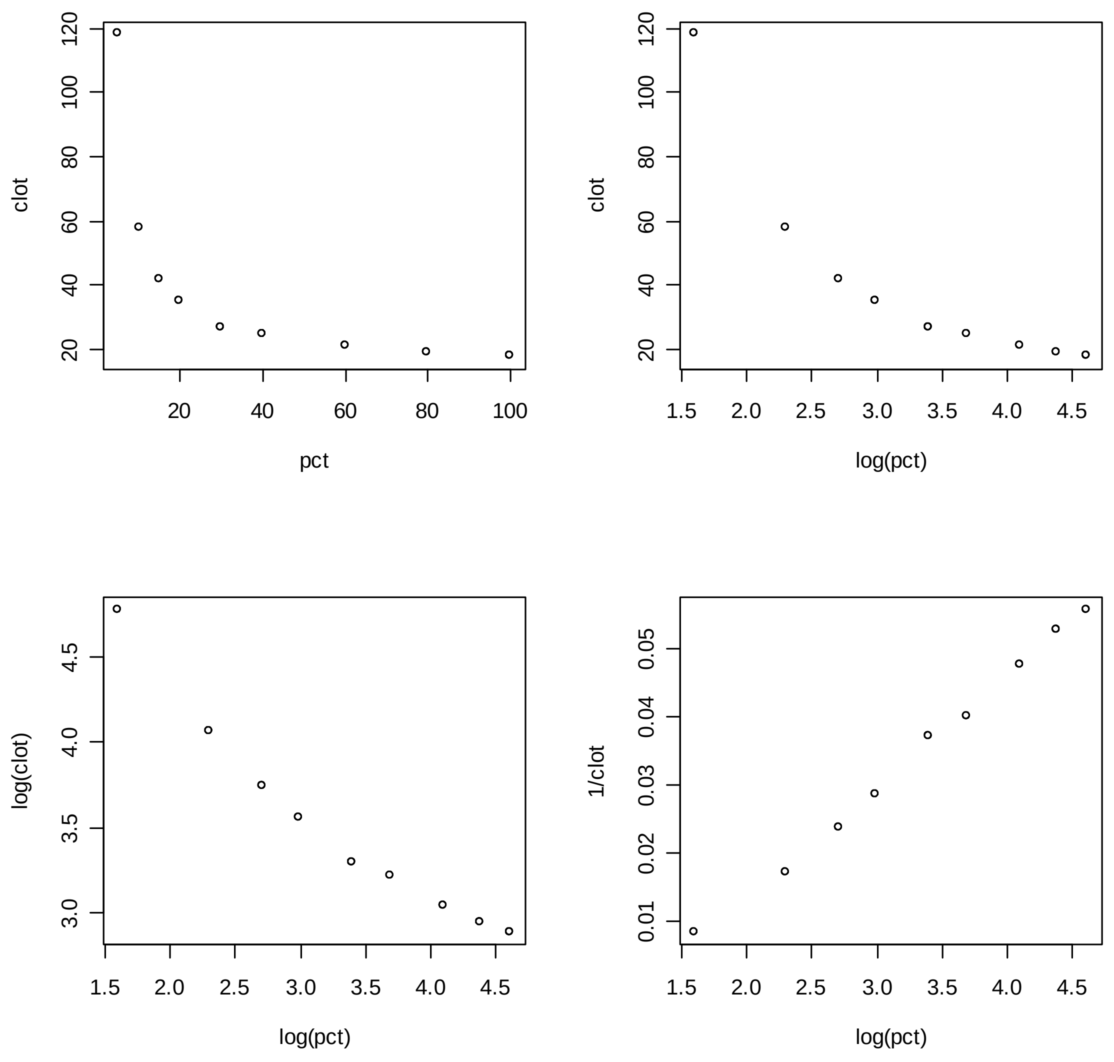
CLOTTING TIME DATA: Gamma Regression

Analysis based on discussion in M. Davidian's longitudinal analysis book

In the development of clotting agents, it is common to perform in vitro studies of time to clotting. The following data are reported in McCullagh and Nelder (1989, section 8.4.2), and are taken from such a study. Here, samples of normal human plasma were diluted to one of 9 different percentage concentrations with prothrombin-free plasma; the higher the dilution, the more the interference with the blood's ability to clot, because the blood's natural clotting capability has been weakened. For each sample, clotting was induced by introducing thromboplastin, a clotting agent, and the time until clotting occurred was recorded (in seconds). 5 samples were measured at each of the 9 percentage concentrations, and the mean clotting times were averaged; thus, the response is mean clotting time over the 5 samples. The response is plotted against percentage concentration in the upper left panel of the first plot. We will discuss the other panels of the figure shortly. It is well-recognized that this type of response, which is by its nature always positive, does not exhibit the same variability at all levels. Rather, large responses tend to be more variable than small ones, and a constant coefficient of variation model, such as the gamma, is often a suitable model for this nonconstant variation.

From the plot, it is clear that a straight-line model for mean response as a function of the percentage concentration would be inappropriate. With this type of data it is often the case that the concentrations are log-transformed, but the plot of the clotting times against the log(percentage concentration) also shows a non-linear relationship. A quadratic model with a log-transformed predictor might seem better, but, because such models eventually curve "back up," this might not be a good model, either. In the lower right and lower left panels, the reciprocals (1/y) and logarithms (log y) of the response, respectively, are plotted against log (percentage concentration). These appear to be roughly like straight lines, the former more-so than the latter. We will return to the implications of these two plots for choosing a model for mean response shortly. Note, of course, that a sensible model for mean response would be one that honors the positivity restriction for the response. Also noticeable from the plot is that the data are of "high quality" - the pattern of change in the response with log(percentage concentration) is very clear and smooth, with very little "noise." This would suggest that if the data really are well-represented by the gamma probability distribution, then the coefficient of variation is "small." From the plot, it is very difficult to see any evidence of that the variance really is nonconstant as the response changes - this is due to the fact that variation is just so small, so it is hard to pick up by eye.



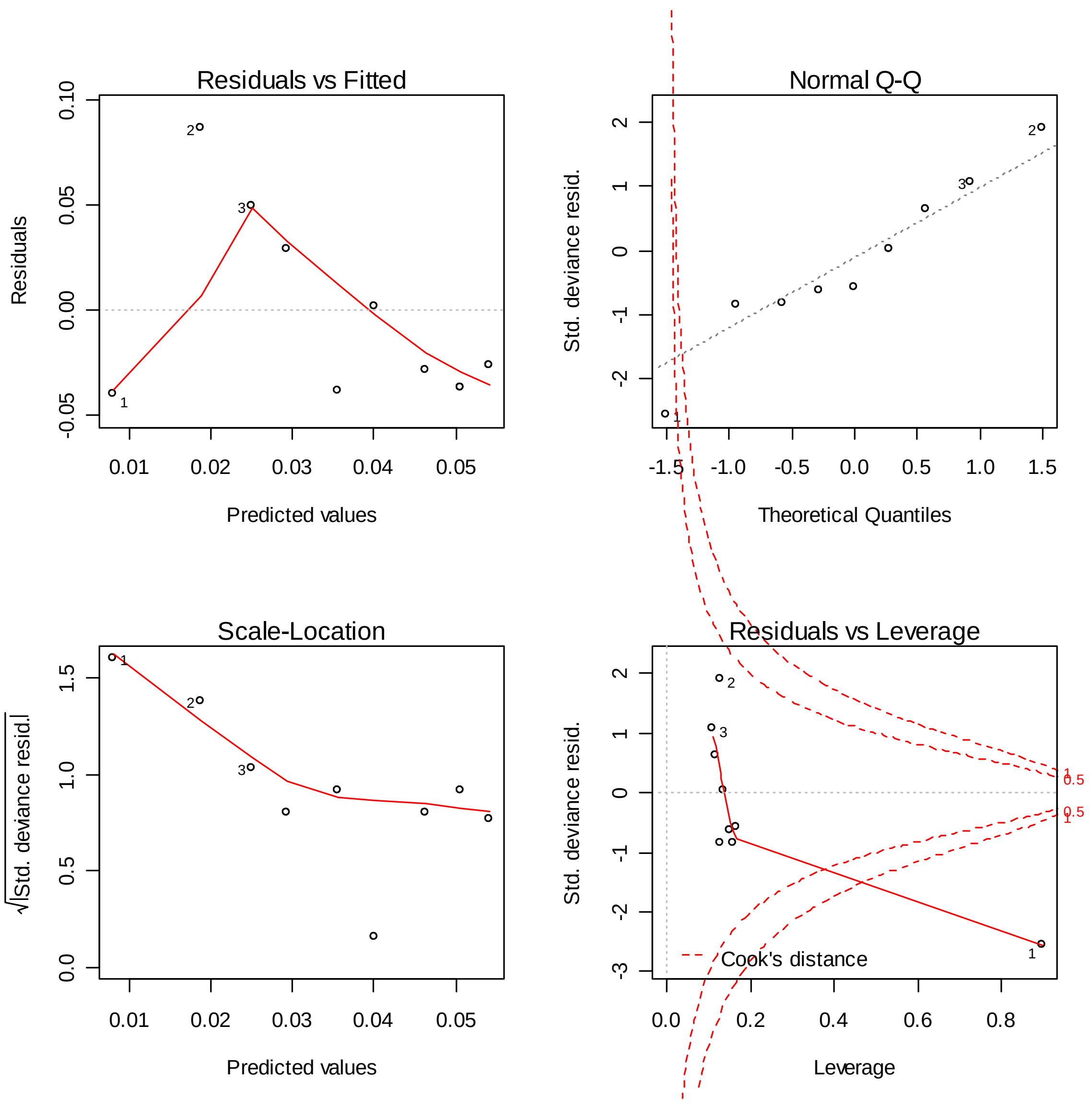
As a first step in the analysis, I fit a gamma regression model with an inverse link, using the log-transformed percentage as a predictor. Output from the model is given below, along with code that defined the data and created the plots above.

```
> pct = c(5,10,15,20,30,40,60,80,100)
  clot = c(118,58,42,35,27,25,21,19,18)

  par(mfrow=c(2,2))
  plot(pct,clot)
  plot(log(pct),clot)
  plot(log(pct),log(clot))
  plot(log(pct),1/clot)
```

```
>  ct = data.frame(pct,clot)
>  ct

  pct clot

1   5  118
2  10   58
3  15   42
4  20   35
5  30   27
6  40   25
7  60   21
8  80   19
9 100   18

> ft = glm(formula = clot ~ log(pct), family = Gamma(link = "inverse"), data = ct)

> gg = summary(ft)
> names(gg)

     "call"          "terms"          "family"         "deviance"   "aic"            "contrasts"
     "df.residual"    "null.deviance"   "df.null"        "iter"    "deviance.resid" "coefficients"
     "aliased"        "dispersion"     "df"    "cov.unscaled"      "cov.scaled"

> gg

Call: glm(formula = clot ~ log(pct), family = Gamma(link = "inverse"), data = ct)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.04008  -0.03756  -0.02637   0.02905   0.08641

Coefficients:
                Estimate Std. Error    t value   Pr(>|t|)
(Intercept) -0.0165       0.000927  -17.85     4.28e-07
log(pct)       0.0153       0.000415   36.98      2.75e-09


(Dispersion parameter for Gamma family taken to be 0.002446013)

    Null deviance:    3.5128 on 8 degrees of freedom
Residual deviance: 0.0167 on 7 degrees of freedom
AIC: 37.99
```

The log(pct) effect is highly significant, and the estimated dispersion parameter, which for the gamma is the estimated coefficient of variation (CV) is tiny: .002. This indicates that the estimated variation for the fitted gamma model, relative to the mean, is small – indicative of the "high quality" alluded to earlier. The diagnostic plots show some evidence of lack of fit (where?), but the overall fit is fairly good, as we will see in some plots to come.

```
> par(mfrow=c(2,2))
> plot(ft)
```



The next several commands illustrate some features of the fitted model that can be obtained and plotted, in particular CIs for the estimated mean response.

```
> beta = ft$coefficients
```
Store regression coefficients (as 2-by-1 vector)

```
> beta

(Intercept)    log(pct)

-0.01655438  0.01534311
```

```
> cov = gg$cov.scaled
```
Store cov matrix of regression coeffs as 2-by-2 matrix

```
> cov
                  (Intercept)        log(pct)
(Intercept)  8.603265e-07 -3.606389e-07
log(pct)     -3.606389e-07  1.721883e-07
```

```
> x = 1:100
```
Create design matrix for pct of 1 to 100

```
> xd = cbind(rep(1,100),log(x))
> xd[1:5,]

     [,1]     [,2]
[1,]    1 0.0000
[2,]    1 0.6931
[3,]    1 1.0986
[4,]    1 1.3862
[5,]    1 1.6094
```

```
>    lp = xd %*% beta
```
Creates estimated linear predictor Xd*beta

```
>  vlp = xd %*% cov %*% t(xd)
```
Cov matrix of linear predictor Xd*cov*Xd’

```
>     q = sqrt( diag(vlp) )
```
SE of linear predictors, as vector

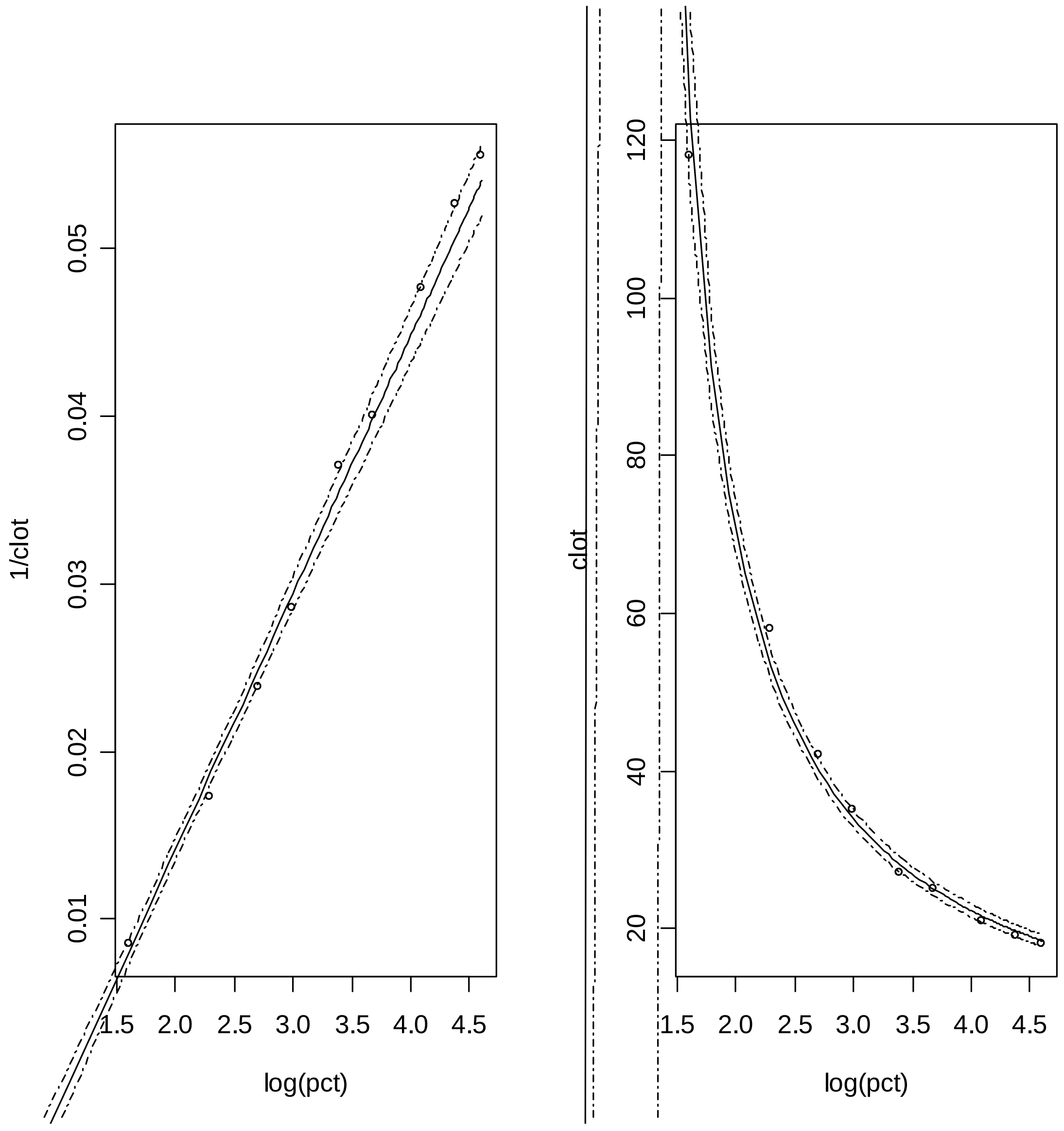
```
> low = lp - 1.96*q
>  up  = lp + 1.96*q
```
Approx lower and upper 95% CI for linear pred

Replot data on inverse response scale, then fit and CI on inverse scale

```
>  par(mfrow=c(1,2))
>  plot(log(pct),1/clot);
>  lines(log(x),lp); lines(log(x),up,lty=4);  lines(log(x),low,lty=4)
```

Plot data on original response scale, and use inverse function applied to linear predictor and CI

```
> plot(log(pct),clot)
>  lines(log(x),1/lp); lines(log(x),1/up,lty=4);   lines(log(x),1/low,lty=4)
```



Notice how well the model appears to fit, either on the transformed scale or on the original scale.

As a second analysis, I refit the gamma regression model, but with a log-link, and the log-transformed predictor.

```
> ft = glm(formula =clot~log(pct), family = Gamma(link = "log"), data = ct)
> summary(ft)

Call:
glm(formula = clot ~ log(pct), family = Gamma(link = "log"), data = ct)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.15599  -0.13254  -0.05629   0.07988   0.24588

Coefficients:
             Estimate   Std. Error    t value    Pr(>|t|)
(Intercept)  5.50323    0.19030       28.92   1.52e-08
log(pct)     -0.60192    0.05531      -10.88    1.22e-05

 (Dispersion parameter for Gamma family taken to be 0.02435409)

    Null deviance:    3.51283  on 8  degrees of freedom
Residual deviance: 0.16261  on 7  degrees of freedom
AIC: 58.482

Number of Fisher Scoring iterations: 5
```

Notice that the log(pct) effect is still highly significant, but there are some suggestions here that the model does not fit as well as the previous fit: the residual deviance is very small, but about 10 times larger than previously, which is reflected in the much larger estimated dispersion parameter.

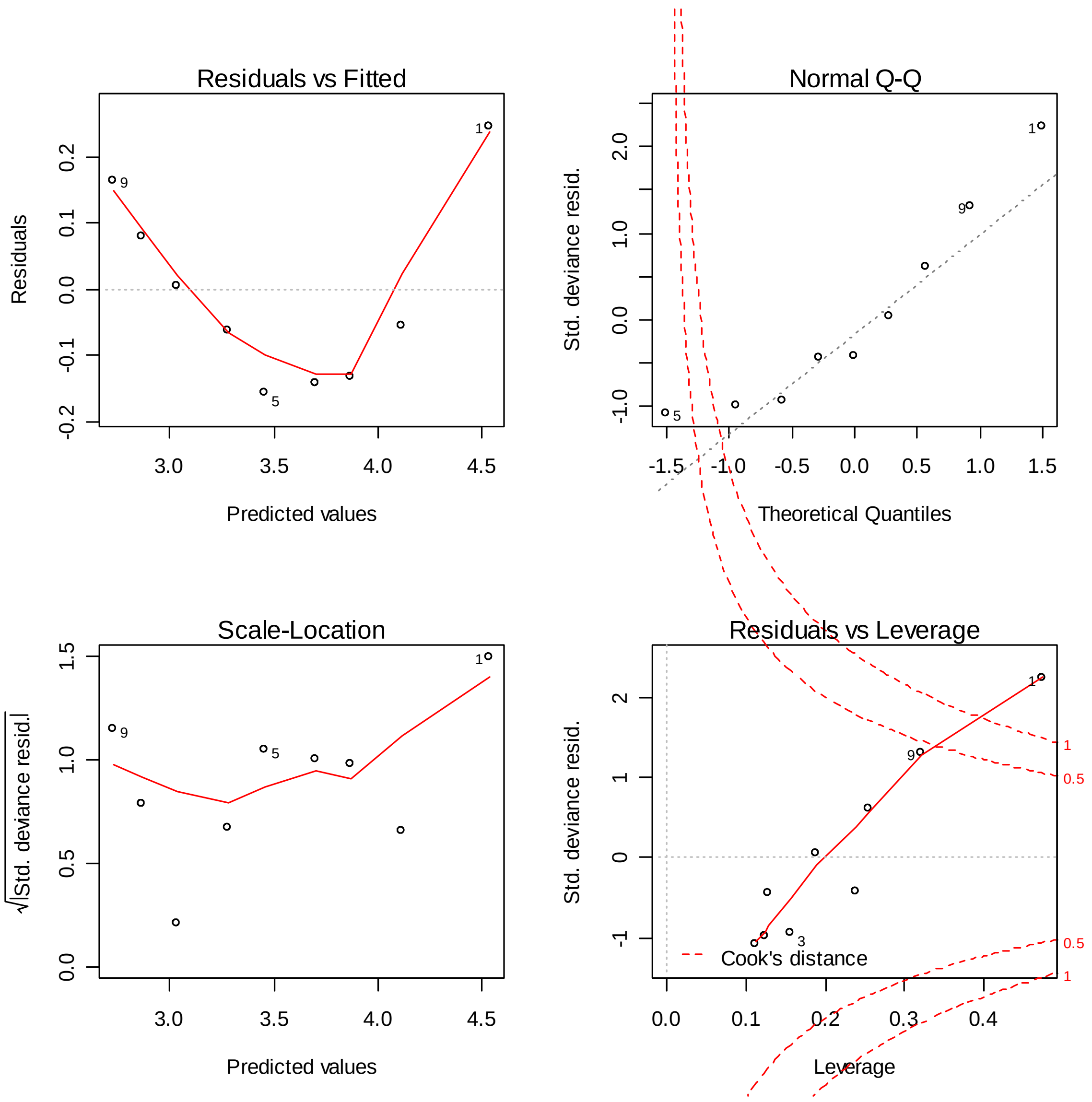
Diagnostics plots, and a plot of the observed and fitted model, with CI on both the transformed scale and the original scale, are given below. What evidence do you see to support a conclusion that the original model fits better? Discuss

Based on these analyses, I would use the first model for inferences and predictions. In particular, the estimated mean clotting time satisfies:

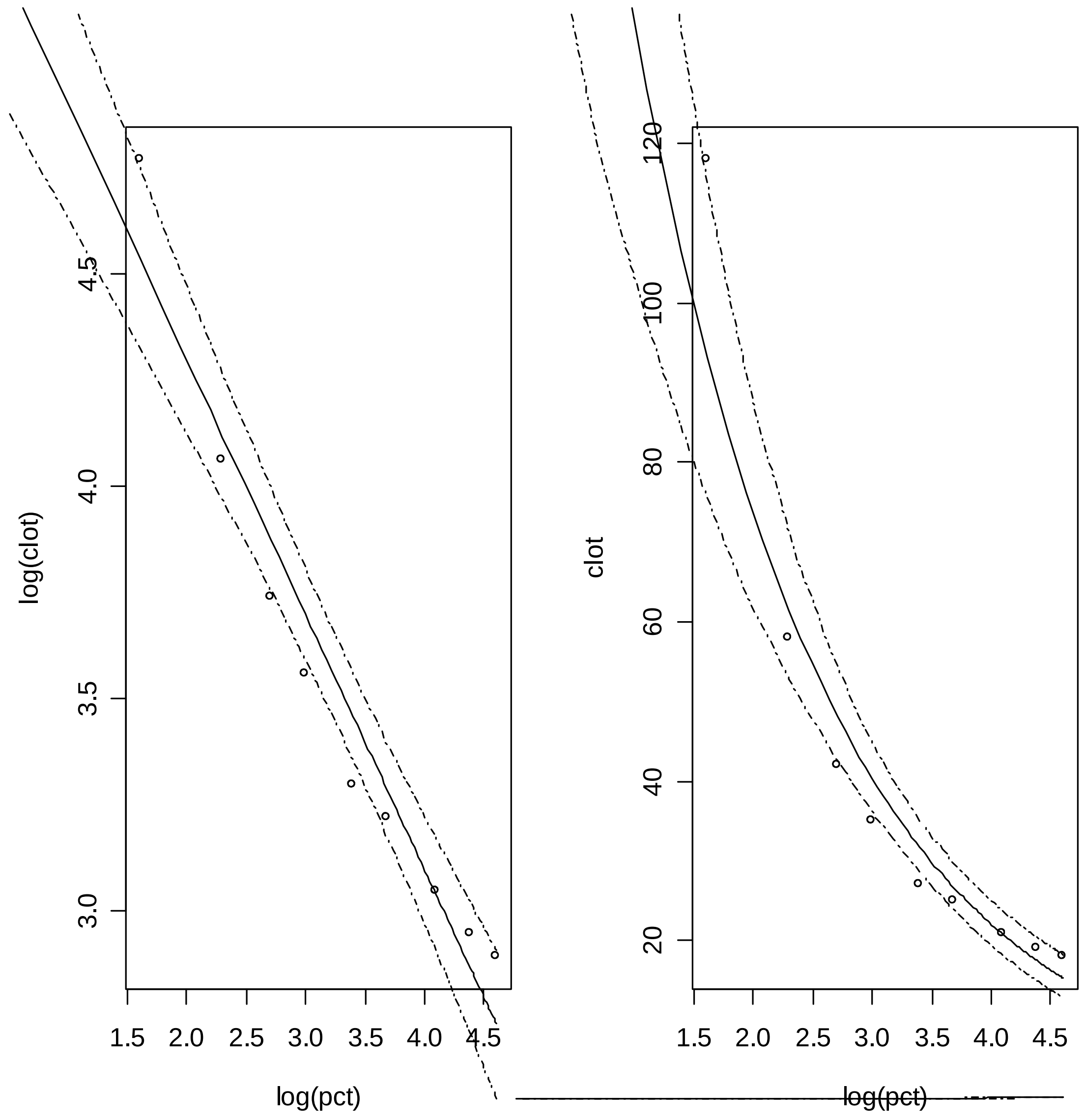
$$\textbf{1/estimated mean} = -0.0165 + 0.0153 \log(\text{pct})$$

Or equivalently

$$\textbf{Estimated mean} = 1/(-0.0165 + 0.0153 \log(\text{pct}))$$



Residuals vs Fitted
Residuals
Predicted values
Normal Q-Q
Std. deviance resid.
Theoretical Quantiles
Scale-Location
√|Std. deviance resid.|
Predicted values
Residuals vs Leverage
Std. deviance resid.
Leverage
Cook's distance


log(clot)
4.5
4.0
3.5
3.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
log(pct)
clot
120
100
80
60
40
20
1.5
2.0
2.5
3.0
3.5
4.0
4.5
log(pct)

<u>Fit Gaussian and Gamma GLMs, each with the 3 available links, and compare</u>
Embedding response variable transformations in a GLM allows for AIC based comparisons across models (realize that this cannot be done with "lm" because you have to physically transform Y…).

We illustrate this idea with the clotting time data:

```
> library(MASS)
### 3 fits, each with different link
> lmod.iv.gam = glm(formula = clot ~ log(pct), family =
                      Gamma(link ="inverse"), data = ct)
> lmod.id.gam = glm(formula = clot ~ log(pct), family =
                      Gamma(link = "identity"), data = ct)
> lmod.lg.gam = glm(formula = clot ~ log(pct), family =
                      Gamma(link = "log"), data = ct)
> lmod.iv.gau = glm(formula = clot ~ log(pct), family =
                      gaussian(link = "inverse"), data = ct)
> lmod.id.gau = glm(formula = clot ~ log(pct), family =
                      gaussian(link = "identity"), data = ct)
> lmod.lg.gau = glm(formula = clot ~ log(pct), family =
                      gaussian(link = "log"), data = ct)

### string all aic's into dataframe along with models & links
> aic=c(mod.iv.gam$aic,mod.id.gam$aic,mod.lg.gam$aic,
               mod.iv.gau$aic, mod.id.gau$aic,mod.lg.gau$aic)
> model=c(rep("gamma",3),rep("normal",3))
> link=c("inv","id","log","inv","id","log")
> model.fits = data.frame(model,link,aic)
> model.fits[order(model.fits$aic),]
   model link      aic
1  gamma  inv 37.98992
4 normal  inv 41.69558
3  gamma  log 58.48166
6 normal  log 61.38859
2  gamma   id 70.43214
5 normal   id 79.51840
```

It seems the gamma(link=inverse) model is the winner, except that from the "glm" help file we read the following for the returned value of "aic":

"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."

I cannot see an easy way to compute the appropriate value of AIC (other than manually coding it), but it should work approximately as a pseudo-AIC. We can however compute the MLE of the dispersion and print the model summary with it to see if it differs a lot from the MME...

```
### MME of dispersion (default)
> summary(mod.iv.gam)

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.0165544  0.0009275  -17.85 4.28e-07 ***
log(pct)     0.0153431  0.0004150   36.98 2.75e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Gamma family taken to be 0.002446059)

    Null deviance: 3.51283  on 8  degrees of freedom
Residual deviance: 0.01673  on 7  degrees of freedom
AIC: 37.99

### MLE of dispersion (but does not re-compute aic...)
> summary(mod.iv.gam, dispersion = gamma.dispersion(mod.iv.gam))

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.0165544  0.0008085  -20.48   <2e-16 ***
log(pct)     0.0153431  0.0003617   42.42   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Gamma family taken to be 0.001858282)

    Null deviance: 3.51283  on 8  degrees of freedom
Residual deviance: 0.01673  on 7  degrees of freedom
AIC: 37.99
```