Goodness of Fit Testing for Logistic Regression

October 15, 2018

The purpose of this lesson is to investigate goodness of fit tests for logistic regression. Specifically, we are interested in testing whether or not a logistic regression model really fits the data. We assume that,

$$P[Y = 1|X = x] \;=\; \pi(x),$$

where π is a reasonable function of x. Of course, the logistic regression model gives a specific functional for for π with some parameters to be estimated. Because of the nature of the response variable Y, namely that it is binary, it is often very difficult to apply the usual tools to assess model validity, like residual plots. This will be illustrated below.

We first wrote a function to simulate data from a logistic regression model:

```
Simlogistic = function(x,alpha,beta){
# function to simulate data from a logistic model
# INPUTS:
# x: n by p matrix of predictor values, or n vector if p=1
# alpha: intercept
# beta: p dimensional coefficient vector
# OUTPUTS:
# y: n vector of binary values with logit P[Y = 1 | X = x] = alpha + sum(beta*x)
if(is.matrix(x)){
logits = alpha + x%*%beta
n = nrow(x)
}
else {
logits = alpha + beta*x
n = length(x)
}
probs = 1/(1+exp(-logits))
y = rbinom(n,1,probs)
}
```

Next, we simulate some data from a logistic model with a quadratic term, and then fit an incorrect model with only a linear term.

```
> yquad = Simlogistic(cbind(x,x^2),-2,c(-1,.25))
> plot(x,yquad)
> fitquad = glm(yquad ~x,family=binomial)
> summary(fitquad)

Call:
glm(formula = yquad ~ x, family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6572  -0.6254  -0.2132   0.5274   2.7018

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.7729     0.7460  -5.058 4.24e-07 ***
x             0.7468     0.1371   5.446 5.16e-08 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 140.006  on 100  degrees of freedom
Residual deviance:  81.056  on  99  degrees of freedom
AIC: 85.056

Number of Fisher Scoring iterations: 5

> plot(x,fitquad$residuals,ylab="residuals",main="Working Residual Plot")
```

The raw data plot and residual plot are shown below. We should be able to see in the raw data plot that there are a few too many values of $y = 1$ on the left hand side, but it can escape notice. Also, it may not be possible to see if the predictor is multivariate. The residual plot definitely has some large values on the left side which is a cause for concern, but it doesn't obviously suggest the need for a quadratic term.

Regarding the output of the summary function, we see that both the intercept and slope parameters are significant. The null deviance is apparently the difference in 2 log likelihood for the model with no intercept vs. the model with intercept, based on the degrees of freedom, and it looks like it is significant based on a χ^2 approximation, which is of dubious accuracy in this setting (the d.f. is too big), which is telling us that the slope is significantly different from 0, as we can see more accurately in the p-value given. The residual deviance does not appear significant, based on the d.f.

In order to do a formal goodness of fit test, we follow a suggestion in the book and bin the data by x-values and then do a χ^2 goodness of fit test with the binned data. This isn't totally trivial to implement.

```
> grp = floor((1:101)/20)
> grp[grp==5] = 4
> grp = 1+grp
> tabquad = table(grp,yquad)
> tabquad
   yquad
grp  0  1
  1 16  3
  2 18  2
  3 15  5
  4  2 18
  5  0 22
> grpfv = tapply(fitquad$fitted.values,grp,mean)
> grpfv
         1          2          3          4          5
0.04626185 0.17018895 0.46394072 0.78420715 0.94337675
> # these are averaged values of the fitted probabilities pi(x)
> temp = as.vector(grpfv)
> temp = c(1-temp,temp)
> temp2 = table(grp)
> temp2=as.vector(temp2)
> temp2
[1] 19 20 20 20 22
> temp1 = temp
> temp1[1:5] = temp1[1:5]*temp2/101
> temp1[6:10] = temp1[6:10]*temp2/101
> sum(temp1)
[1] 1
> chisq.test(as.vector(tabquad),p=temp1)
```

```
Chi-squared test for given probabilities

data:  as.vector(tabquad)
X-squared = 12.65, df = 9, p-value = 0.1791

Warning message:
In chisq.test(as.vector(tabquad), p = temp1) :
  Chi-squared approximation may be incorrect
> pchisq(12.65,df=7,lower.tail=F)
[1] 0.08110936
```

Note that we had to correct the d.f. of the chi-squared from 9 to 7 since the test we are using doesn't "know" that we estimated 2 parameters. The p-value is on the small side, but not small enough to reject the null hypothesis at the usual 0.05 level. The null hypothesis here is

$$H_0 : \quad \text{logit}\pi(x) \;=\; \alpha + \beta x, \text{ for some } \alpha,\, \beta.$$

That is, that the logistic model is correct.

We have a warning that the chi-squared approximation may be inaccurate (a better adjective than "incorrect"). There is no way I can think of to do an exact test in this case (as we can do when testing independence). Probably, the best we can do is a parametric bootstrap. Assuming the logistic regression model is correct, our best guess at the coefficient vector is the MLE. So, we simulate data from the fitted model, and compute the test statistic for each simulated data set. The proportion of simulated test statistic values greater or equal to the observed value then gives us an approximate p-value. Asymptotic theory shows this works well to control the type I error probability.

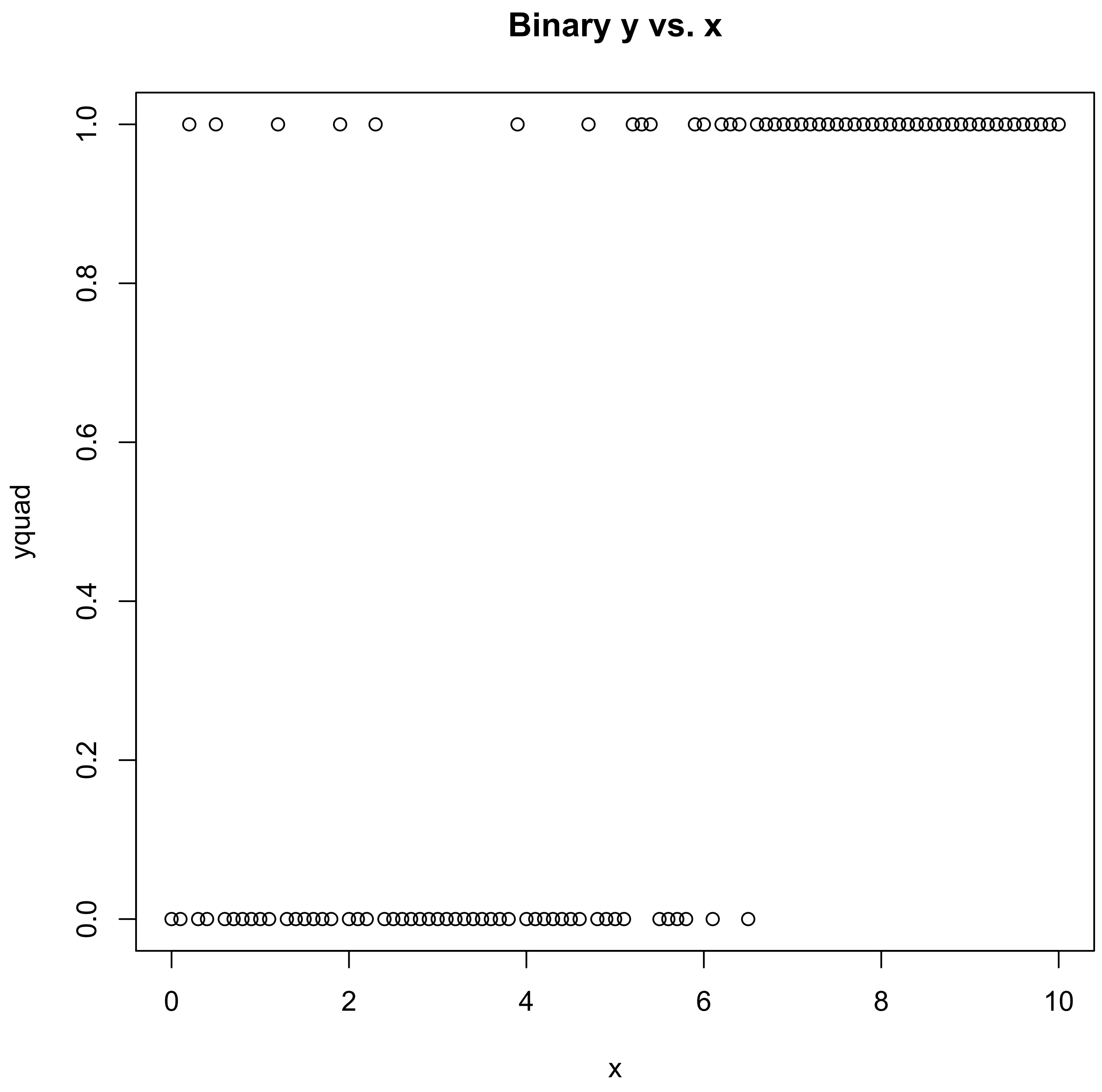

Figure 1: Plot of raw data.

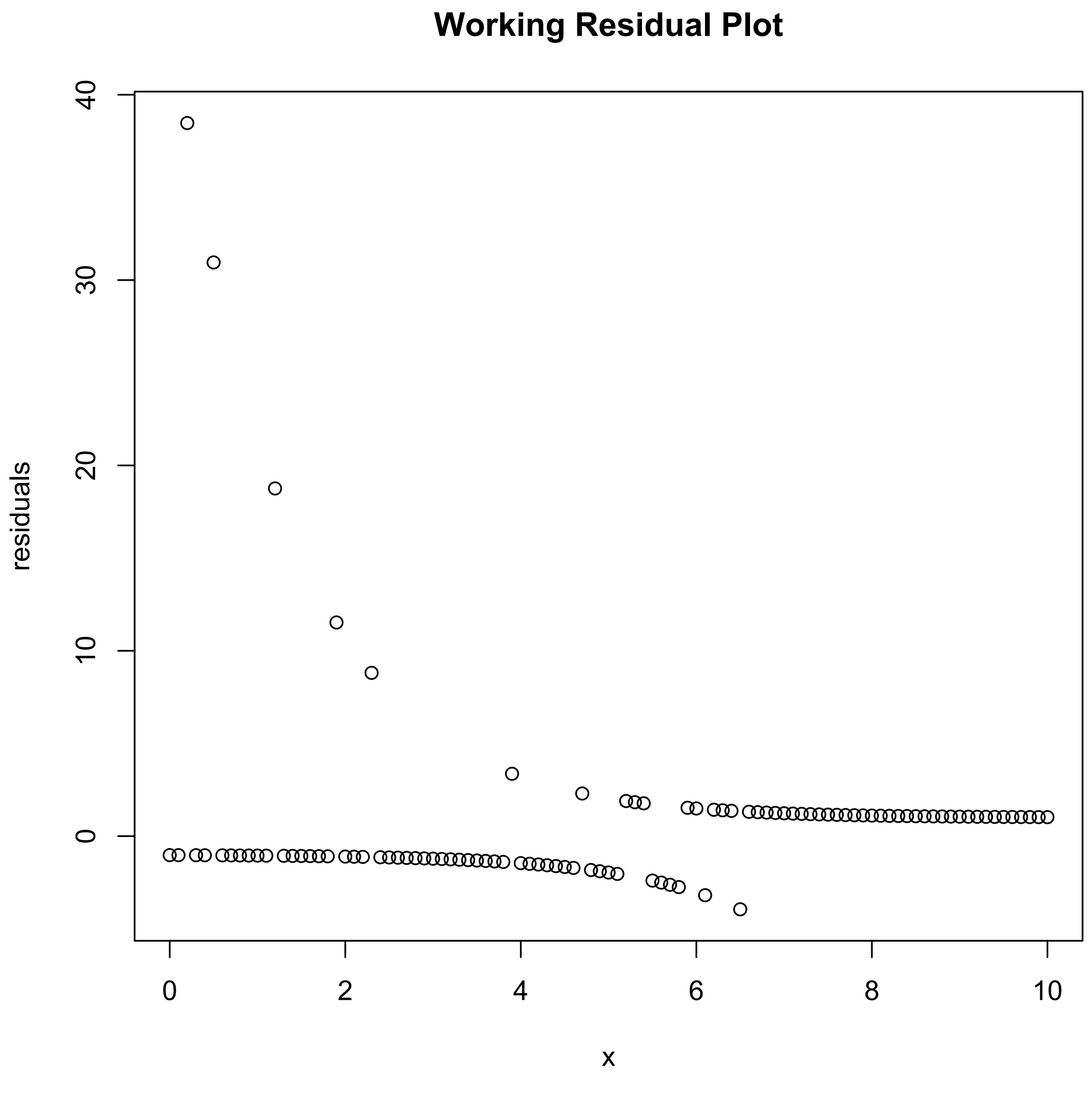

Figure 2: Plot of working residuals.