Generalized Linear Models in R

Andrew McAdam

2019-03-13

Contents

Note

This document is a work in progress. In many places it is not as well annotated as I would like, but it's a start!

Generalized Models

We have spent a fair amount of time going over general linear models. We now want to extend these to consider situations where we have non-normal error distributions. We talked about the concepts behind generalized linear models (GLiMs) in class. Here we will go over how they are implemented and interpreted in R.

There are some additional packages that we will load for GLiMs

```
library (gam)
```

```
## Loading required package: splines
## Loading required package: foreach
## Loaded gam 1.16
```

```
library (MASS)
library (boot)
```

We are going to return to the lizard egg data to start with. I am going to load a new data file.

```r
egg2<-read.table("data/egg.temp2.csv", sep=",", header=T)

summary (egg2)
```

```
##      LayDay          clutch           bin           postlay     
##  Min.   : 75.0   Min.   :1.000   Min.   : 19.0   Min.   :3.250  
##  1st Qu.: 96.0   1st Qu.:1.000   1st Qu.:115.0   1st Qu.:4.430  
##  Median :115.0   Median :2.000   Median :136.0   Median :4.810  
##  Mean   :118.6   Mean   :1.653   Mean   :133.5   Mean   :4.832  
##  3rd Qu.:140.0   3rd Qu.:2.000   3rd Qu.:158.0   3rd Qu.:5.210  
##  Max.   :188.0   Max.   :4.000   Max.   :178.0   Max.   :7.220  
##   clutch.size    eggcode    eggmass       post.eggmass     dam.oby  
##  Min.   :1.000   C:741   Min.   :0.26   Min.   :0.1900   bb:225  
##  1st Qu.:4.000   M:506   1st Qu.:0.40   1st Qu.:0.3500   bo:301  
##  Median :5.000           Median :0.44   Median :0.4100   by:352  
##  Mean   :5.018           Mean   :0.44   Mean   :0.4048   oo: 78  
##  3rd Qu.:6.000           3rd Qu.:0.48   3rd Qu.:0.4500   yo:110  
##  Max.   :9.000           Max.   :0.91   Max.   :0.9100   yy:181  
##      hatch       
##  Min.   :0.0000  
##  1st Qu.:0.0000  
##  Median :1.0000  
##  Mean   :0.7298  
##  3rd Qu.:1.0000  
##  Max.   :1.0000  
```

Meta-Data

These data were from a captive breeding experiemnt I ran with side-blotched lizards (*Uta stansburiana*) while I was a postdoc at UC Santa Cruz with Barry Sinervo. Multiple female lizards and one male were housed within a single holding holding area (bin).

You should get used to writing out similar data descriptions in your own Rmd files. They are very helpful!

- *LayDay* - day of the year (Julian date) on which teh egg was laid.
- *clutch* - whether the clutch was the first, second, third, or fourth clutch of the year.
- *bin* - the housing bin in which the female was raised. There were more than one female per bin.
- *postlay* - mass of female after she finished laying her eggs (g?).
- *clutch.size* - number of eggs in the clutch.
- *eggcode* - whether or not the egg was miniaturized by removing some yolk. (C = control; M = miniaturized).
- *eggmass* - mass of the egg (g) when laid.
- *post.eggmass* - mass of the egg (g) after yolk was removed.
- *dam.oby* - throat colour genotype of the mother.
- *hatch* - whether or not the egg successfully hatched (0 = no; 1 = yes).

This datafile is similar to the egg.temp.csv file that we have worked with in the past, but this one also includes a response variable that indicates whether or not the egg hatched (hatch)

Note that about 73% of the eggs hatched We will need to change "bin" to a factor

```
egg2$bin<-factor(egg2$bin)
```

Specifying a GLiM in R

We will start with a GLiM to predict whether or not an egg will hatch. This is a binary response so we will be dealing with a binomial family. In this case the GLiM is the equivalent of a logistic regression.

```
hatch.glm<-glm(hatch~eggcode+LayDay+post.eggmass, family=binomial, data=egg2)
```

Note that we are using glm instead of lm and we therefore need to specify the family type. We did not specify a link function so the default "link=logit" is used.

```
summary (hatch.glm)
```

```
##
## Call:
## glm(formula = hatch ~ eggcode + LayDay + post.eggmass, family = binomial,
##     data = egg2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3011  -1.1999   0.6390   0.8161   1.3262
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.847376   0.507975  -3.637 0.000276 ***
## eggcodeM     -0.509826   0.150804  -3.381 0.000723 ***
## LayDay        0.012742   0.002738   4.654 3.25e-06 ***
## post.eggmass  3.992286   1.105930   3.610 0.000306 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1455.3  on 1246  degrees of freedom
## Residual deviance: 1375.4  on 1243  degrees of freedom
## AIC: 1383.4
##
## Number of Fisher Scoring iterations: 4
```

Note that this is very similar to what we are used to seeing. Instead of dealing with variance though we are dealing now with DEVIANCE. Deviance is defined to be twice the difference between the log-likilihood of the best possible model and the log-likelihood of the current model. The log-liklihood of the best possible model is based on the variance in yoru response variable.

Also not that we are no longer testing the significance of our parameters based on a t test statistic. Instead we are now using a Z test statistic. This test statistic does not have an associated degrees of freedom. So when you report the significance of a parameter in a GLiM you woudl report b+/-SE, Z and P (no df needed).

The proportion of deviance explained is

```
1-1375.4/1455.3
```

```
## [1] 0.05490277
```

This is the summary command but we can also test the significance of terms rather than parameters using the anova command.

```r
anova (hatch.glm, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: hatch
##
## Terms added sequentially (first to last)
##
##
##              Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                          1246     1455.3
## eggcode       1   32.635      1245     1422.6 1.112e-08 ***
## LayDay        1   33.547      1244     1389.1 6.955e-09 ***
## post.eggmass  1   13.635      1243     1375.5  0.000222 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that in this case the terms are assessed from first to last. The model is assessed in a "Type I" style where the order matters. This analysis is basically a series of likelihood ratio tests, comparing progressively more complex models.

Testing Assumptions in GLiMs

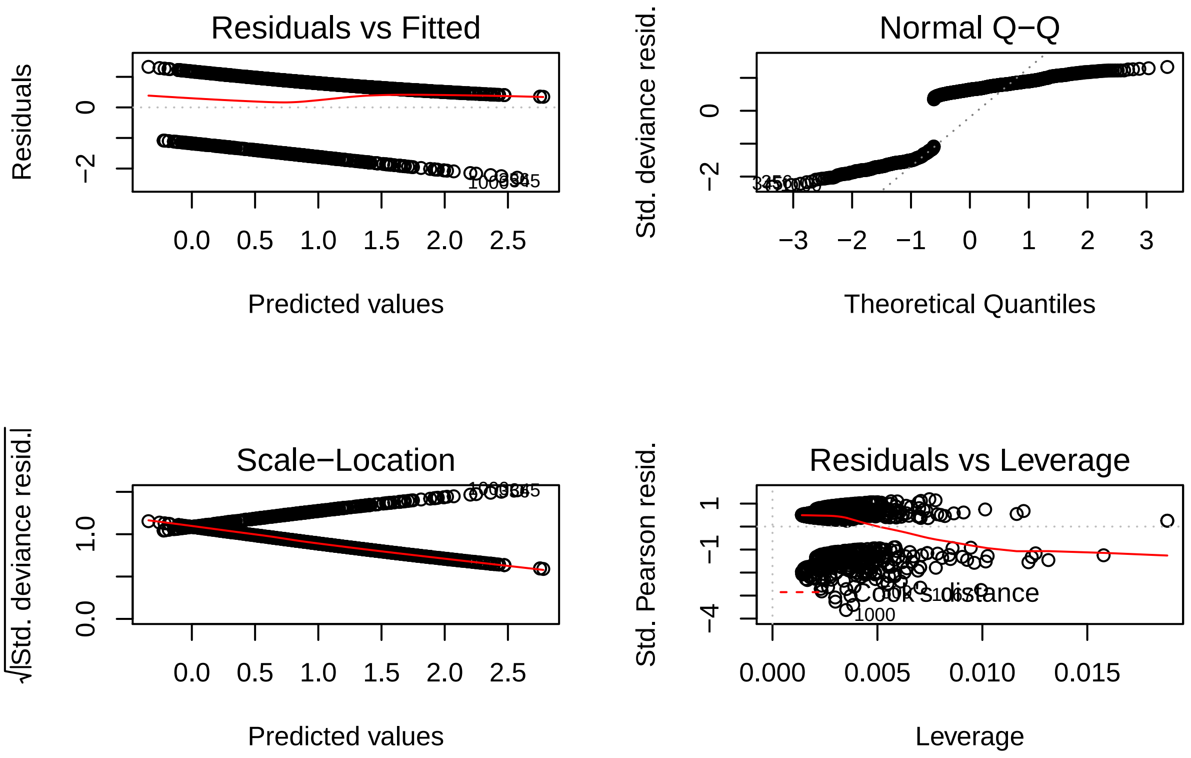
Many of the assumptions of a GLiM are similar to teh assumptions of a glm. These include:

- Independence of observations - good sampling
- Correct specification of the variance function
- Correct specification of the link function
- Correct form of the explanatory variables - look for nonlinearities
- Lack of undue influence of individual observations – Cook's distance
- Correct specification of the dispersion parameter

Diagnostic Plots

Regular diagnostic plots don't help us very much for GLiMs.

```r
par (mfrow=c(2,2))
plot (hatch.glm)
```



Note that the residuals are never going to look very good in a binomial glm. We are used to using these plots to test for homogeneity in the residulas, but in a GLiM the residuals are never going to appear homogeneous, because the response variable is often very discontinuous. Nevertheless, we can still use these diagnostics to look at points with high leverage and high residuals (the last plot - Cook's distances).

I recently became aware of a package for assessing diagnostic plots for GLiMs. More information can be found at: https://theoreticalecology.wordpress.com/2016/08/28/dharma-an-r-package-for-residual-diagnostics-of-glmms/

Here is the Vignette: https://cran.r-project.org/web/packages/DHARMa/vignettes/DHARMa.html

I will give it a try here.

```
#install.packages("DHARMa")
library (DHARMa)
```

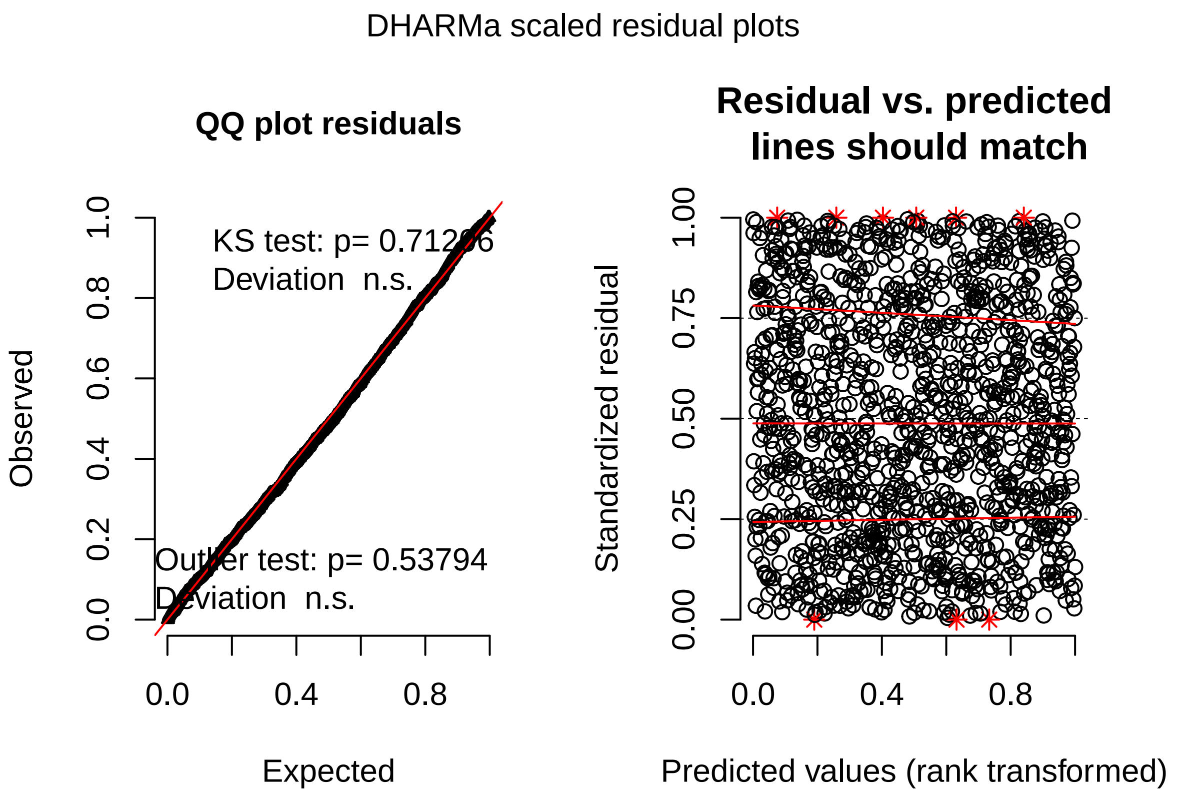
```
## Warning: package 'DHARMa' was built under R version 3.5.2
```

```
simulation_output_hatch.glm<-simulateResiduals(fittedModel = hatch.glm, n = 250)
```

Here we now expect a uniform distribution (flat distribution) of simulated residuals.

We can visualize these simulated residuals using

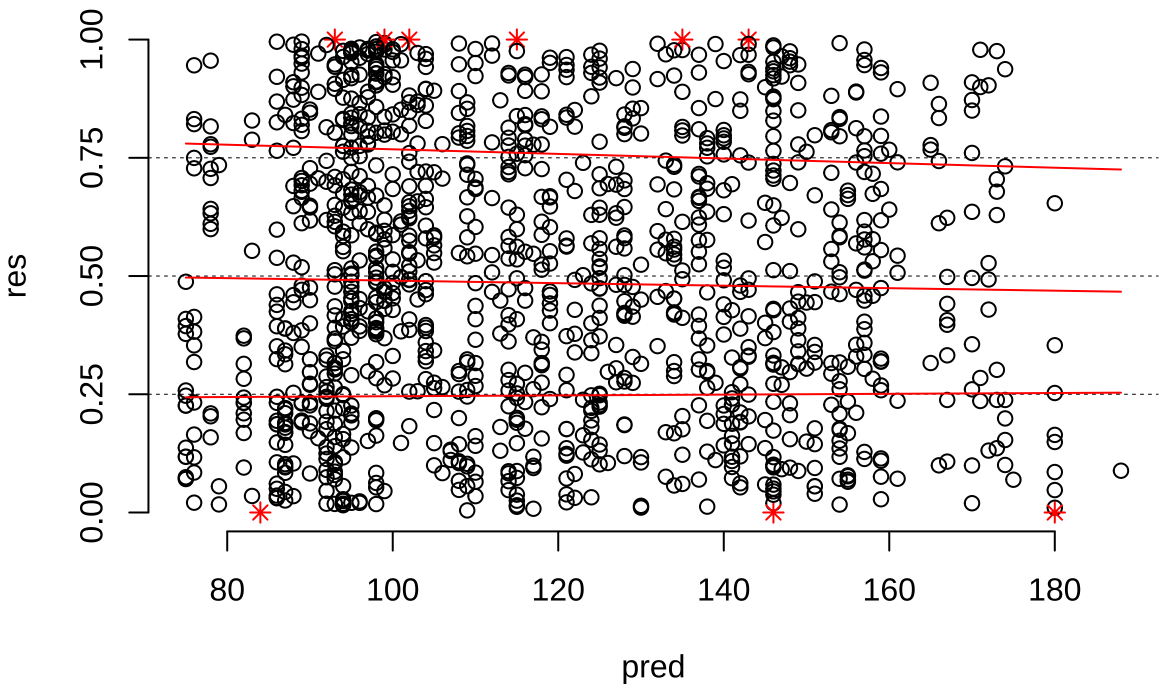
```
plot(simulation_output_hatch.glm)
```



So the above plot shows no deviation from the expected distribution. In the plot on the right, 'outliers' are plotted as red stars. The red lines in the plot on the right should be straight accross and at the values on the y axis of 0.25, 0.5 and 0.75. These look very good above!!

We should also test whether there is a relationship between the residuals and our specific predictors

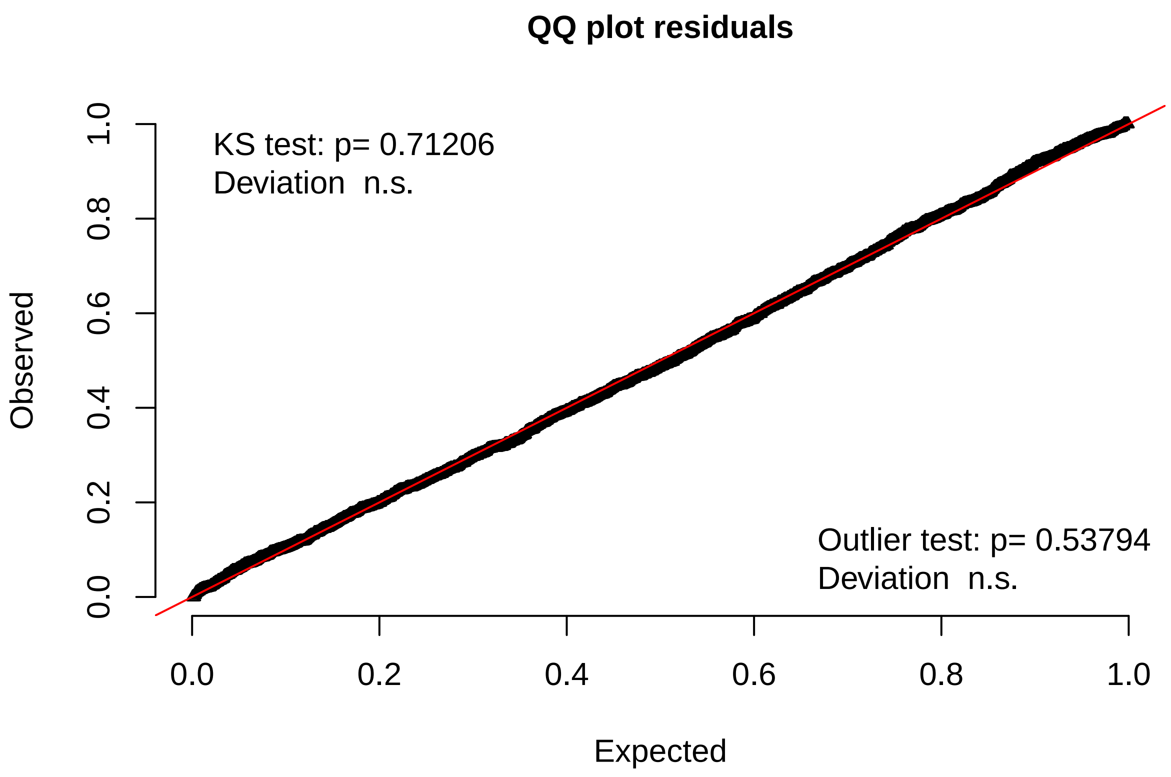
```r
plotResiduals(egg2$LayDay, simulation_output_hatch.glm$scaledResiduals)
```



This also looks very good.

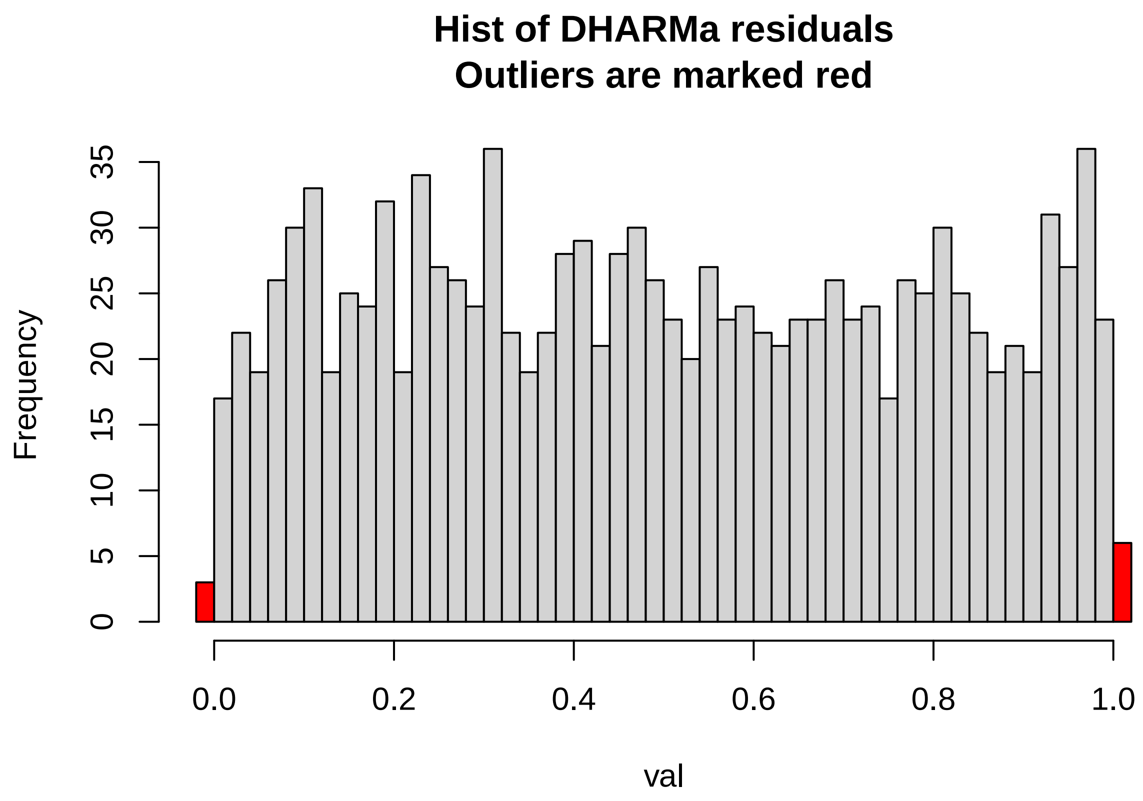
This package also runs some specific tests, which seem very nice!

```
testUniformity(simulation_output_hatch.glm)
```



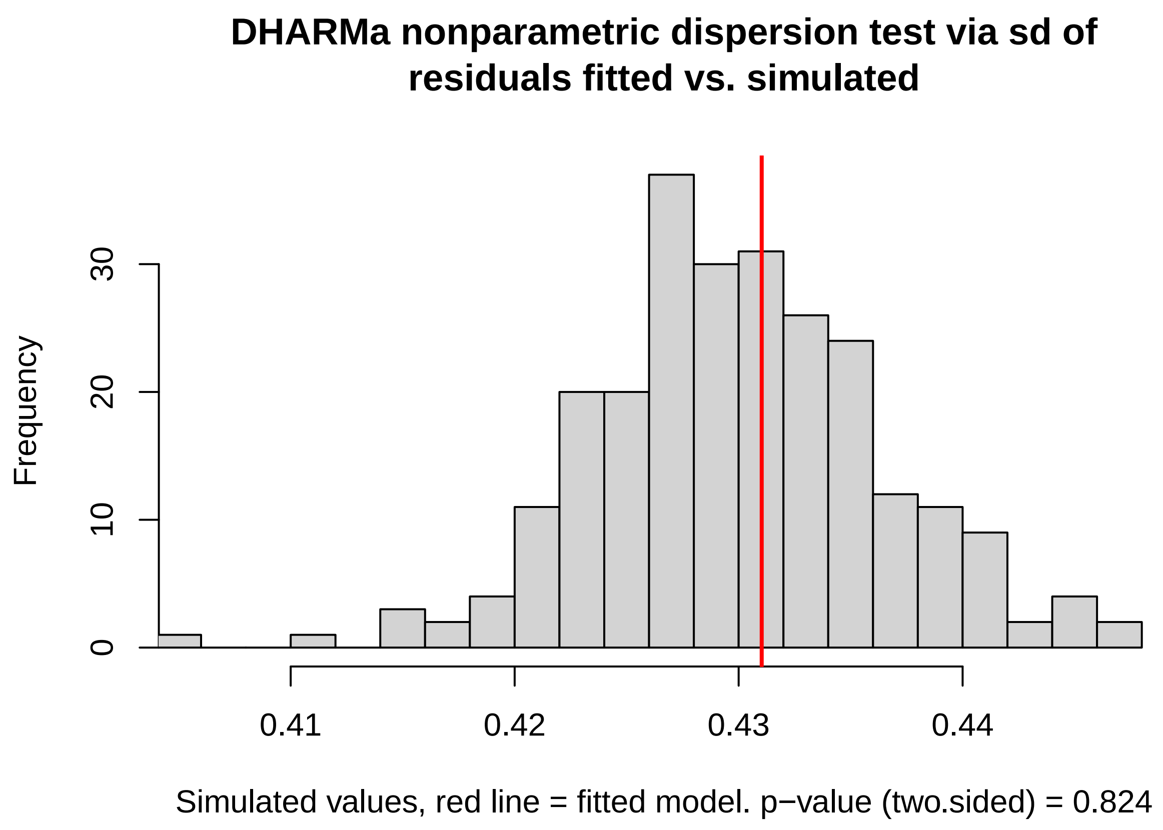
```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  simulationOutput$scaledResiduals
## D = 0.019809, p-value = 0.7121
## alternative hypothesis: two-sided
```

```
testOutliers(simulation_output_hatch.glm)
```



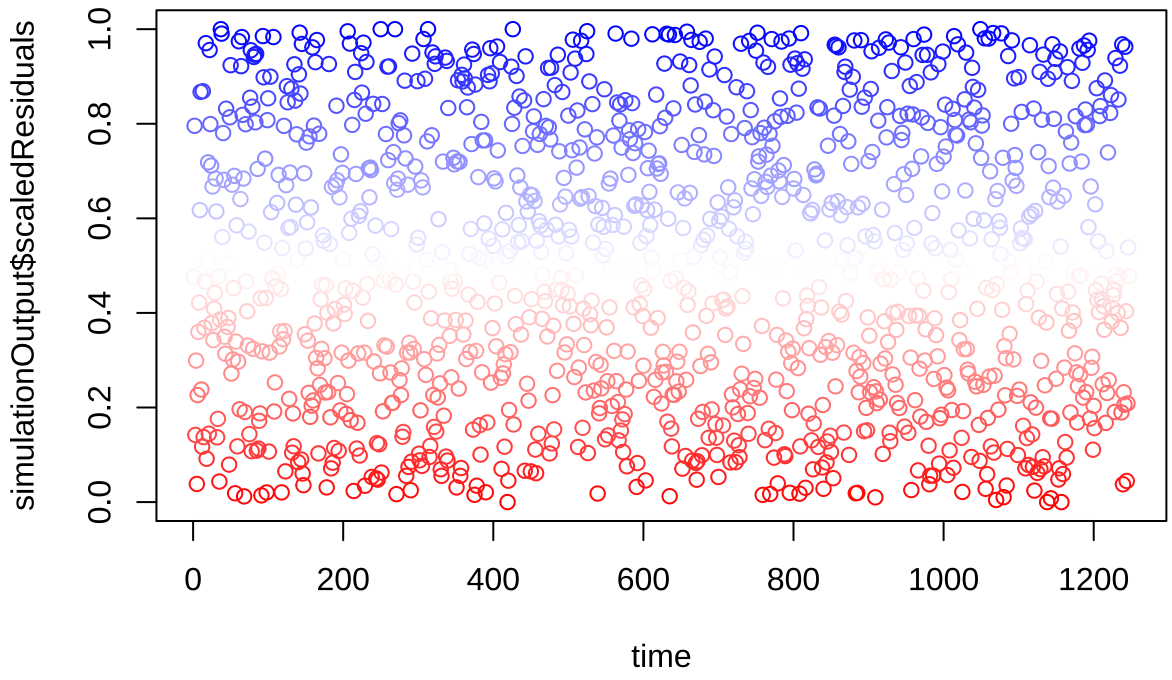
```
##
##  DHARMa outlier test based on exact binomial test
##
## data:  simulation_output_hatch.glm
## outLow = 3.0000e+00, outHigh = 6.0000e+00, nobs = 1.2470e+03,
## freqH0 = 3.9841e-03, p-value = 0.5379
## alternative hypothesis: two.sided
```

```
testDispersion(simulation_output_hatch.glm)
```



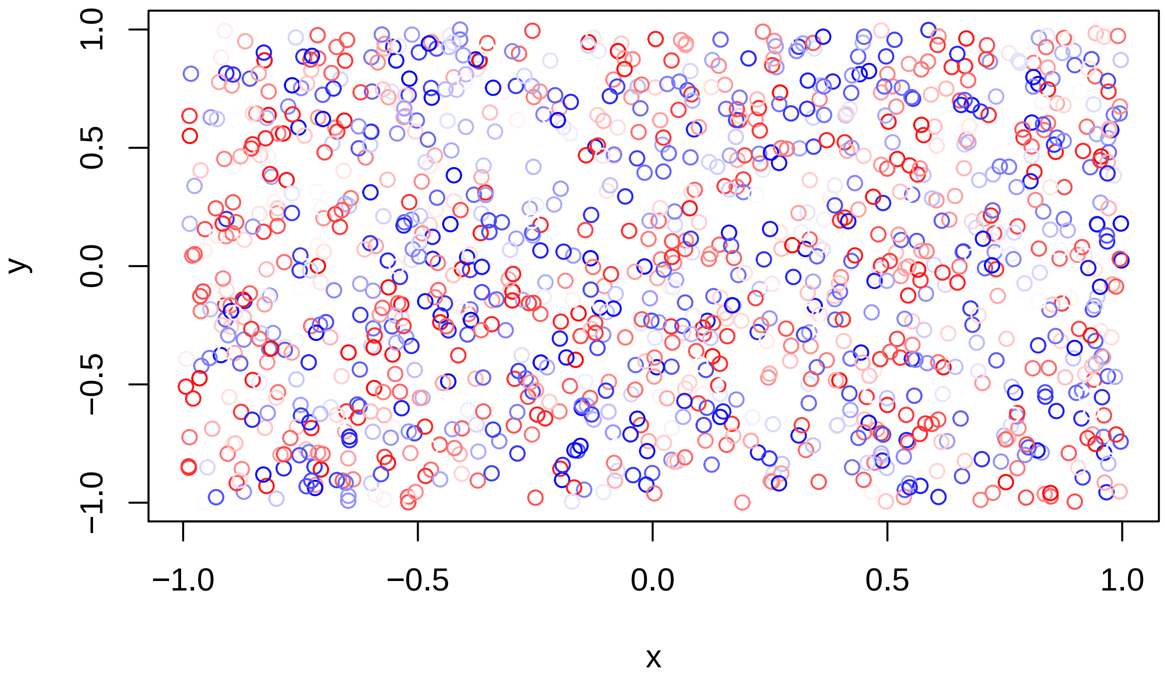
```
##
##  DHARMa nonparametric dispersion test via sd of residuals fitted
##  vs. simulated
##
## data:  simulationOutput
## ratioObsSim = 1.0023, p-value = 0.824
## alternative hypothesis: two.sided
```

```
#testZeroinflation(simulation_output_hatch.glm)
#testGeneric(simulation_output_hatch.glm)
testTemporalAutocorrelation(simulation_output_hatch.glm)
```



```
##
##  Durbin-Watson test
##
## data:  simulationOutput$scaledResiduals ~ 1
## DW = 2.0917, p-value = 0.105
## alternative hypothesis: true autocorrelation is not 0
```

```
testSpatialAutocorrelation(simulation_output_hatch.glm)
```



```
##
##  DHARMa Moran's I test for spatial autocorrelation
##
```

```
## data:  simulation_output_hatch.glm
## observed = 0.00273010, expected = -0.00080257, sd = 0.00208520,
## p-value = 0.09023
## alternative hypothesis: Spatial autocorrelation
```

Over-Dispersion

One important test above is the test for overdispersion. In a GLiM the error variance is not estimated. Instead it is fixed at some value based on the variance function that is determined by the assumed distribution. If your data don't fit this assumed distribution well then the modelk could be over-dispersed (too much unexplained variation) or under-dispersed (too little variation). Overdispersion can cause your inferences to be overly liberal (too likely to reject the null; p-values are too small) so it is of concern. There are several possible causes of over-dispersion including:

- unexplained variation - try including an additional predictor to account for this variation. Sometimes, however, there is nothing you can do.
- structure in the data - if there is a lack of independence in the data this can lead to overdispersion. Fitting a random effect (see the section on mixed effect models) can sometimes account for this.
- a mis-specified model - perhaps your assumed distriubution is not a good one.

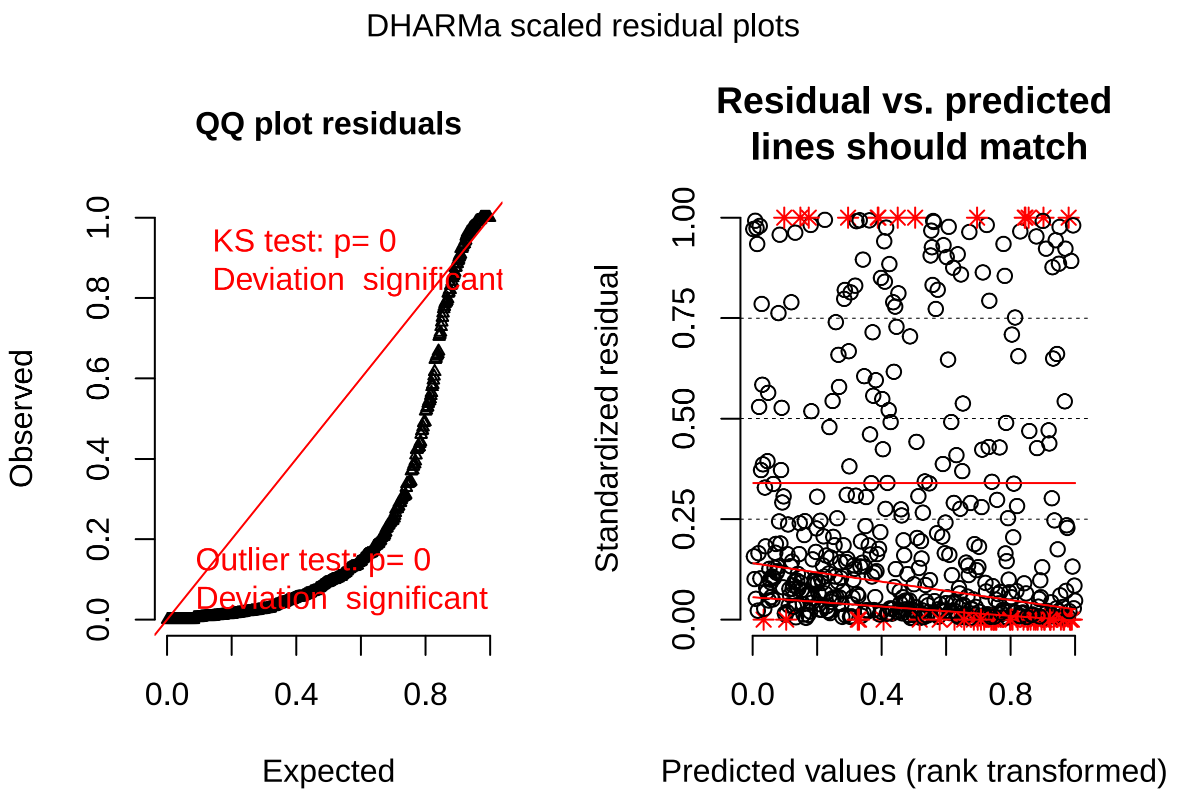
The DHARMa package provides an example of what overdispersion would look like.

```r
library (lme4)
```

```
## Loading required package: Matrix
```

```r
testData = createData(sampleSize = 500, overdispersion = 2, family = poisson())
fittedModel <- glmer(observedResponse ~ Environment1 + (1|group) , family = "poisson", data = testData)

simulationOutput <- simulateResiduals(fittedModel = fittedModel)
plot(simulationOutput)
```



Note that instead of a uniform distribution of simulated residuals they are clustered around zero.

Here is an example of under-dispersion where we have too few residuals around the tail - too many around the value of 0.5

```
testData = createData(sampleSize = 500, intercept=0, fixedEffects = 2, overdispersion = 0, family = poi
fittedModel <- glmer(observedResponse ~ Environment1 + (1|group) , family = "poisson", data = testData)
```

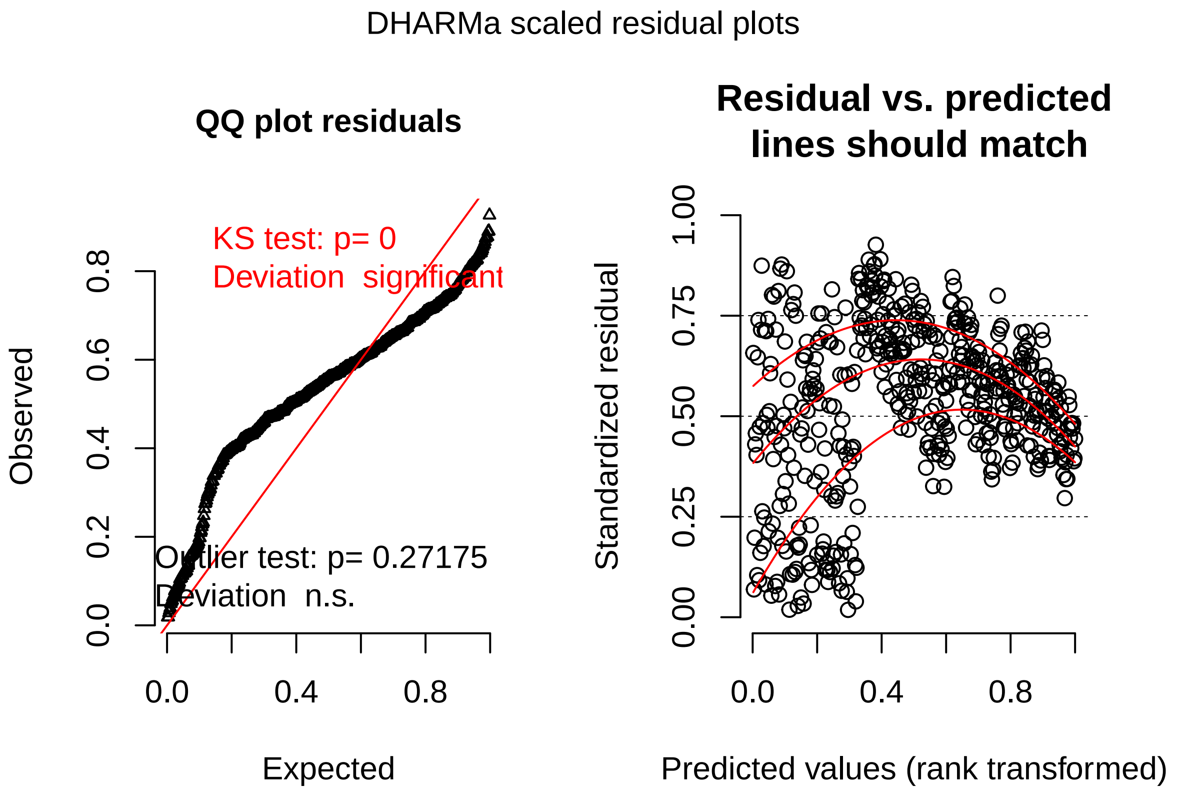
```
## singular fit
```

```
summary(fittedModel)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: observedResponse ~ Environment1 + (1 | group)
##    Data: testData
##
##      AIC      BIC   logLik deviance df.resid
##    990.6   1003.3   -492.3    984.6      497
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -0.60655 -0.36415 -0.08241  0.20780  1.01379
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  group  (Intercept) 0        0
## Number of obs: 500, groups:  group, 10
##
## Fixed effects:
```

```
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.19838    0.06059  -3.274  0.00106 **
## Environment1  2.29112    0.08876  25.813  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## Environmnt1 -0.823
## convergence code: 0
## singular fit
```

```
simulationOutput <- simulateResiduals(fittedModel = fittedModel)
plot(simulationOutput)
```



As far as I can tell this DHARMa package is great for testing these assumptions.

There are, however, some simpler ways to test for over-dispersion. As a start, we can estimate the amount of dispersion in a model by comparing the resodual df to the residual deviance in a GLiM. We expect that these ought to be similar. NOTE that this is just an estimate.

```
 summary (hatch.glm)
```

```
##
## Call:
## glm(formula = hatch ~ eggcode + LayDay + post.eggmass, family = binomial,
##     data = egg2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3011  -1.1999   0.6390   0.8161   1.3262
```

```
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.847376   0.507975  -3.637 0.000276 ***
## eggcodeM     -0.509826   0.150804  -3.381 0.000723 ***
## LayDay        0.012742   0.002738   4.654 3.25e-06 ***
## post.eggmass  3.992286   1.105930   3.610 0.000306 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1455.3  on 1246  degrees of freedom
## Residual deviance: 1375.4  on 1243  degrees of freedom
## AIC: 1383.4
##
## Number of Fisher Scoring iterations: 4
```

Note the similarity between the residual deviance and the residual df.

When you are looking at the relationship between the residual deviance and the residual df that this is only a reliable measure of overdispersion if n*p (for binomial) is large

```r
length(egg2$hatch)*mean(egg2$hatch)
```

```
## [1] 910
```

This is large

We could still get a measure of the dispersion parameter by using the quasi family

```r
hatch.glm4<-glm(hatch~eggcode+LayDay+I(LayDay^2)+post.eggmass, family=quasibinomial, data=egg2, na.acti

summary (hatch.glm4)
```

```
##
## Call:
## glm(formula = hatch ~ eggcode + LayDay + I(LayDay^2) + post.eggmass,
##     family = quasibinomial, data = egg2, na.action = na.omit)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2910  -1.0873   0.5929   0.7952   1.5763
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -9.982e+00  1.513e+00  -6.597 6.18e-11 ***
## eggcodeM     -5.332e-01  1.524e-01  -3.498 0.000485 ***
## LayDay        1.528e-01  2.452e-02   6.231 6.33e-10 ***
## I(LayDay^2)  -5.717e-04  9.882e-05  -5.785 9.15e-09 ***
## post.eggmass  3.945e+00  1.096e+00   3.600 0.000331 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasibinomial family taken to be 0.9952764)
##
##     Null deviance: 1455.3  on 1246  degrees of freedom
## Residual deviance: 1343.0  on 1242  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 4
```

Note that now the dispersion parameter is estimated rather than assuming that it is 1. Notice that the estimated parameter is very close to one.

What would a very poorly fit model look like (i.e. the wrong link)?

```
hatch.poor<-glm(post.eggmass~eggcode+I(LayDay^2), family=quasibinomial, data=egg2, na.action=na.omit)
```

In this case the response variable is normally distributed but we have specified the quasibinomial family

```
summary (hatch.poor)
```

```
##
## Call:
## glm(formula = post.eggmass ~ eggcode + I(LayDay^2), family = quasibinomial,
##     data = egg2, na.action = na.omit)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.42469  -0.08084  -0.00374   0.07454   1.05018
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.102e-01  1.952e-02 -21.015   <2e-16 ***
## eggcodeM    -2.967e-01  1.538e-02 -19.292   <2e-16 ***
## I(LayDay^2)  9.693e-06  1.163e-06   8.333   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasibinomial family taken to be 0.01680608)
##
##     Null deviance: 28.533  on 1246  degrees of freedom
## Residual deviance: 21.242  on 1244  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 3
```

Notice that the model is highly under-dispersed. This is evident from the residual deviance and df as well as the estimated dispersion parameter.

Nonlinearities

We should also look for nonlinearities

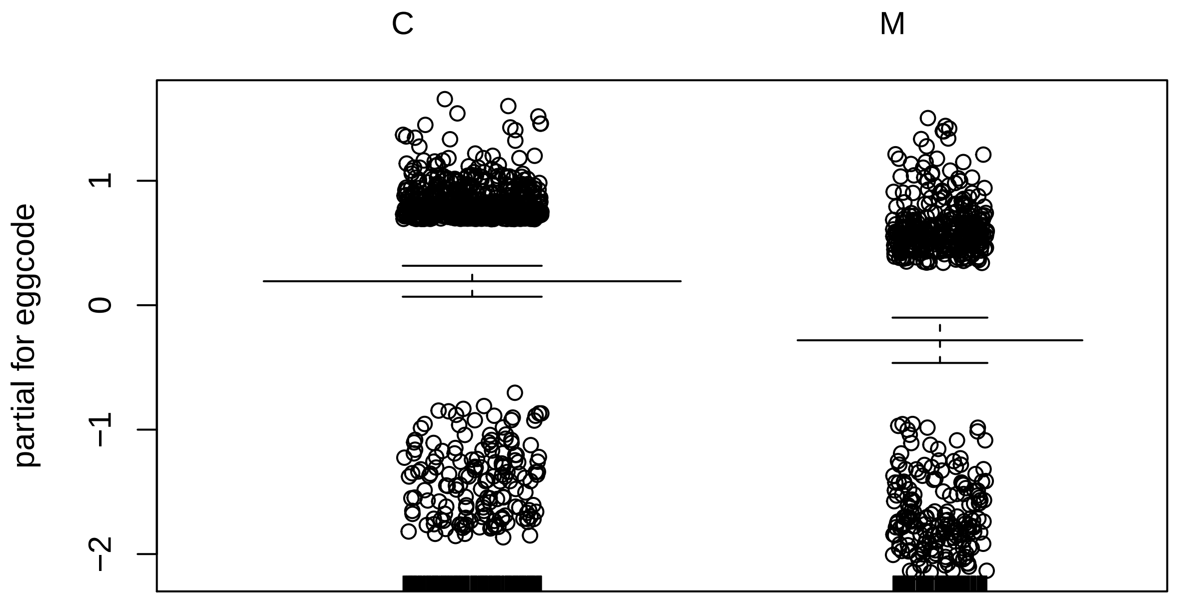
```
hatch.gam<-gam(hatch~eggcode+s(LayDay)+s(post.eggmass), family=binomial, data=egg2, na.action=na.omit)
anova (hatch.gam)
```

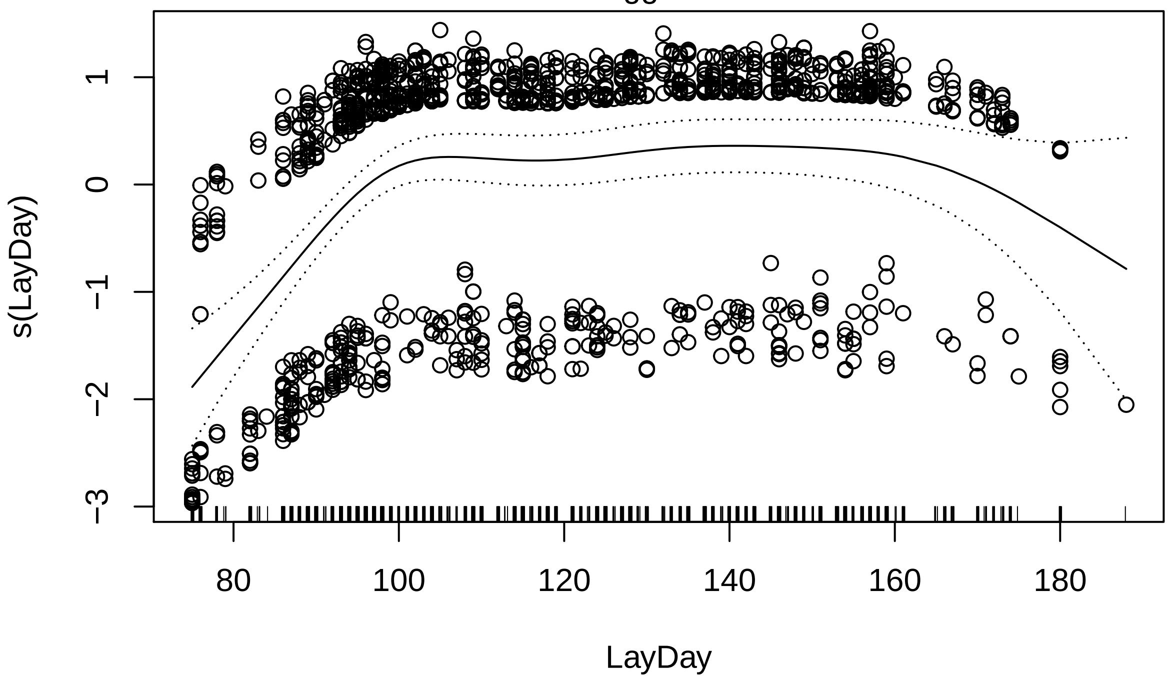
```
## Anova for Nonparametric Effects
##                 Npar Df Npar Chisq    P(Chi)
## (Intercept)
## eggcode
## s(LayDay)             3     54.948 7.045e-12 ***
## s(post.eggmass)       3      8.483     0.037 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

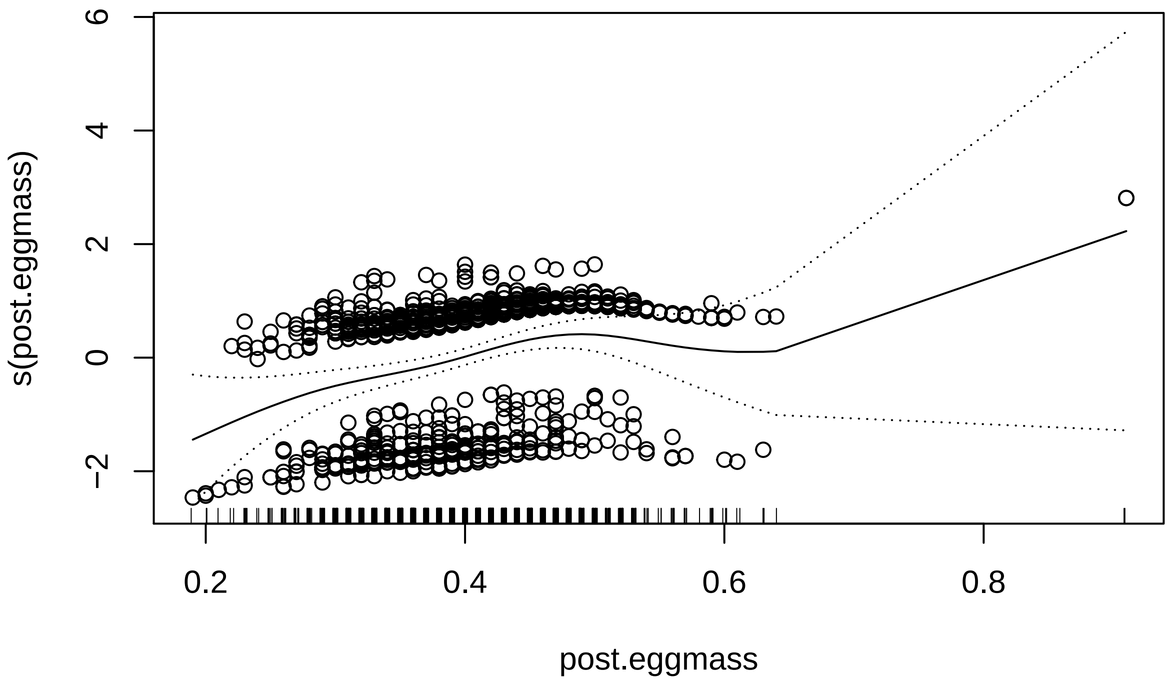
There seems to be strong nonlinear effects for layDay and maybe post-egg mass. We will try and correct LayDay first and then see if that helps the relationship for posteggmass. Let's see what the relationship looks like.

The plot.Gam function is good for plotting generalized additive models. The value on the y axis is the response variable corrected for other terms in the model (this is a partial plot).

```r
plot.Gam(hatch.gam, se=T, residuals = T)
```







Note that we seem to have some sort of quadratic effect of LayDate and maybe an outlier for post.eggmass. I would normally go back and double-check that value but I remember that egg. It is not a mistake. We might want to try and run the model without this point just to make sure that our conclusions don't depend on that data point. The Cook's distance values suggest that the results do not, but we could check.

So let's fit a quadratic effect for LayDay

```
hatch.glm2<-glm(hatch~eggcode+LayDay+I(LayDay^2)+post.eggmass, family=binomial, data=egg2, na.action=na
summary (hatch.glm2)
```

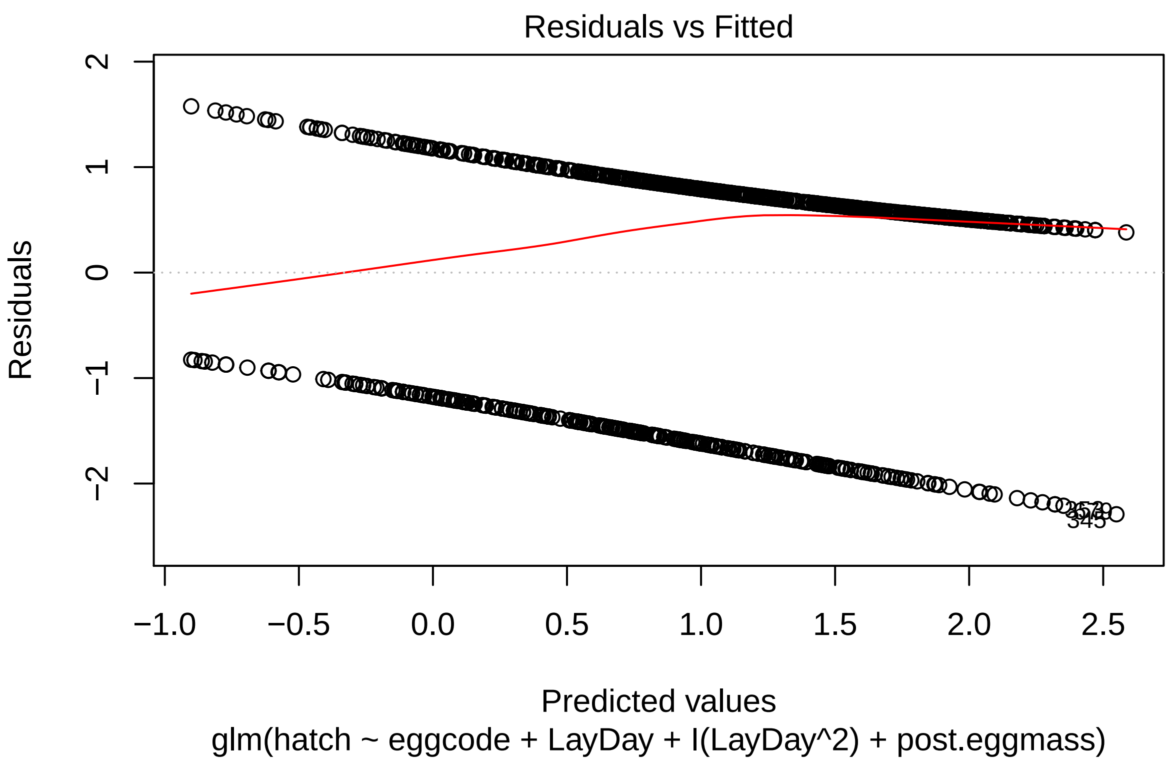
```
##
## Call:
## glm(formula = hatch ~ eggcode + LayDay + I(LayDay^2) + post.eggmass,
##     family = binomial, data = egg2, na.action = na.omit)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2910  -1.0873   0.5929   0.7952   1.5763
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -9.982e+00  1.517e+00  -6.582 4.65e-11 ***
## eggcodeM     -5.332e-01  1.528e-01  -3.490 0.000483 ***
## LayDay        1.528e-01  2.458e-02   6.216 5.09e-10 ***
## I(LayDay^2)  -5.717e-04  9.905e-05  -5.772 7.85e-09 ***
## post.eggmass  3.945e+00  1.098e+00   3.591 0.000329 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1455.3  on 1246  degrees of freedom
## Residual deviance: 1343.0  on 1242  degrees of freedom
## AIC: 1353
##
## Number of Fisher Scoring iterations: 4
```

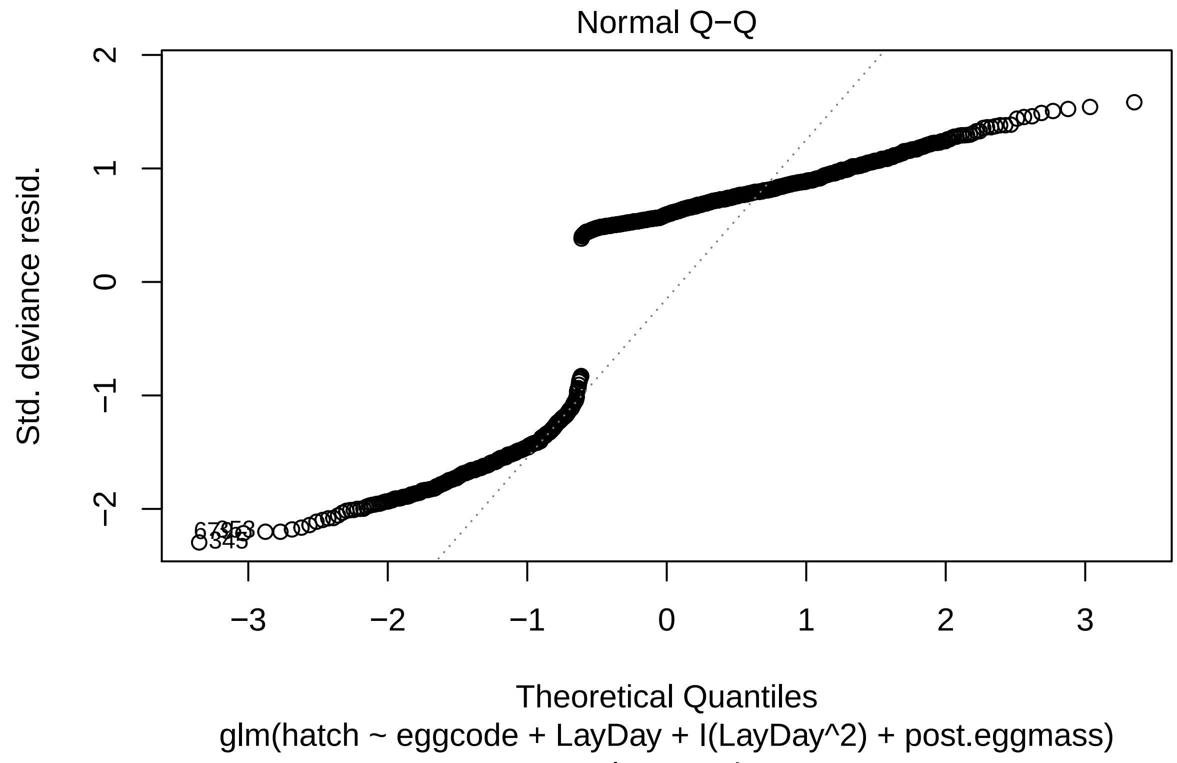
Do any non-linearities remain after we consider the quadratic effect?

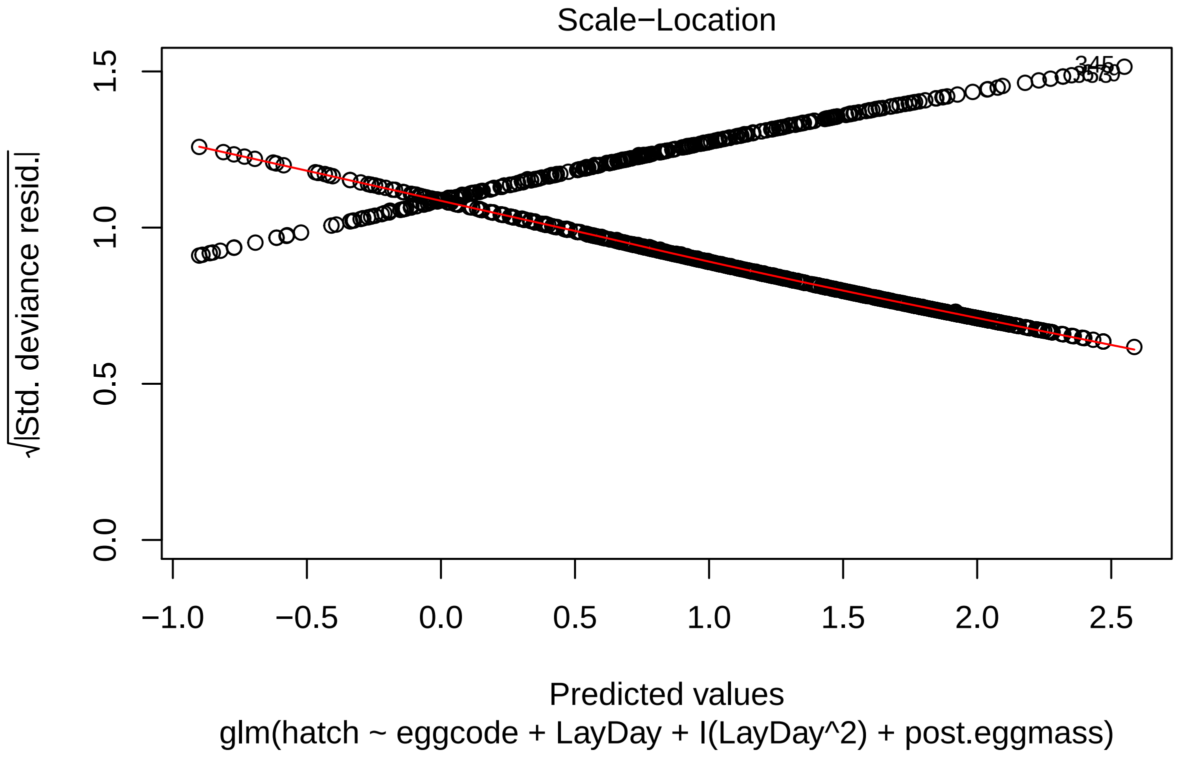
```
hatch.gam2<-gam(hatch~eggcode+s(LayDay)+I(LayDay^2)+s(post.eggmass), family=binomial, data=egg2, na.acti
anova (hatch.gam2)
```

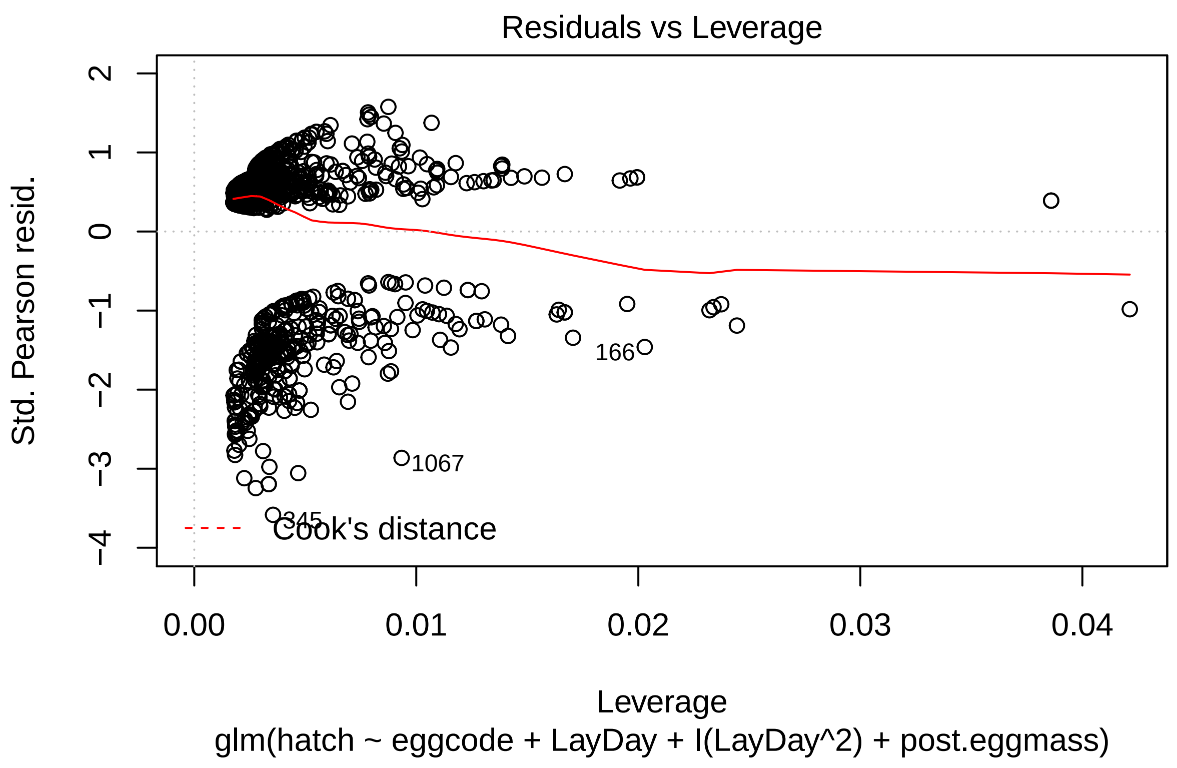
```
## Anova for Nonparametric Effects
##                   Npar Df Npar Chisq    P(Chi)
## (Intercept)
## eggcode
## s(LayDay)               3    24.4535 2.009e-05 ***
## I(LayDay^2)
## s(post.eggmass)         3     8.8104   0.03192 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(hatch.glm2)
```




Normal Q−Q
Std. deviance resid.
Theoretical Quantiles
glm(hatch ~ eggcode + LayDay + I(LayDay^2) + post.eggmass)
6795
345
2
1
0
−1
−2
−3
−2
−1
0
1
2
3


Scale−Location
√|Std. deviance resid.|
Predicted values
glm(hatch ~ eggcode + LayDay + I(LayDay^2) + post.eggmass)
345
1.5
1.0
0.5
0.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5



It seems like the nonlinearity for posteggmass is driven by that one point. We will exclude it and rerun the model.

```
hatch.glm3<-glm(hatch~eggcode+LayDay+I(LayDay^2)+post.eggmass, family=binomial, data=egg2, na.action=na
```

We can compare the results of the two models

```
summary (hatch.glm2)
```

```
##
## Call:
## glm(formula = hatch ~ eggcode + LayDay + I(LayDay^2) + post.eggmass,
##     family = binomial, data = egg2, na.action = na.omit)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2910  -1.0873   0.5929   0.7952   1.5763
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -9.982e+00  1.517e+00  -6.582 4.65e-11 ***
## eggcodeM     -5.332e-01  1.528e-01  -3.490 0.000483 ***
## LayDay        1.528e-01  2.458e-02   6.216 5.09e-10 ***
## I(LayDay^2)  -5.717e-04  9.905e-05  -5.772 7.85e-09 ***
## post.eggmass  3.945e+00  1.098e+00   3.591 0.000329 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1455.3  on 1246  degrees of freedom
## Residual deviance: 1343.0  on 1242  degrees of freedom
## AIC: 1353
##
## Number of Fisher Scoring iterations: 4
```

```r
summary (hatch.glm3)
```

```
##
## Call:
## glm(formula = hatch ~ eggcode + LayDay + I(LayDay^2) + post.eggmass,
##     family = binomial, data = egg2, subset = -1067, na.action = na.omit)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3133  -1.0799   0.5908   0.7950   1.5780
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.014e+01  1.521e+00  -6.665 2.65e-11 ***
## eggcodeM     -5.073e-01  1.533e-01  -3.309 0.000936 ***
## LayDay        1.534e-01  2.463e-02   6.230 4.68e-10 ***
## I(LayDay^2)  -5.741e-04  9.924e-05  -5.785 7.27e-09 ***
## post.eggmass  4.227e+00  1.110e+00   3.809 0.000140 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1452.6  on 1245  degrees of freedom
## Residual deviance: 1338.5  on 1241  degrees of freedom
## AIC: 1348.5
##
## Number of Fisher Scoring iterations: 4
```

There is very little change in the parameters of the model by excluding that point.

Probit Model

Recall our original model:

```r
hatch.glm<-glm(hatch~eggcode+LayDay+post.eggmass, data=egg2, family=binomial)
```

In this case we are using the default link function for the binomial family. This is the "logit" function which converts probabilities into log odds = log(prop/(1-prop)). An equivalent model would be:

```r
hatch.glm<-glm(hatch~eggcode+LayDay+post.eggmass, data=egg2, family=binomial(logit))
```

We could also specify

```r
hatch.glm<-glm(hatch~eggcode+LayDay+post.eggmass, data=egg2, family=binomial(probit))
```

I am less familiar with the probit link but apparently these two links will often give similar results except when there are many very high or very low probabilities in which case probit is preferred.

```r
hatch.glm<-glm(hatch~eggcode+LayDay+post.eggmass, data=egg2, family=binomial(logit))
summary (hatch.glm)
```

```
##
## Call:
## glm(formula = hatch ~ eggcode + LayDay + post.eggmass, family = binomial(logit),
##     data = egg2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3011  -1.1999   0.6390   0.8161   1.3262
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.847376   0.507975  -3.637 0.000276 ***
## eggcodeM     -0.509826   0.150804  -3.381 0.000723 ***
## LayDay        0.012742   0.002738   4.654 3.25e-06 ***
## post.eggmass  3.992286   1.105930   3.610 0.000306 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1455.3  on 1246  degrees of freedom
## Residual deviance: 1375.4  on 1243  degrees of freedom
## AIC: 1383.4
##
## Number of Fisher Scoring iterations: 4
```

```
anova (hatch.glm, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: hatch
##
## Terms added sequentially (first to last)
##
##
##              Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                          1246     1455.3
## eggcode       1   32.635      1245     1422.6 1.112e-08 ***
## LayDay        1   33.547      1244     1389.1 6.955e-09 ***
## post.eggmass  1   13.635      1243     1375.5  0.000222 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that this is the same approach as we used for an lm. The only difference is that now we are dealing with an F test. Note that the anova output assesses (in sequence) whether the addition of the next term significantly improves the fit of the model relative to the simpler model in which that term is not yet included. We can show the sequence of these tests as:

```
temp1<-glm(hatch~1, data=egg2, family=binomial(logit))
temp2<-glm(hatch~eggcode, data=egg2, family=binomial(logit))
temp3<-glm(hatch~eggcode+LayDay, data=egg2, family=binomial(logit))
temp4<-glm(hatch~eggcode+LayDay+post.eggmass, data=egg2, family=binomial(logit))

anova (hatch.glm, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: hatch
##
## Terms added sequentially (first to last)
##
##
##              Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                         1246     1455.3
## eggcode       1   32.635      1245     1422.6 1.112e-08 ***
## LayDay        1   33.547      1244     1389.1 6.955e-09 ***
## post.eggmass  1   13.635      1243     1375.5  0.000222 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova (temp1, temp2, temp3, temp4, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model 1: hatch ~ 1
## Model 2: hatch ~ eggcode
## Model 3: hatch ~ eggcode + LayDay
## Model 4: hatch ~ eggcode + LayDay + post.eggmass
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      1246     1455.3
## 2      1245     1422.6  1   32.635 1.112e-08 ***
## 3      1244     1389.1  1   33.547 6.955e-09 ***
## 4      1243     1375.5  1   13.635  0.000222 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We wouldn't normally do this. Normally you would use the anova output to assess the significance of the terms in your model with the understanding that this is based on a Type Iish philosophy (i.e. order matters)

Note the reminder that terms are ordered from first to last.

Interpreting parameters in a GLiM

We will start simple

```
hatch.simple<-glm(hatch~post.eggmass, family=binomial, data=egg2)

summary (hatch.simple)
```

```
##
## Call:
## glm(formula = hatch ~ post.eggmass, family = binomial, data = egg2)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.279  -1.304   0.682   0.821   1.248
```

```
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -1.6059     0.3808  -4.217 2.48e-05 ***
## post.eggmass    6.5468     0.9623   6.803 1.02e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1455.3  on 1246  degrees of freedom
## Residual deviance: 1405.1  on 1245  degrees of freedom
## AIC: 1409.1
##
## Number of Fisher Scoring iterations: 4
```

When you get a predicted value remember that it is on the log-odds scale because of the logit link function.

So if we wanted to calculate the predicted probability of survival for an egg that was 0.2g we would first get the predicted value on the log-odds scale.

```
-1.6059+6.5468*0.2
```

```
## [1] -0.29654
```

This is on the log-odds scale. We can then use a function in the boot package to back-transform this value. Note that the two colons just specifies that the function inv.logit is located within the boot package. This notation is not needed, but it can help clarify situations where there are two functions in different packages with the same name. It also helps to make it more clear to you that this is why I needed to load teh boot package at the start of this document.

```
boot::inv.logit(-0.29654)
```

```
## [1] 0.4264035
```

We can check this against our calculation

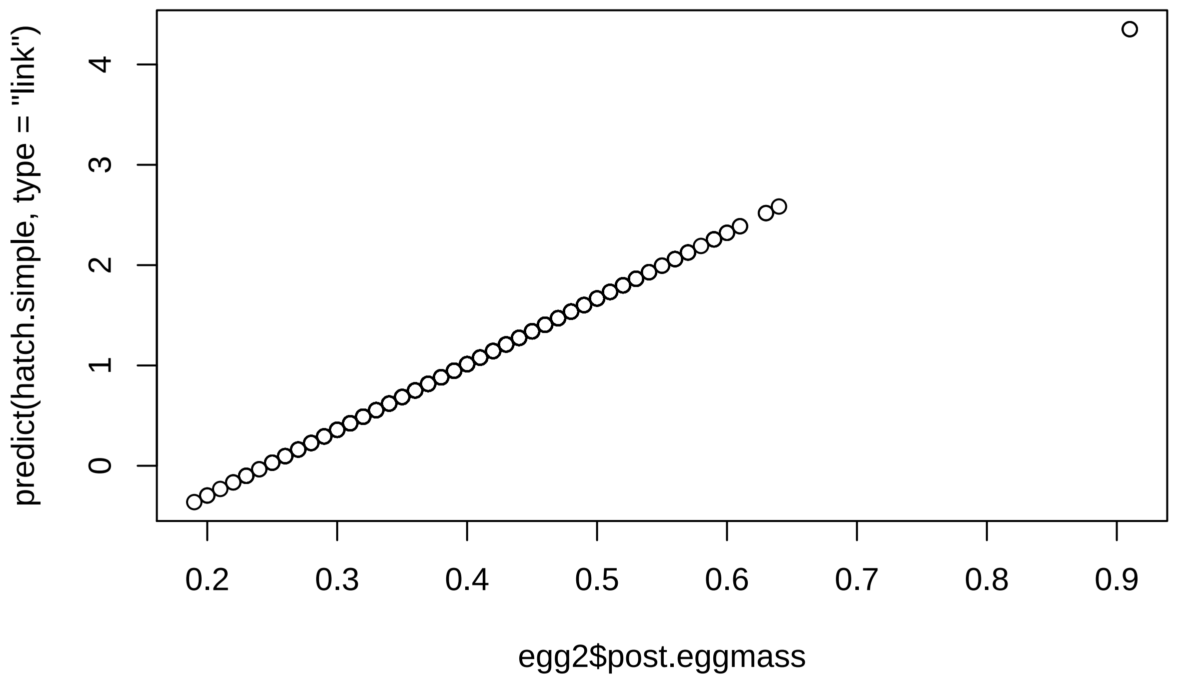
```
log(0.426/0.564)
```

```
## [1] -0.2806149
```

Therefore there is about a 43% chance that an egg that is 0.2g will survive until hatching.

We can also plot the predicted values against egg mass

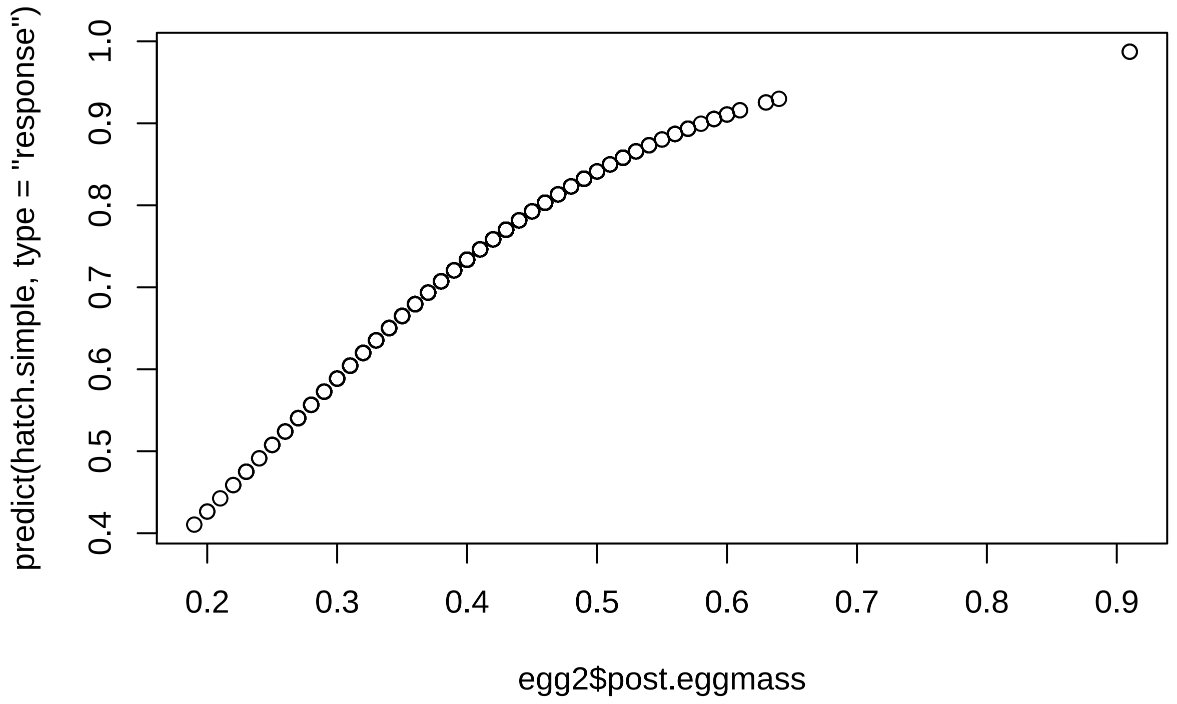
```
plot(egg2$post.eggmass, predict(hatch.simple, type="link"))
```



Note that this line has the slope and intercept that is presented in the model summary. However the logit link function means that these predicted values are not on the original scale that they were measured on.

We can instead get the predicted plot on the original raw scale by specifying type="response".

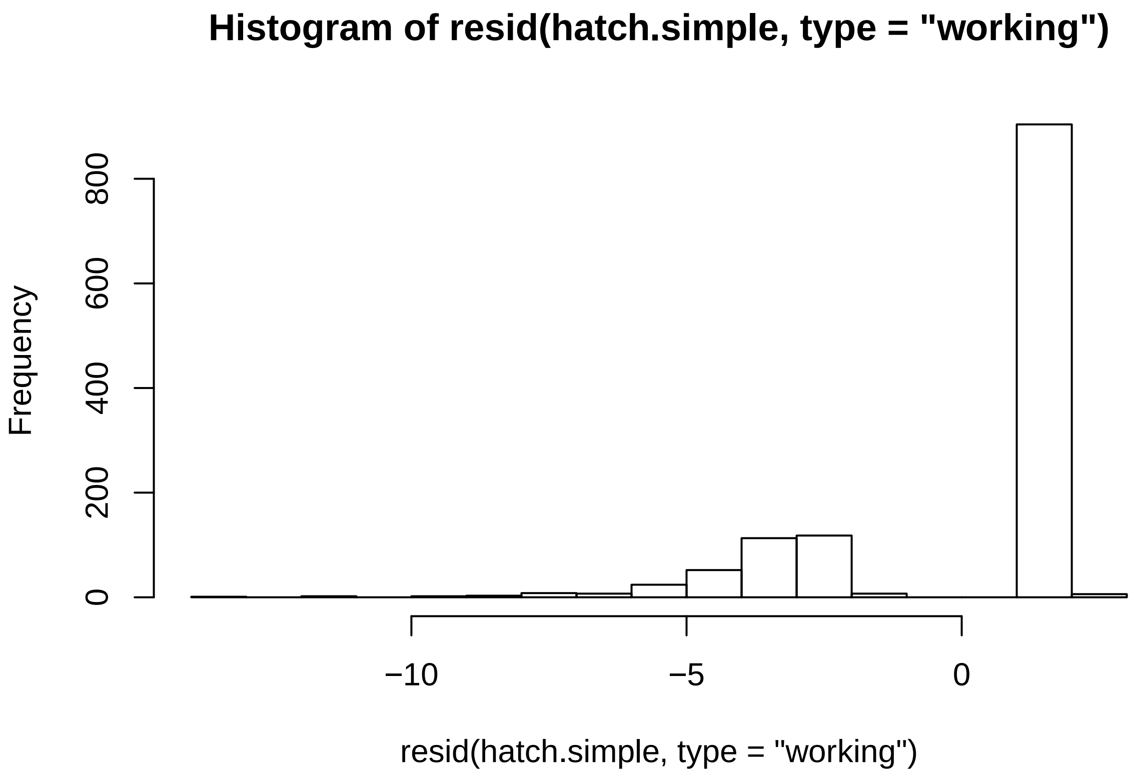
```
plot(egg2$post.eggmass, predict(hatch.simple, type="response"))
```



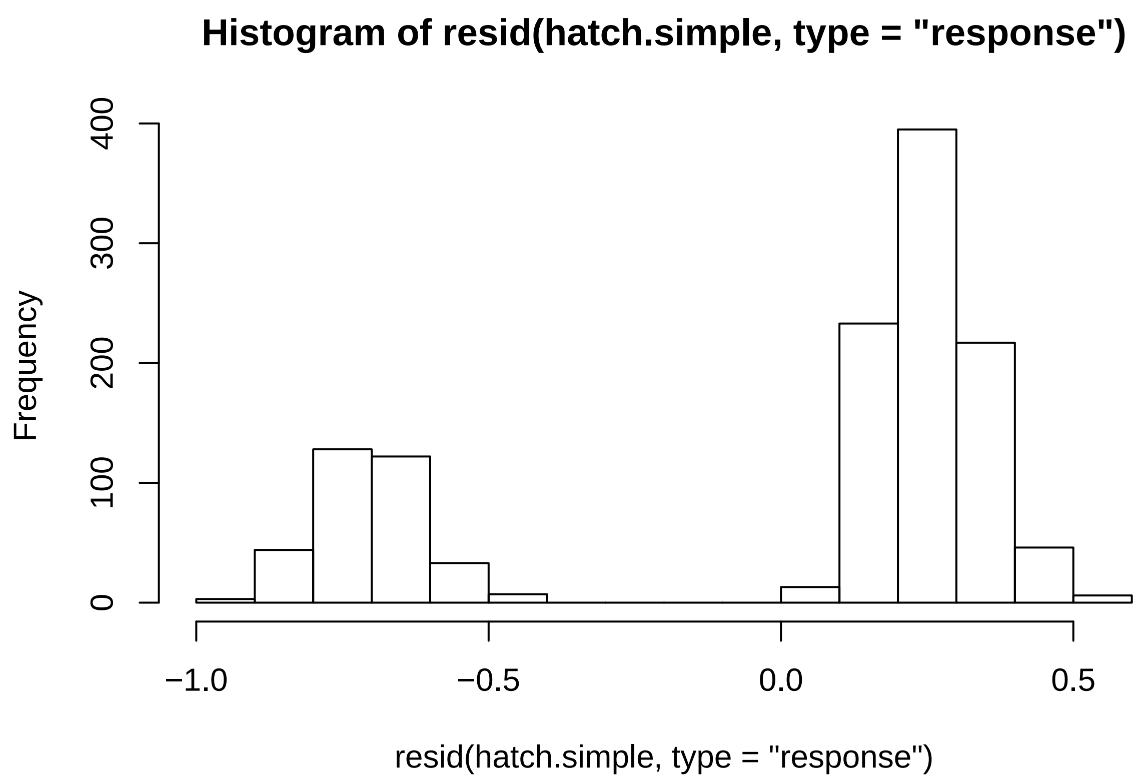
Residuals

With GLiM's there are also 4 different types of residuals

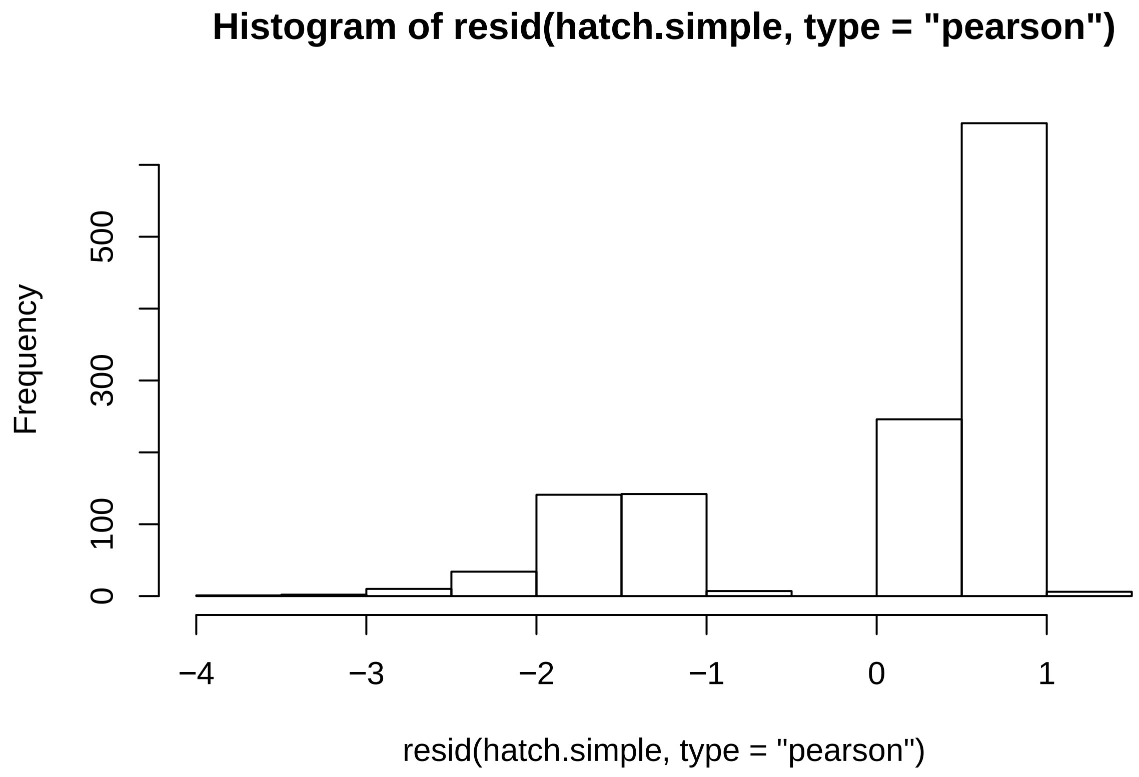
```r
hist(resid(hatch.simple, type="working"))
```



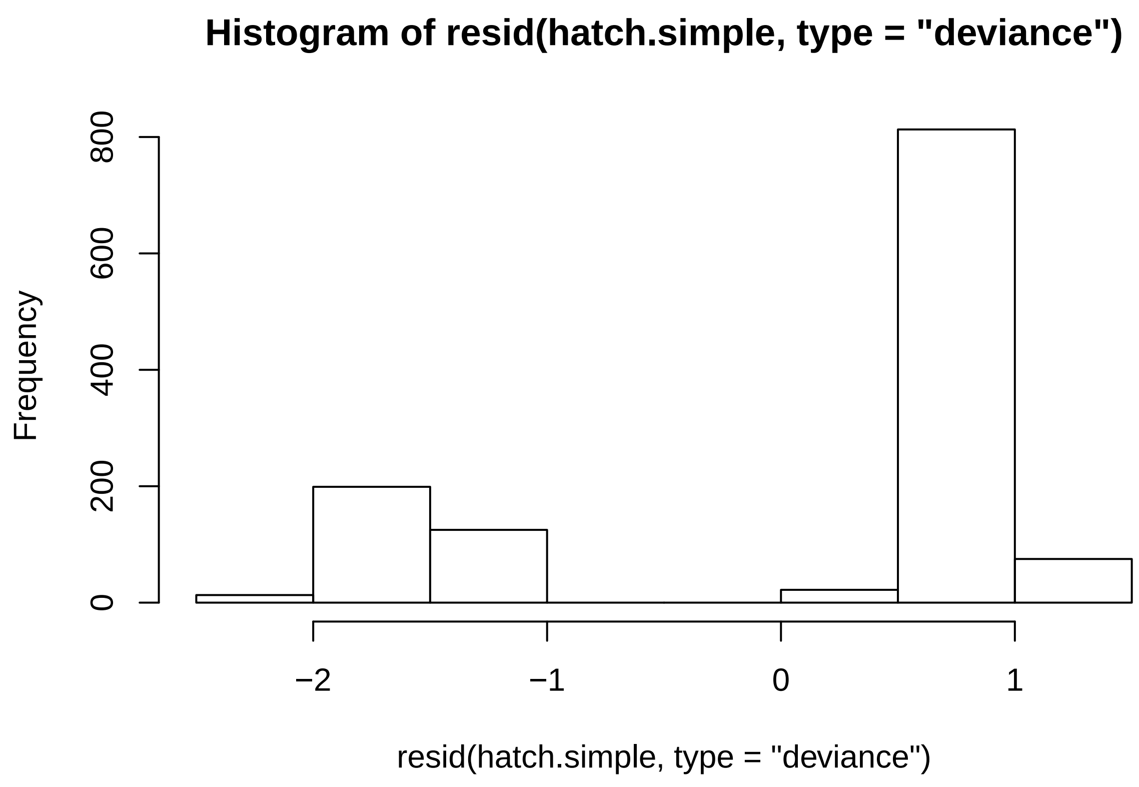
```r
hist(resid(hatch.simple, type="response"))
```



```
hist(resid(hatch.simple, type="pearson"))
```



```
hist(resid(hatch.simple, type="deviance"))
```



Deviance residuals are the best for diagnostic plots.