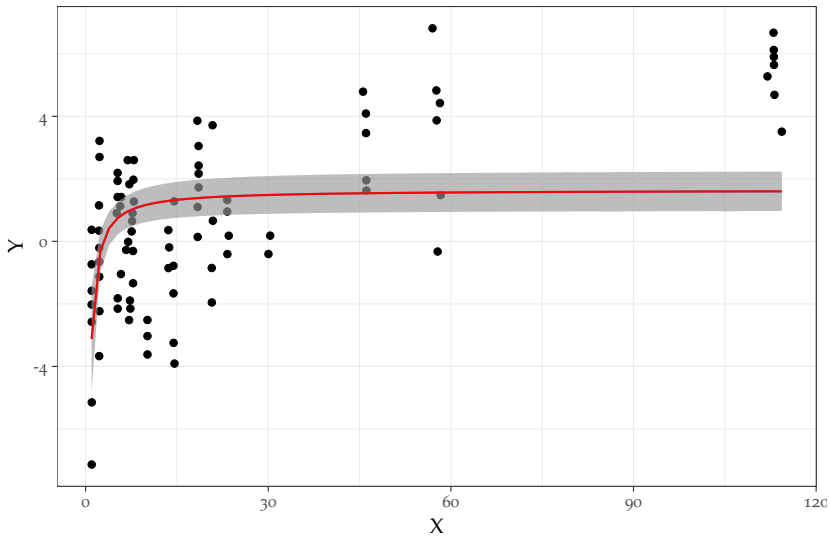


Another bad example

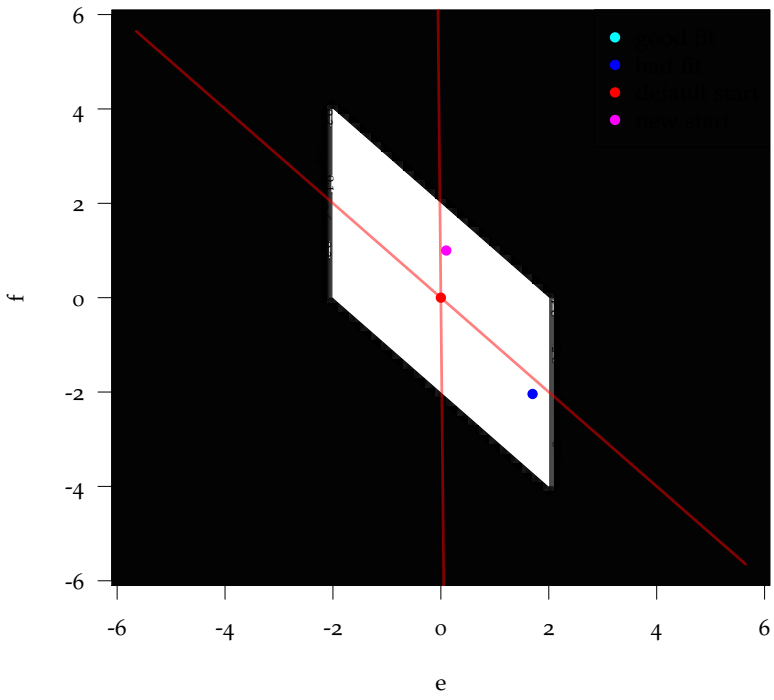
Fitting a *Beverton-Holt* model (*Michaelis-Menten, Monod, ...*): $y = ax/(b+x)$

Inverse-link trick: $1/y = (b+x)/ax = (b/a)(1/x) + (1/a)$:
`glm(y ~ I(1/x), family=gaussian(link="inverse"))`

```
L <- load("../data/bevholt_ex.RData")
g1 <- ggplot(dat,aes(X,Y)) + geom_point()
g1 + geom_smooth(method = "glm", family = gaussian(link = "inverse"),
                 formula = y ~ I(1/x), start = c(0.01, 1))+
    geom_smooth(method = "glm", family = gaussian(link = "inverse"),
                 formula = y ~ I(1/x), colour="red")

## Warning:  Ignoring unknown parameters:  family, start
## Warning:  Ignoring unknown parameters:  family
```




```
## Warning in readChar(con, 5L, useBytes = TRUE): cannot
open compressed file 'bevholt_ex.RData', probable reason 'No
such file or directory'
## Error in readChar(con, 5L, useBytes = TRUE): cannot open
the connection
## Error in ff(x, y):  object 'dat' not found
```


6
4
2
0
-2
-4
-6
f
-6
-4
-2
0
2
4
6
e