Generalized Linear Models

August 13, 2015

Link Function

The link function η is the function that *links* $\mu = E[y]$ to $X \times b$, i.e.,

$$\eta(\mu) = X \times b.$$

In the case of the logistic regression, the link function is

$$\log(\frac{\mu}{1-\mu})$$

But there are many other link functions for other types of data.

Other Distributions

Here are a bunch of other families of variables you can estimate by `glm`

You will use `glm( .... , family=`

- `binomial(link = "logit")`
- `gaussian(link = "identity")`
- `Gamma(link = "inverse")`
- `inverse.gaussian(link = "1/mu^2")`
- `poisson(link = "log")`
- `quasi(link = "identity", variance = "constant")`
- `quasibinomial(link = "logit")`
- `quasipoisson(link = "log")`

Example: Poisson

- Number of users visiting my website is Poisson.
- Does it depend on the tempretaure of that day?

```
# data generation
temp = runif(1000, min=50, max=100)
meanvis = 50 + temp/10
numvisits = rpois(n=1000,lambda = meanvis)
m1 = glm(numvisits~temp , family = poisson)
```

Results from the example

```
summary(m1)
```

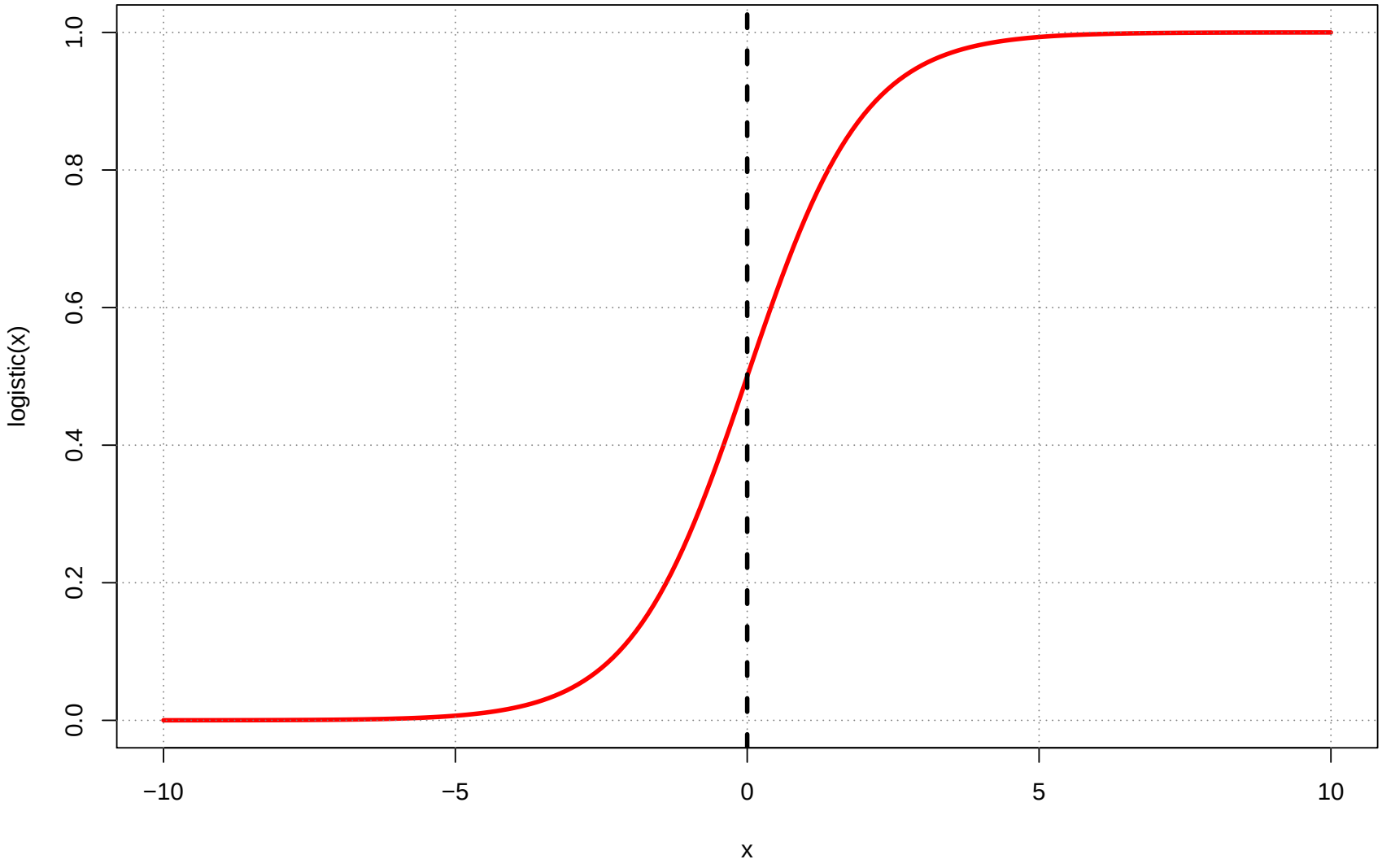
```
##
## Call:
## glm(formula = numvisits ~ temp, family = poisson)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1246  -0.7322  -0.0232   0.6925   3.2314
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 3.9733107  0.0219789 180.779  < 2e-16 ***
## temp        0.0010545  0.0002869   3.676 0.000237 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1024.4  on 999  degrees of freedom
## Residual deviance: 1010.9  on 998  degrees of freedom
## AIC: 6899.3
##
## Number of Fisher Scoring iterations: 4
```

2-Other models beyond `glm`

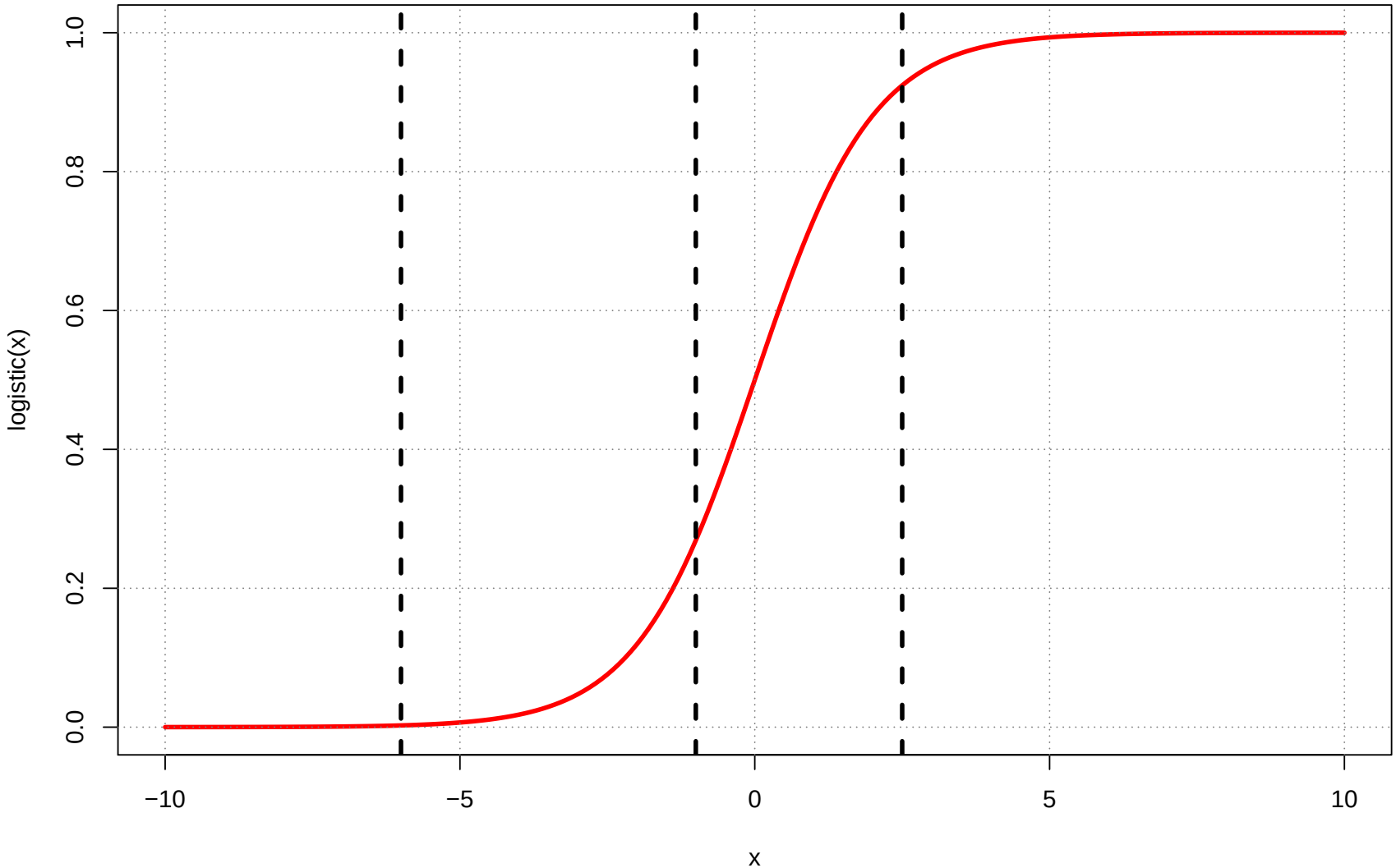
There are other models. Multinomial logit and ordinal logit are perhaps the most useful for you. Mulitnomial logit is more complicated so look it up for yourself if you wish, but ordinal logit is a simple extension of logit.

Instead of two outcomes=two regions (separated at middle) of the S-shaped curve of the logistic function, we can imagine there are many breaking points leading to many regions, each region corresponding to an outcome; and we let the algorithm figure optimal points for these breaking points.

That means, we had



And we are going to have this



Example

In last week's analysis, we assumed that partyid, ranging from strong Democratic to strong Republican was an interval variable. Well, with ordinal logit we can do the analysis without that assumption (although we need to make a few others):

```
library(MASS)
# Using week 13 data
# Data from GSS
fpartyid = factor(GSS$partyid) #has to be a factor
m <- polr(fpartyid~daughter1+educ, data=GSS, Hess=TRUE)
summary(m)
```

3- Using predict

predict can be used to predict outcomes of lm and glm.

Its usage for predicting outcome for newdata is like the following.

For glm you have to specify whether you want type = link (i.e., just $X \times b$) or type=response (the transformed $X \times b$).

```
preict(model , newdata )
```

Simple example of predict

```
x = runif(1000)
gender = c(rep('MALE',500) , rep('FEMALE',500) )
y = ( 1 + x + 2 * (gender=='MALE')  + rnorm(1000, mean=0,sd=.7) )  > 3
model = glm(y~x+gender , family = binomial)
model
```

```
##
## Call:  glm(formula = y ~ x + gender, family = binomial)
##
## Coefficients:
## (Intercept)            x   genderMALE
##      -5.524        1.985        5.840
##
## Degrees of Freedom: 999 Total (i.e. Null);  997 Residual
## Null Deviance:       1339
## Residual Deviance: 574.1     AIC: 580.1
```

Example, continued

```
#increase in response for males when there is a change in x
#from 0.5 to 1
ndata = data.frame( x = c(.5, 1) , gender = c('MALE' , 'MALE'))
ndata
```

```
##     x gender
## 1 0.5   MALE
## 2 1.0   MALE
```

```
predict(model, newdata = ndata, type = 'link')
```

```
##        1        2
## 1.308037 2.300681
```

```
predict(model, newdata = ndata, type = 'response')
```

```
##         1         2
## 0.7871846 0.9089334
```

```
 prd.pr = predict(model, newdata = ndata, type = 'response')
cat('change in probabiliy = ', prd.pr[2]-prd.pr[1] )
```

```
## change in probabiliy =  0.1217489
```