Tail correction in the 'egp family

H. Rue

October 4, 2024

This is a short note to document the tail correction implemented in the *egp* family. The likelihood has the following form

$$F(y) = \left[(1-(1+\xi\frac{y}{\sigma})^{-\frac{1}{\xi}}\right]^{\kappa} \qquad y \geq 0$$

where

$$\sigma = \frac{\xi\exp(\eta)}{(1-\alpha^{1/\kappa})^{-\xi}-1}$$

for the linear predictor η and a given quantile level α. Normally we would require $-1/2 < \xi < 1/2$.

The "tail' ' issue comes with $\xi < 0$ when

$$1-(1+\xi\frac{y}{\sigma})^{-1/\xi} = 0$$

and lower, which impose a critical value for σ depending $(\eta, \xi, \alpha, \kappa)$. Values of σ below this critical value has zero density, or $-\infty$ log-density.

Since we are looking at this conditionally on the hyperparameters, then this impose a critical value for η, $\eta_c = \eta_c(\xi, \alpha, \kappa)$. Our aim is is to discuss how an adaptive penalty is implemented allowing for values of $\eta < \eta_c()$ with a smooth behavior.

Assume $\xi < 0$, and define

$$a = (1-\alpha^{1/\kappa})^{-\xi} - 1$$

then

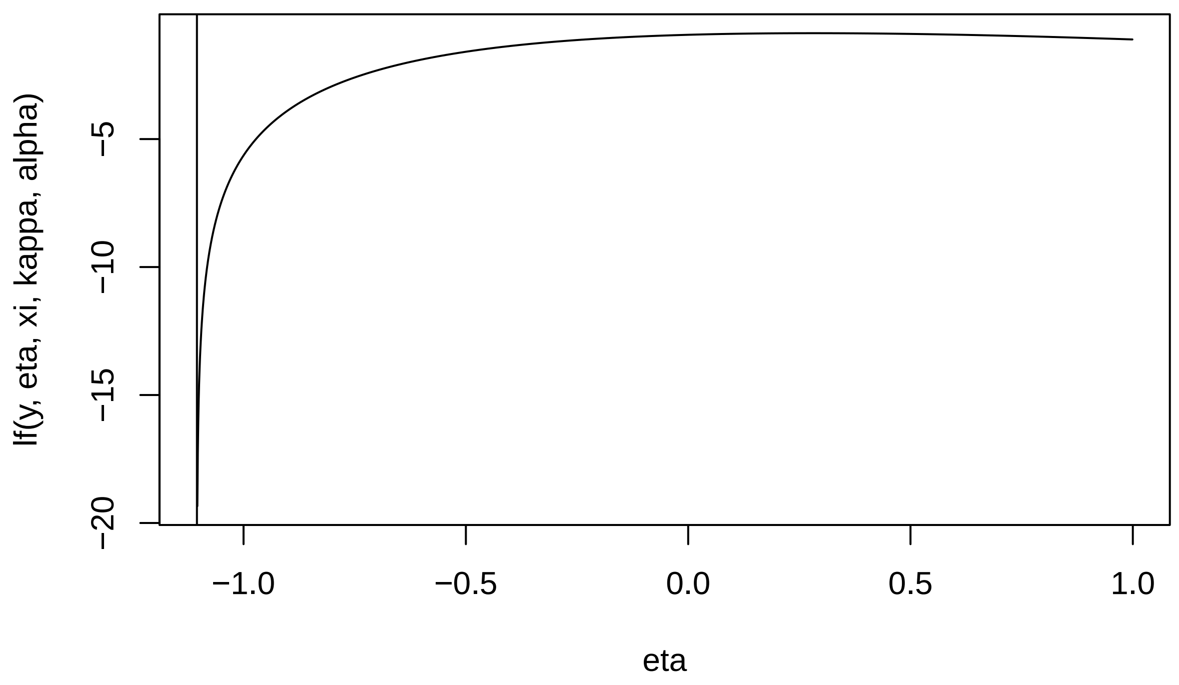
$$\eta_c = \log(-ay).$$

We can plot the log-density to see the issue.

```r
lf <- function(y, eta, xi, kappa, alpha) {
    xii <- -1/xi
    q <- exp(eta)
    a <- ((1-alpha^(1/kappa))^(-xi) -1)
    sigma <- xi * q / a
    yy <- 1 + xi * y / sigma
    idx.fail <- which(yy < 0)
    yy[idx.fail] <- 1
    lyy <- log(yy)
    lsigma <- log(sigma)
    values <- (log(kappa) + (kappa-1) * log1p(-yy^xii) - lsigma + (xii-1) * lyy)
    values[idx.fail] <- -Inf
    return (values)
}
```

and then

```r
xi <- -0.25
y <- 1
alpha <- 0.8
kappa <- 1
a <- ((1-alpha^(1/kappa))^(-xi) -1)
eta.c <- log(-a*y)
eta <- seq(eta.c, 1, by=0.001)
plot(eta, lf(y, eta, xi, kappa, alpha), type="l")
abline(v=eta.c)
```

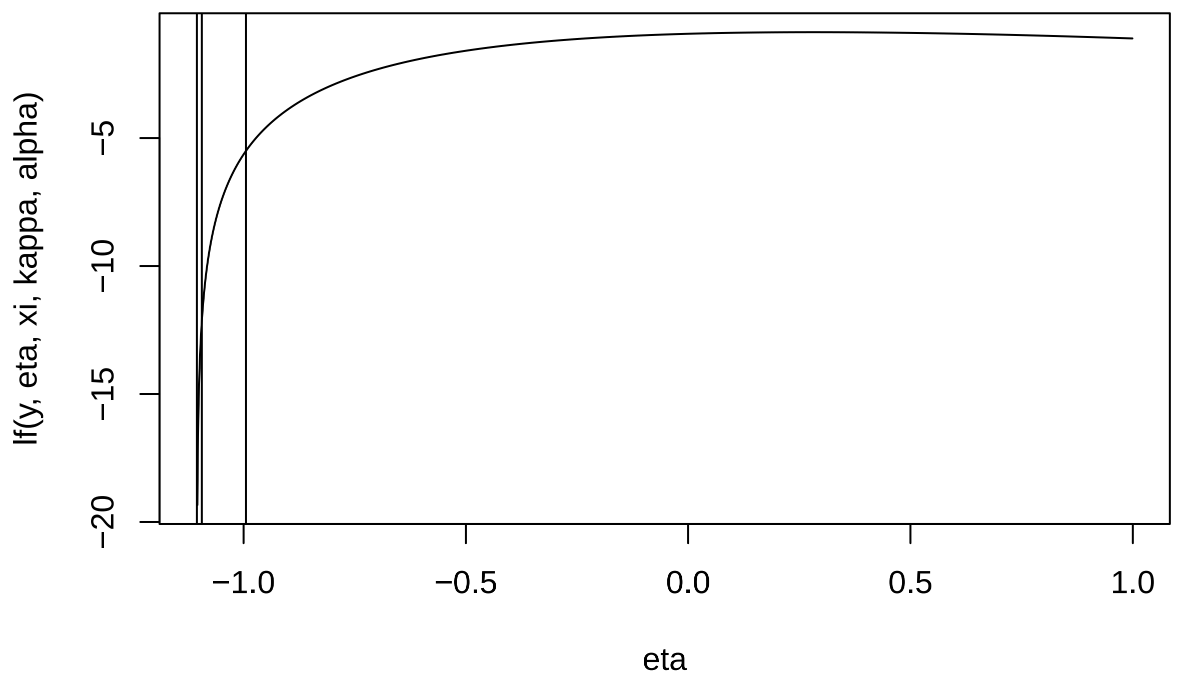


To check the local Taylor-expansion, then the function ‘c012’ returns the exact taylor expansion of the log-likelihood around a location η (we source only as its a little to long to print)

```r
source("c012.R")
```

Here is the “trick’ ’. Define

```r
ff <- c(0.90, 0.99)
if (eta.c > 0) {
    eta.H <- 1/ff[1] * eta.c
    eta.L <- 1/ff[2] * eta.c
} else {
    eta.H <- ff[1] * eta.c
    eta.L <- ff[2] * eta.c
}
plot(eta, lf(y, eta, xi, kappa, alpha), type="l")
abline(v=eta.c)
abline(v=eta.L)
abline(v=eta.H)
```

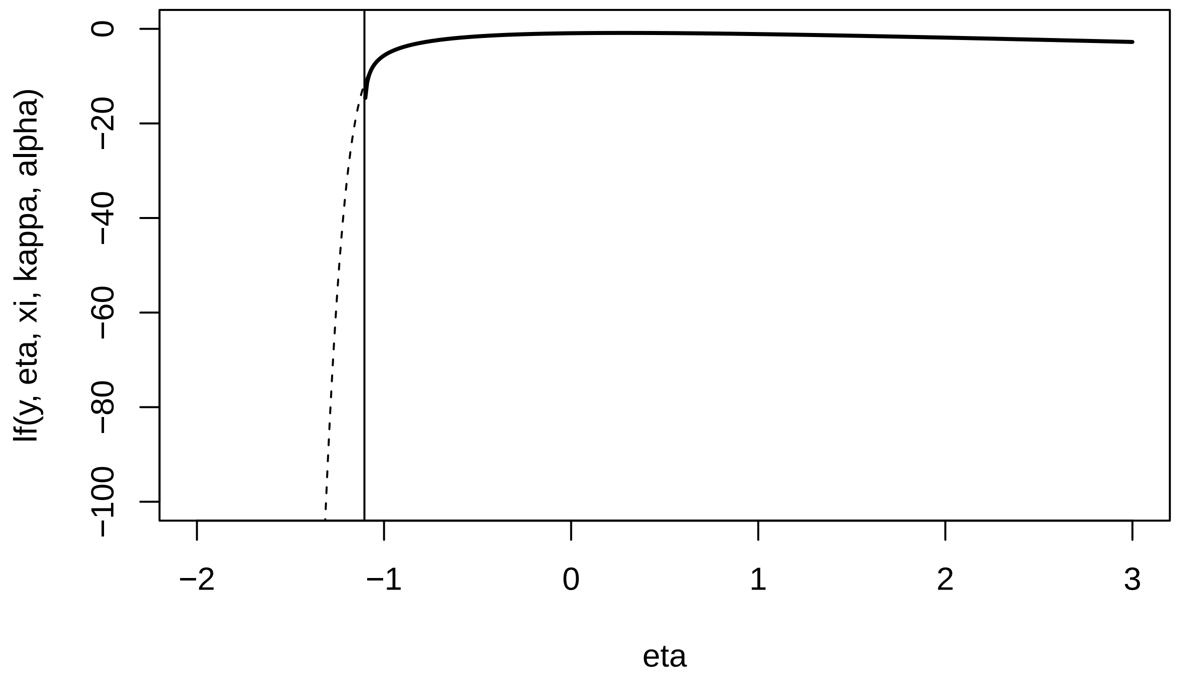


We can use the Taylor-expansion in the interval (η_L, η_H), we just need to map the interval $(-\infty, \eta_H)$ into that interval.

```
eta.star <- function(eta, eta.L, eta.H) {
    ee <- (eta.H - eta) / (eta.H - eta.L)
    return (eta.H + (eta.L - eta.H) * ee/(1+ee))
}
```

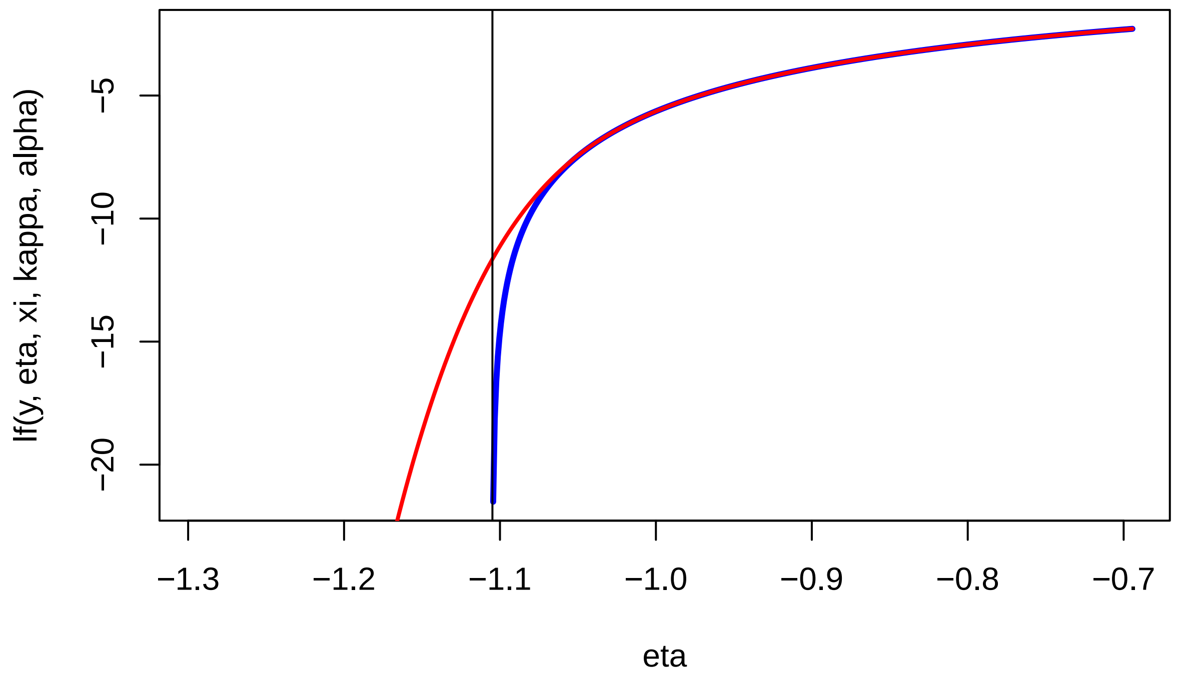
```
lf.new <- function(y, eta, xi, kappa, alpha, ff=c(0.90,0.99)) {
    xii <- -1/xi
    q <- exp(eta)
    a <- ((1-alpha^(1/kappa))^(-xi) -1)
    sigma <- xi * q / a
    yy <- 1 + xi * y / sigma
    yy[yy <= 0] <- 1
    lyy <- log(yy)
    lsigma <- log(sigma)

    values <- rep(NA, length(eta))
    if (xi > 0) {
        values <- (log(kappa) + (kappa-1) * log1p(-yy^xii) - lsigma + (xii-1) * lyy)
        return (values)
    } else {
        eta.c <- log(-a*y)
        if (eta.c > 0) {
            eta.H <- 1/ff[1] * eta.c
            eta.L <- 1/ff[2] * eta.c
        } else {
            eta.H <- ff[1] * eta.c
            eta.L <- ff[2] * eta.c
        }
        idx.fail <- which(eta <= eta.H)
        idx.ok <- which(eta > eta.H)

        if (length(idx.ok) > 0) {
            values[idx.ok] <- (log(kappa) +
                (kappa-1) * log1p(-yy[idx.ok]^xii) -
                    lsigma[idx.ok] + (xii-1) * lyy[idx.ok])
        }

        if (length(idx.fail) > 0) {
            eeta <- eta.star(eta[idx.fail], eta.L, eta.H)
            cc <- c012(eeta, y, xi, kappa, alpha)
            values[idx.fail] <- (cc$c0 + (eta[idx.fail] - eeta) * cc$c1
                + 0.5 * cc$c2 * (eta[idx.fail] - eeta)^2)
         }
         return (values)
    }
}
```

```
eta <- seq(-2, 3, by=0.01)
plot(eta, lf(y, eta, xi, kappa, alpha), type="l", ylim=c(-100,0), lwd=2)
lines(eta, lf.new(y, eta, xi, kappa, alpha), lty=2)
abline(v=eta.c)
```



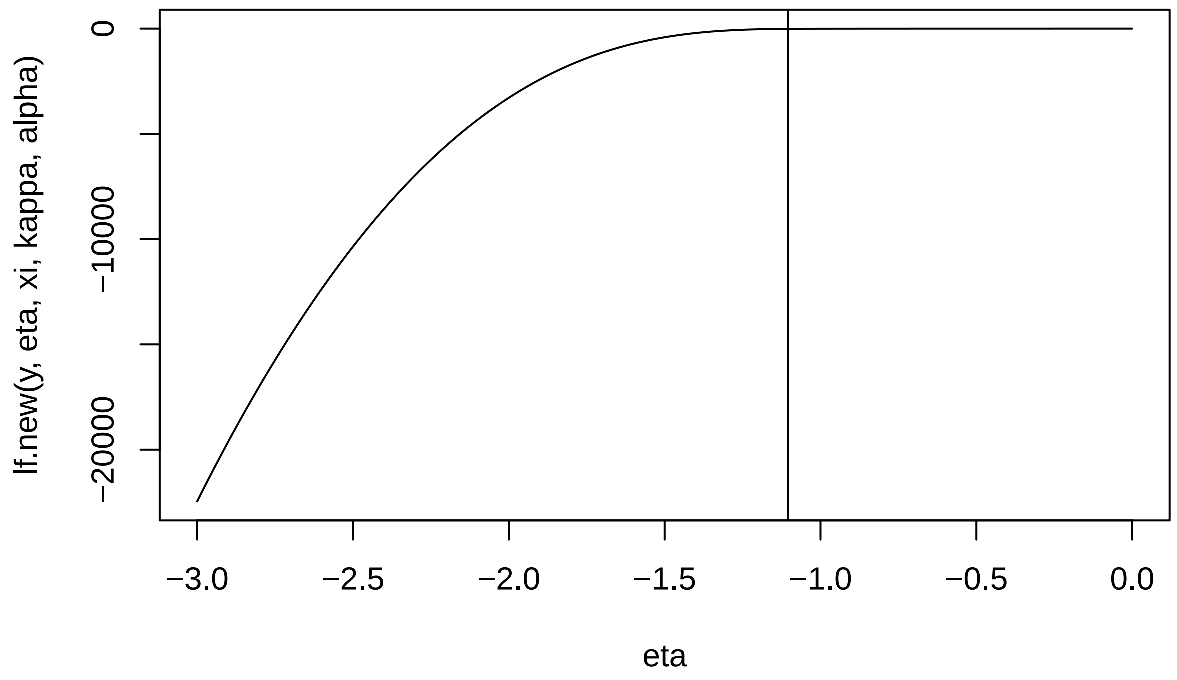
We can zoom into the η_H location

```r
eta <- eta.H + seq(-0.3, 0.3, by=0.001)
plot(eta, lf(y, eta, xi, kappa, alpha), type="l",lwd=3, col="blue")
lines(eta, lf.new(y, eta, xi, kappa, alpha), lwd=2, col="red")
abline(v=eta.c)
```



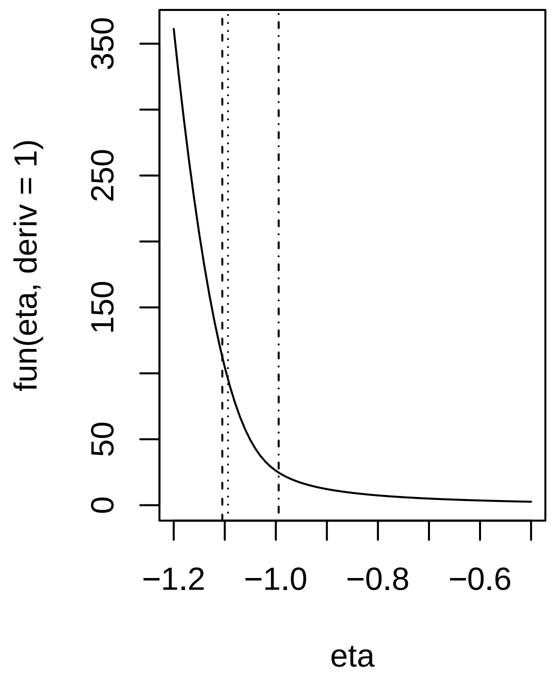
or zoom out

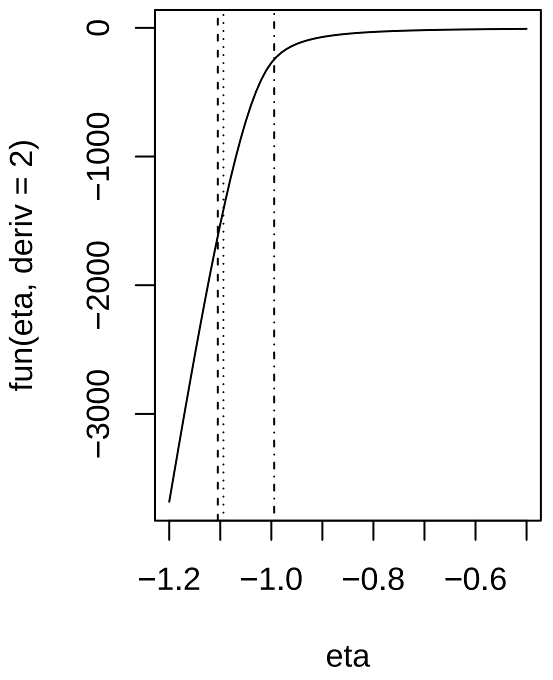
```
eta <- seq(-3, 0, by=0.01)
plot(eta, lf.new(y, eta, xi, kappa, alpha), type="l")

abline(v=eta.c)
```



and also check the first two derivatives

```
eta <- seq(-1.2, -0.5, by=0.01)
fun <- splinefun(eta, lf.new(y, eta, xi, kappa, alpha))
par(mfrow=c(1,2))
plot(eta, fun(eta, deriv=1), type="l")
abline(v=eta.c,lty=2)
abline(v=eta.L,lty=3)
abline(v=eta.H,lty=4)
plot(eta, fun(eta, deriv=2), type="l")
abline(v=eta.c,lty=2)
abline(v=eta.L,lty=3)
abline(v=eta.H,lty=4)
```


fun(eta, deriv = 1)
0
50
150
250
350
−1.2
−1.0
−0.8
−0.6
eta


fun(eta, deriv = 2)
0
−1000
−2000
−3000
−1.2
−1.0
−0.8
−0.6
eta