Hardness of wood is important but difficult to measure. Density is easier to measure. Can hardness be predicted from density? Data and some discussion from `http://gbi.agrsci.dk/statistics/courses/phd07/misc/learn/tex/Glm/gamma/handout.pdf`.

```
> timber = read.table("timber.txt", header = T)
> attach(timber)
> hardness

 [1]  484  427  413  517  549  648  587  704  979  914 1070 1020 1210  989 1160
[16] 1010 1100 1130 1270 1180 1400 1760 1710 2010 1880 1980 1820 2020 1980 2310
[31] 1940 3260 2700 2890 2740 3140

> density

 [1] 24.7 24.8 27.3 28.4 28.4 29.0 30.3 32.7 35.6 38.5 38.8 39.3 39.4 39.9 40.3
[16] 40.6 40.7 40.7 42.9 45.8 46.9 48.2 51.5 51.5 53.4 56.0 56.5 57.3 57.6 59.2
[31] 59.8 66.0 67.4 68.8 69.1 69.1

> scatter.smooth(density, hardness)
> scatter.smooth(density, log(hardness))
> timber.lm.1 = lm(hardness ~ density + I(density^2))
> scatter.smooth(fitted(timber.lm.1), residuals(timber.lm.1))
> abline(h = 0, lty = "dashed")
> scatter.smooth(fitted(timber.lm.1), abs(residuals(timber.lm.1)))
> scatter.smooth(log(fitted(timber.lm.1)), log(abs(residuals(timber.lm.1))))
```

The residual plots suggest that the variance is not constant, but increases as the mean increases. Possibly a gamma glm is more appropriate.

```
> timber.glm.log = glm(hardness ~ density + I(density^2), data = timber,
+     family = Gamma(link = "log"))
> summary(timber.glm.log)

Call:
glm(formula = hardness ~ density + I(density^2), family = Gamma(link = "log"),
    data = timber)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.22023  -0.06138  -0.01605   0.07130   0.18986

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.147e+00  2.089e-01  19.848  < 2e-16 ***
density       9.133e-02  9.316e-03   9.803 2.66e-11 ***
I(density^2) -5.210e-04  9.776e-05  -5.329 6.98e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
(Dispersion parameter for Gamma family taken to be 0.01017894)

    Null deviance: 11.26788  on 35  degrees of freedom
Residual deviance:  0.33499  on 33  degrees of freedom
AIC: 455.64

Number of Fisher Scoring iterations: 4

> scatter.smooth(fitted(timber.glm.log), abs(residuals(timber.glm.log,
+      "pearson")))
```

The residual plots for the gamma glm look better. Next look at an ordinary linear model with logged response versus the gamma glm with log link. You'll notice there isn't much difference. Perhaps this is because for the dispersion parameters in this dataset, the gamma and lognormal densities look very similar.

```
> timber.lm.logged = lm(log(hardness) ~ density + I(density^2),
+      data = timber)
> summary(timber.glm.log)

Call:
glm(formula = hardness ~ density + I(density^2), family = Gamma(link = "log"),
    data = timber)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.22023  -0.06138  -0.01605   0.07130   0.18986

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.147e+00  2.089e-01  19.848  < 2e-16 ***
density       9.133e-02  9.316e-03   9.803 2.66e-11 ***
I(density^2) -5.210e-04  9.776e-05  -5.329 6.98e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Gamma family taken to be 0.01017894)

    Null deviance: 11.26788  on 35  degrees of freedom
Residual deviance:  0.33499  on 33  degrees of freedom
AIC: 455.64

Number of Fisher Scoring iterations: 4

> summary(timber.lm.logged)
```

```
Call:
lm(formula = log(hardness) ~ density + I(density^2), data = timber)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.22331 -0.05708 -0.01104  0.07500  0.18871 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)   4.138e+00  2.087e-01  19.828  < 2e-16 ***
density       9.152e-02  9.305e-03   9.835 2.45e-11 ***
I(density^2) -5.228e-04  9.764e-05  -5.354 6.49e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1 

Residual standard error: 0.1008 on 33 degrees of freedom
Multiple R-squared: 0.9723,	Adjusted R-squared: 0.9706 
F-statistic: 578.9 on 2 and 33 DF,  p-value: < 2.2e-16 

> scatter.smooth(fitted(timber.lm.logged), abs(residuals(timber.lm.logged,
+     "pearson")))
> xxx = seq(0, 2, 0.01)
> plot(xxx, dgamma(xxx, shape = 1/0.0102, scale = 0.0102), type = "l")
> plot(xxx, dlnorm(xxx, meanlog = 0, sdlog = 0.1008), type = "l")
> detach(timber)
```

Next fit some gamma glms “by hand.” The data here are on survival times and white blood cell counts for leukemia patients. Data from `http://www.sci.usq.edu.au/staff/dunn/Datasets/Books/Hand/Hand-R/leukwbc-R.html`

```
> leuk = read.table("leukwbc-R.dat", header = T)
> attach(leuk)
> WBC

 [1]   2300    750   4300   2600   6000  10500  10000  17000   5400   7000
[11]   9400  32000  35000 100000 100000  52000 100000   4400   3000   4000
[21]   1500   9000   5300  10000  19000  27000  28000  31000  26000  21000
[31]  79000 100000 100000

> Time

 [1]  65 156 100 134  16 108 121   4  39 143  56  26  22   1   1   5  65  56  65
[20]  17   7  16  22   3   4   2   3   8   4   3  30   4  43

> Dev = 1
> DeltaDev = 1
> glm.out = matrix(0, nrow = 0, ncol = 3)
> dimnames(glm.out) = list(NULL, c("deviance", "beta0", "beta1"))
> mu = (Time + mean(Time))/2
> eta = mu
> while (abs(DeltaDev) > 1e-08) {
+     w = 1/mu^2
+     z = eta + (Time - mu)
+     lmod = lm(z ~ WBC, weights = w)
+     eta = lmod$fit
+     mu = eta
+     OldDev = Dev
+     Dev = 2 * sum((Time - mu)/mu - log(Time/mu))
+     DeltaDev = Dev - OldDev
+     glm.out = rbind(glm.out, c(Dev, coef(lmod)[1], coef(lmod)[2]))
+ }
> glm.out
```

```
       deviance    beta0         beta1
 [1,] 105.85717 18.68645 -0.0001322015
 [2,]  54.15655 47.46267 -0.0002658700
 [3,]  54.11750 48.98845 -0.0003007976
 [4,]  54.11086 48.44135 -0.0002876404
 [5,]  54.11000 48.64837 -0.0002925387
 [6,]  54.10988 48.57139 -0.0002907057
 [7,]  54.10986 48.60021 -0.0002913904
 [8,]  54.10986 48.58945 -0.0002911345
 [9,]  54.10986 48.59347 -0.0002912301
[10,]  54.10986 48.59197 -0.0002911944
[11,]  54.10986 48.59253 -0.0002912077
```

```
> leuk.glm.identity = glm(Time ~ WBC, data = leuk, family = Gamma(link = "identity"))
> coef(leuk.glm.identity)
```

```
  (Intercept)           WBC
48.5918057181 -0.0002911905
```

```
> deviance(leuk.glm.identity)
```

```
[1] 54.10986
```

```
> Dev = 1
> DeltaDev = 1
> glm.out = matrix(0, nrow = 0, ncol = 3)
> dimnames(glm.out) = list(NULL, c("deviance", "beta0", "beta1"))
> mu = (Time + mean(Time))/2
> eta = 1/mu
> while (abs(DeltaDev) > 1e-08) {
+     w = mu^2
+     z = eta - (Time - mu)/mu^2
+     lmod = lm(z ~ WBC, weights = w)
+     eta = lmod$fit
+     mu = 1/eta
+     OldDev = Dev
+     Dev = 2 * sum((Time - mu)/mu - log(Time/mu))
+     DeltaDev = Dev - OldDev
+     glm.out = rbind(glm.out, c(Dev, coef(lmod)[1], coef(lmod)[2]))
+ }
> glm.out
```

```
     deviance      beta0        beta1
[1,] 53.63451 0.01354250 3.073437e-07
[2,] 51.20016 0.01528393 4.598223e-07
[3,] 50.99119 0.01531096 5.433133e-07
[4,] 50.98866 0.01525590 5.552754e-07
[5,] 50.98866 0.01525495 5.554444e-07
[6,] 50.98866 0.01525495 5.554444e-07
```

```
> leuk.glm.inverse = glm(Time ~ WBC, data = leuk, family = Gamma(link = "inverse"))
> coef(leuk.glm.inverse)
```

```
 (Intercept)          WBC
1.525495e-02 5.554444e-07
```

```
> deviance(leuk.glm.inverse)
```

```
[1] 50.98866
```

```
> Dev = 1
> DeltaDev = 1
> glm.out = matrix(0, nrow = 0, ncol = 3)
> dimnames(glm.out) = list(NULL, c("deviance", "beta0", "beta1"))
> mu = (Time + mean(Time))/2
> eta = log(mu)
> while (abs(DeltaDev) > 1e-08) {
+     w = rep(1, length(Time))
+     z = eta + (Time - mu)/mu
+     lmod = lm(z ~ WBC, weights = w)
+     eta = lmod$fit
+     mu = exp(eta)
+     OldDev = Dev
+     Dev = 2 * sum((Time - mu)/mu - log(Time/mu))
+     DeltaDev = Dev - OldDev
+     glm.out = rbind(glm.out, c(Dev, coef(lmod)[1], coef(lmod)[2]))
+ }
> glm.out
```

```
      deviance    beta0         beta1
 [1,] 57.27951 3.587142 -9.794857e-06
 [2,] 53.25847 4.046923 -1.197747e-05
 [3,] 53.08843 3.951407 -1.079142e-05
 [4,] 53.08568 3.957758 -1.109752e-05
 [5,] 53.08546 3.955074 -1.100695e-05
 [6,] 53.08544 3.955876 -1.103454e-05
 [7,] 53.08544 3.955632 -1.102618e-05
 [8,] 53.08544 3.955706 -1.102872e-05
 [9,] 53.08544 3.955683 -1.102795e-05
[10,] 53.08544 3.955690 -1.102818e-05
```

```
> leuk.glm.log = glm(Time ~ WBC, data = leuk, family = Gamma(link = "log"))
> coef(leuk.glm.log)
```

```
  (Intercept)           WBC
 3.955701e+00 -1.102857e-05
```

```
> deviance(leuk.glm.log)
```

```
[1] 53.08544
```

```
> detach(leuk)
```

Inverse Gaussian example: Data on projected and actual sales. Data and some discussion from "Extending the linear model with R" by Julian Faraway.

```
> plot(actual ~ projected, data = cpd)
> cpd.lm = lm(actual ~ projected - 1, data = cpd)
> abline(cpd.lm)
> plot(residuals(cpd.lm) ~ fitted(cpd.lm))
> plot(residuals(cpd.lm) ~ log(fitted(cpd.lm)))
> cpd.glm = glm(actual ~ projected - 1, data = cpd, family = inverse.gaussian(link = "iden
> plot(residuals(cpd.glm) ~ log(fitted(cpd.glm)))
```

Plotting versus log of fitted values just spreads the points out horizontally. The variance of the residuals from the linear model seems increasing, but the inverse Gaussian model "overcorrects" and the variance of the residuals from this model is clearly decreasing.