Statistics 520 Computing Notes
Basic Generalized Linear Models

On the course web page in the Computing Functions folder is an entry called `basicglm.txt` that is a link to a set of functions for fitting and analyzing basic generalized linear models. While there are a set of functions here, the one you need to call is named `basic.glm`.

Syntax

The syntax of `basic.glm` is

```
object<-basic.glm(xmat,y,link,random,startb=0,ns=1,ews=0,pwr=0,outfile=F)
```

Here,

1. `xmat` is the matrix of covariates to use in the regression model, including a column of 1s if an "intercept" term is to be used in the model and should have dimension $n \times p$.

2. `y` is an n vector of response observations.

3. `link` is an integer that indicates the link function to be used. Current choices are:

   (a) 1 = identity link, $g(\mu) = \mu$

   (b) 2 = log link, $g(\mu) = \log(\mu)$; $\mu > 0$

   (c) 3 = logit link, $g(\mu) = \log\{\mu/(1-\mu)\}$; $0 < \mu < 1$

   (d) 4 = log-log link, $g(\mu) = -\log\{-\log(\mu)\}$; $0 < \mu < 1$

   (e) 5 = complementary log-log link, $g(\mu) = \log\{-\log(1-\mu)\}$; $0 < \mu < 1$

   (f) 6 = inverse link, $g(\mu) = 1/\mu$

   (g) 7 = square inverse link, $g(\mu) = 1/\mu^2$

(h) 8 = power link, $g(\mu) = \mu^{\lambda}$; $\mu > 0$

4. `random` is an integer that indicates the random component to be used. Current choices are:

   (a) 1 = Poisson

   (b) 2 = Binomial

   (c) 3 = Binary

   (d) 4 = Normal

   (e) 5 = Gamma

   (f) 6 = Inverse Gaussian

5. `startb` is an optional argument that gives starting values for the estimation algorithm. If this argument is not specified, the function determines starting values automatically.

6. `ns` is an optional argument that gives a vector of binomial sample sizes if a binomial random component is used.

7. `ews` is an optional argument that gives a vector of external weights to be used, if that is appropriate for the model.

8. `pwr` is an optional argument that specifies a power if a power link function is to be used.

9. `outfile` is an optional argument that gives the name of a file to which results should be written. This argument should be enclosed in double quotes as "filename".

The arguments `startb`, `ns`, `ews`, `pwr`, and `outfile` are all optional. If `startb` is not given, the function will try to determine appropriate starting values automatically.

If the function fails to run with automatically determined starting values you will need to provide them explicitly. If these optional arguments are to be used, it is wise to give them by name. For example, to run `basicglm` for a model with Poisson random component, log link, and with an $X-$matrix contained in the object 'myxmat', responses contained in the object 'myys', starting values contained in the object 'mystart', and to produce an external file 'myresults.txt' containing output, one would use

```
> trial<-basicglm(myxmat,myys,1,2,startb=mystart,outfile="myresults.txt")
```

The output of the function `basicglm` that would be saved in the object 'trial' immediately above consists of a list of 4 components with the names `$estb`, `$ests`, `$vals`, and `$invinf`. The contents of these components are as follows.

1. `$estb` contains the estimated regression coefficients.

2. `$ests` contains other estimated quantities with additional names `$phi` for the estimated dispersion parameter, `$loglik.sat` for the saturated model log likelihood (computed with the estimated value of ϕ), `$loglik.fitted` for the fitted model log likelihood evaluated at the parameter values $\hat{\boldsymbol{\beta}}$ and $\hat{\phi}$, `$udev` for the unscaled deviance, `$sdev` for the scaled deviance, and `$pcs` for Pearson's chi-squared statistic.

3. `$vals` is a data frame having components `$y` for the observed responses, `$muhat` for the estimated expected values, `$rawres` for the raw residuals, `$devres` for the deviance residuals, `$stdevres` for the standardized deviance residuals, and `$pearsonres` for the Pearson residuals. Note here that `basicglm` produces deviance residuals as given in the course notes using terms in the scaled deviance. Some authors would call these standardized deviance residuals. What is given by the function as standardized deviance residuals are

defined in expression (4.36) of the course notes, and some authors would call these studentized deviance residuals.

4. `$invinf` contains the expected inverse information matrix for the regression coefficients.

A Note on Maximized Likelilhoods

The function `basicglm`, like many other functions both packaged and ones I might provide, gives a value of the maximized log likelihood. If you want to use such values for additional computations or to compare the results of different functions you must know how they are computed. In an algorithm to maximize a function any constants may be dropped because they will not influence the outcome. But,if you want to construct a likelihood ratio test, for example, using one part (the full model, let's say) computed without constant terms and the other part (the reduced model, let's say) computed with constant terms will not produce a valid test. You may well need to evaluate a log likelihood externally to what is produced by some function to determine how that likelihood is being computed.

A Note on the R Function glm

I don't know all about the built-in R function `glm`, but I am aware of the following points:

1. The form of exponential dispersion families used by `glm` is parameterized using the reciprocal of the dispersion parameter ϕ in our course notes.

2. What is reported by `glm` as Residual Deviance is the unscaled deviance.

3. The default residuals produced by `glm` appear to be not deviance residuals, but a "working" residual from the last iteration of the Fisher scoring algorithm for maximum likelihood estimation of regression parameters (I believe these are the $\tilde{\boldsymbol{z}}$ from expression (4.24) in the notes.

Simulating an Example

One of the functions included in the Generalized Linear Models link on the course web page is called `simbasicglm` and it simulates from basic generalized linear models. The syntax of this function is

```
simbasicglm(b,xmat,phi,link,random,ns=1,pwr=0)
```

The arguments are

1. `b`is a vector of regression parameters
2. `xmat` is a matrix of covariates, just as in `newbasic.glm`
3. `phi` is a value for the dispersion parameter (put in 1 for binomial or binary
4. `link` is an integer indicating the link function, with values the same as `newbasic.glm`
5. `random` is an integer indicating the random component. Currently, all that are available are 1 =Poisson, 2 =binomial, 3 =binary, 4 =normal and 5 =gamma.

The function produces a vector of response values.

Let's try this for a model with one type of covariate x_i, which will be 35 equally spaced values between 0.1 and 15, a gamma random component, and a power link function having power 0.25,

$$g(\mu_i) = \mu_i^{0.25} = \beta_0 + \beta_1 x_i = \eta_i.$$

After a little trial and error, parameter values of $\beta_0 = 0.5$ and $\beta_1 = 0.1$ were chosen as suitable.

```
jxmat<-matrix(c(rep(1,35),seq(0.1,15,15/35)),35,2,byrow=F)
examp1<-simbasicglm(c(0.5,0.1),jxmat,2.0,8,5,pwr=0.25)
```

A scatterplot of the simulated values of is shown in Figure 1 Although these are just simulated values, the pattern in Figure 1 is somewhat reminiscent of certain problems involving chemical concentrations as a function of a covariate. Now we know the true random component and link function used to simulate these values, and we will first use that information in fitting a basic generalized linear model. If this were an actual application we would of course need to determine what model to use, and we will consider such an analysis later. These data are on the course web page in the Data folder as `glmexample1`.

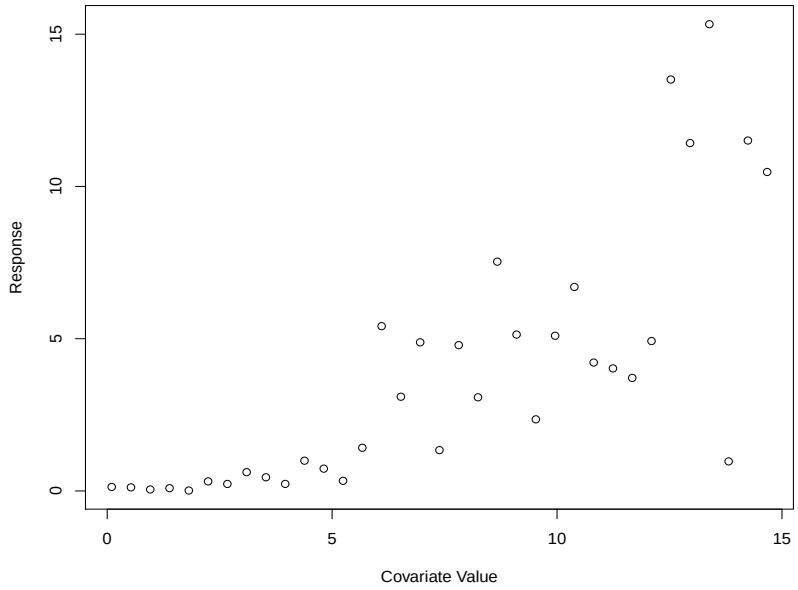

Figure 1: Scatterplot of simulated data.

Fitting the Actual Model

To fit the model using a gamma random component and power link with power 0.25,

```
exampresult1<-basicglm(jxmat,examp1,8,5,pwr=0.25)
```

```
And what we see on the screen is
```

```
MODEL DEFINITION:
Random Component: Gamma
Link Function: Power

Number of Covariates (including intercept):  2
Number of Observations: 35
```

```
ESTIMATION RESULTS:

Coefficient Estimates:
0.5095377
0.1011802
Inverse Information:
              [,1]          [,2]
[1,]  0.0017911047 -2.344833e-04
[2,] -0.0002344833  5.706958e-05
Estimated Phi:
2.167378
Unscaled Deviance:
17.809
Scaled Deviance:
38.59885
Saturated Model Log Likelihood:
-19.96295
Fitted Model Log Likelihood:
-39.26237
Warning messages:
1: In log(1 - y) : NaNs produced
2: In log(1 - muhat) : NaNs produced
```

To add true and estimated expectation functions to our scatterplot:

```
eta<-0.5+0.1*jxmat[,2]
```

```
tru
lines(jxmat[,2],true)
lines(jxmat[,2],exampresult1$vals$muhat,lty=2)
legend(1,15,legend=c("True","Estimated"),lty=c(1,2))
```

and the result should be Figure 2.

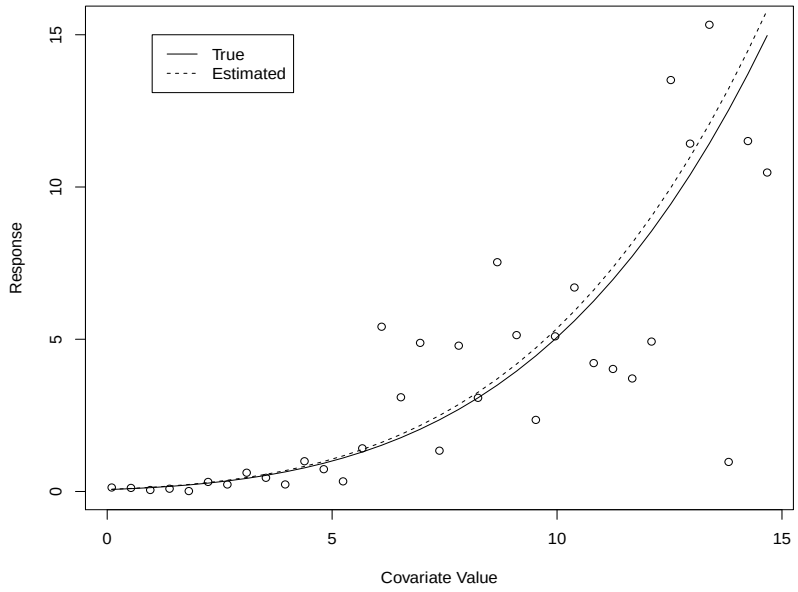

Figure 2: Scatterplot of simulated data with true and estimated expectation functions.

Visually, the fitted model is doing quite well, which we would expect. While inferential quantities (e.g., interval estimates, confidence band for the expectation function, etc.) could be produced from the output of `basicglm` we will present only a standard residual plot in this example. To construct the residual plot requires only the commands

```
plot(glmexampresult1$vals$muhat,glmexampresult1$vals$stdevres)
```

```
abline(0,0)
```

and results in the top panel of Figure 3. This plot gives an impression of decreasing spread of the residuals for increasing fitted value, but we must bear in mind the relative density of values along the horizontal axis. While the range of residuals appears smaller on the right-hand side of the plot than to the left, the number of residuals on the right is also smaller. If, as is sometimes recommended for residual plots, we change the scale of the horizontal axis by plotting standardized deviance residuals against the logarithm of the fitted values we obtain the plot shown in the lower panel of Figure 3. This plot certainly does not give the same visual impression as the plot in the upper panel of the figure. But somehow it is not entirely satisfying either.

There are several possibilities that might be suggested for why these residual plots seem less than "perfect", even for simulated data and for estimation of the correct model. First, it could be that because of the combination of random model component and rather severe link function, what we see in Figure 4 is "as good as it gets" for this model. But another possibility is that, due to variability in realizations of the model, the values in this particular data set might actually be better described by something other than what we know is the true model.

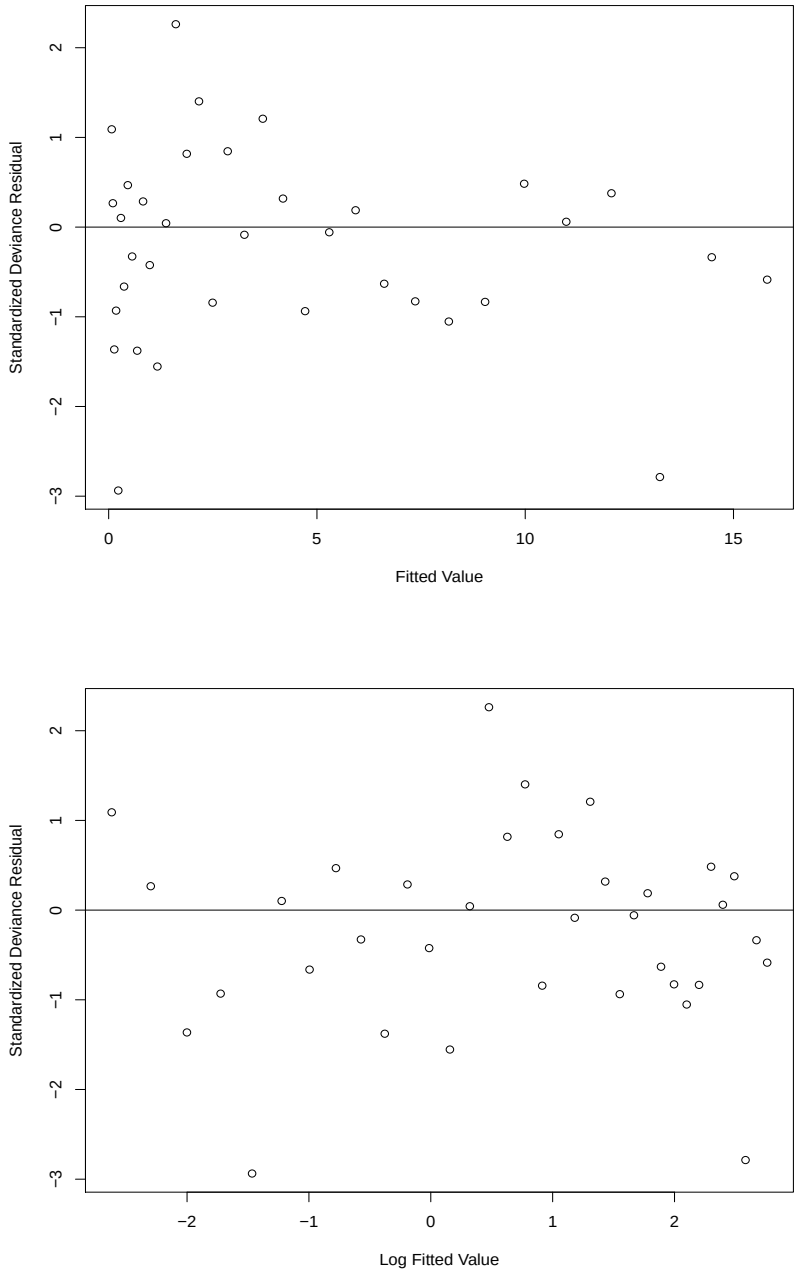

Figure 3: Residual plots for simulated example.

Assuming the Model is Not Known

Now let's do our best to approach selection and estimation of a basic generalized linear model as if we did not know what the true model was that generated the data we have available. The scatterplot in Figure 1 indicates increasing variances as expected values increase and an expectation curve that is convex. Beginning with the random component we might construct our basic Box-Cox plot to examine the relation between means and variances. With $n = 35$ observations we cannot create too many bins and have a sufficient number of observations in each bin, so let's try 5 bins.

```
j<-regmeanvar(jxmat[,2],examp1,5)
```

This results in the values jof Table 1.

Bin	Mean	Variance	n
1	0.2050545	0.04387229	7
2	1.3673327	3.35072923	7
3	4.2641122	3.90182931	7
4	4.4322938	1.81406863	7
5	10.5370499	25.01095999	6

Table 1: Group means and variances for construction of a Box-Cox plot.

A Box-Cox plot can constructed from these values using commands

```
plot(log(j$mean),log(sqrt(j$vars)),xlab="Log Group Mean",ylab="Log Group Std. Deviati
q<-matrix(c(rep(1,5),log(j$mean)),5,2,byrow=F)
solve(t(q)%*%q)%*%t(q)%*%log(sqrt(j$vars))
           [,1]
[1,] -0.2467865
[2,]  0.7118708
```

```
abline(-0.247,0.712)
```

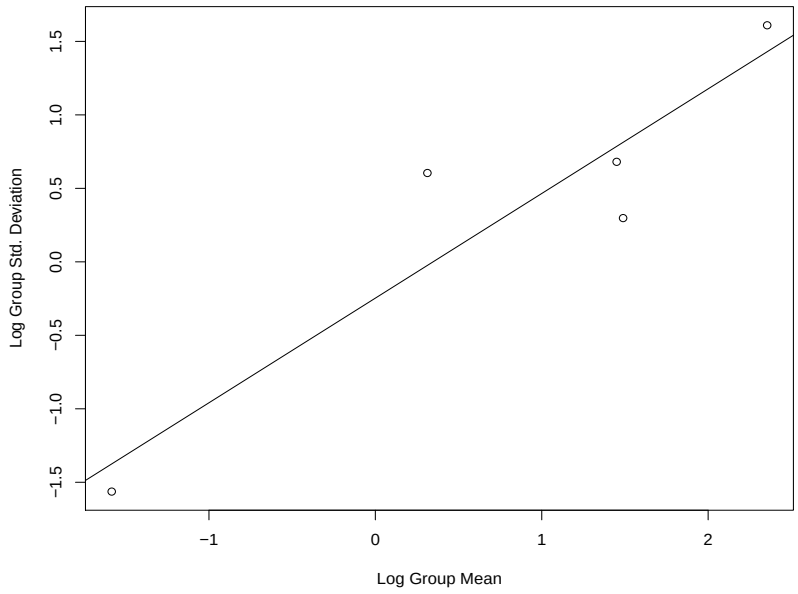

Figure 4: Box-Cox plot constructed from the values of Table 1.

Although constructed from a small number of bins, Figure 4 does suggest that a straight line could be used to roughly describe the relation between log mean and log standard deviation, with a slope of about 0.7, which suggests a variance function of something like $V(\mu_i) = \mu_i^1.4$, which is a bit smaller power than the μ_i^2 of a gamma random component but certainly does not suggest a lager power such as the μ_i^3 of an inverse Gaussian random component. At the risk of forming smaller groups we might see what happens if we increase the number of bins used in construction of the Box-Cox plot to 7, say. A Box-Cox plot constructed using 7 bins is presented in Figure 5, and the least squares line has a slope of 0.936 which does point to a gamma random component. So we have some indication of a gamma random component with no suggestion of either a normal or inverse Gaussian alternative.

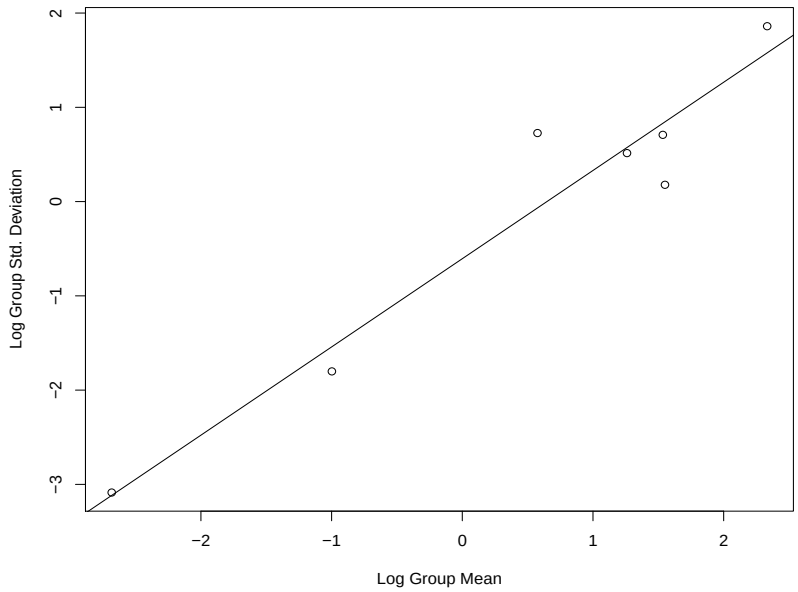

Figure 5: Box-Cox plot constructed from binning the data into 7 groups.

Turning attention to the choice of link function, we might start with consideration of a log link to avoid the need to choose a power. Plotting the logarithm of responses against the covariate values produces the plot in Figure 6. With the exception of a couple of points, a straight line seems to be a reasonable description of this plot, so we might try a model with gamma random component and log link. The call to `newbasic.glm` needed for fitting this model has already been given, except for the change from a power link to a log link in the argument. A plot similar to that of Figure 2 is presented in Figure 7, showing the true expectation function along with that estimated using a log link function. The estimated curve is not as close to the true curve as was true in Figure 2, but in the absence of knowledge of the truth the estimated expectation function in Figure 7 does not appear visually unreasonable.

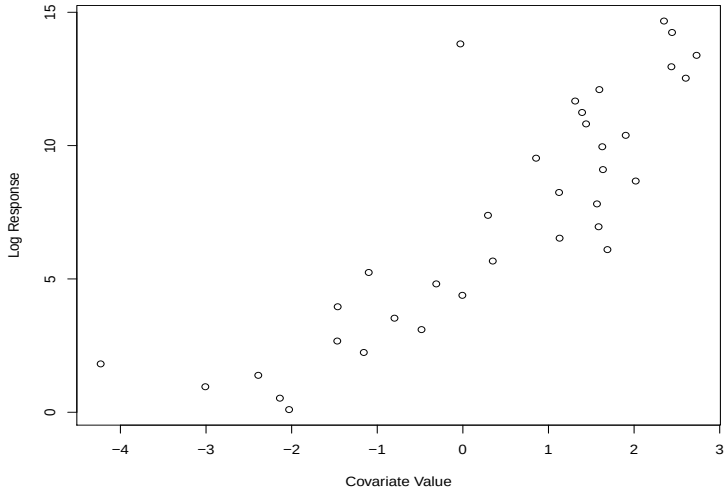

Figure 6: Scatterplot of log transformed responses against covariates.

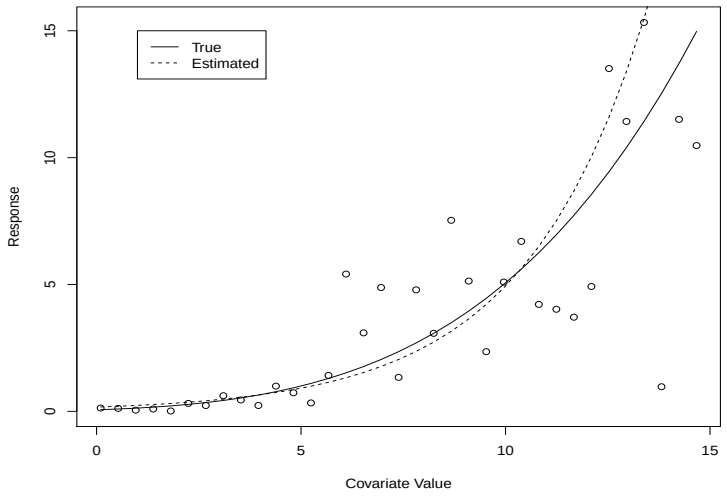

Figure 7: Scatterplot of simulated data along with true expectation function and that estimated from a model with log link.

We can again examine residual plots for this estimated model, and plots analogous to those of Figure 3 are presented in Figure 8. While the plot in the upper panel of Figure 8 is perhaps even less pleasing than the corresponding plot in Figure 3, it has the same basic pattern and varies greatly in the density of points across the horizontal axis. It is the plot in the lower panel of Figure 8 that casts serious doubt on a model with a log link function. While the behavior of this plot for large fitted values may also be discerned from the scatterplot and fitted curve of Figure 7, the same is not true for the behavior at small fitted values. A model with log link is not entirely capable of fitting small and large values because it curves too strongly at the ends. This does suggest a power link function with a power somewhat greater than 0 (which corresponds to a log link) but less than 1 (which corresponds to a straight line).

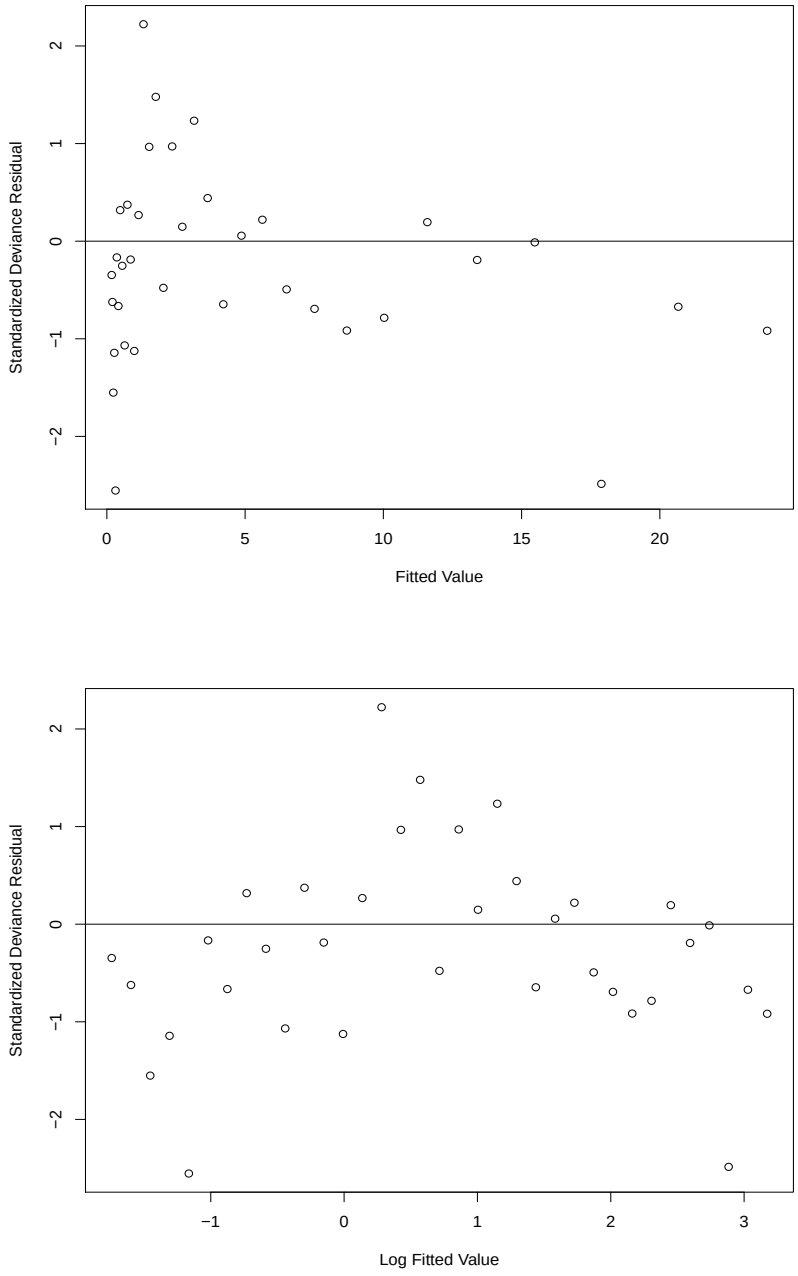

Figure 8: Residual plots for simulated example.