Data 609 - HW 4

Avery Davidowitz

2023-03-17

```
library("ggplot2")
```

Ex. 1

For example 19 on page 79 in the book, carry out the regression using R.

```
x <- c(-.98, 1, 2.02, 3.03, 4)
y <- c(2.44, -1.51, -.47, 2.54, 7.52)
lm1 <- lm(y ~ x)
summary(lm1)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##       1       2       3       4       5
##  2.9547 -2.8511 -2.7671 -0.7037  3.3671
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.4038     2.2634   0.178    0.870
## x             0.9373     0.9058   1.035    0.377
##
## Residual standard error: 3.481 on 3 degrees of freedom
## Multiple R-squared:  0.263,  Adjusted R-squared:  0.01739
## F-statistic: 1.071 on 1 and 3 DF,  p-value: 0.3769
```

So we have a simple linear model with $\beta_0 = .4038$ and $\beta_1 = .9373$ having x account for .263 of the variability of y.

Ex. 2

Implement the nonlinear curve-fitting of example 20 on page 83 for the following data:

```
x <- c(.1, .5, 1, 1.5, 2, 2.5)
y <- c(.1, .28, .4, .4, .37, .32)
df <- data.frame(x, y)
fx <- function(x, a, b){
```

```r
  x / (a + (b * x^2))}

nlm1 <- nls(y ~ fx(x, a, b), data = df , start = list(a = 1, b = 1))
summary(nlm1)
```

```
##
## Formula: y ~ fx(x, a, b)
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a  1.48544    0.08777   16.92 7.15e-05 ***
## b  1.00212    0.05019   19.96 3.71e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01739 on 4 degrees of freedom
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 3.899e-07
```

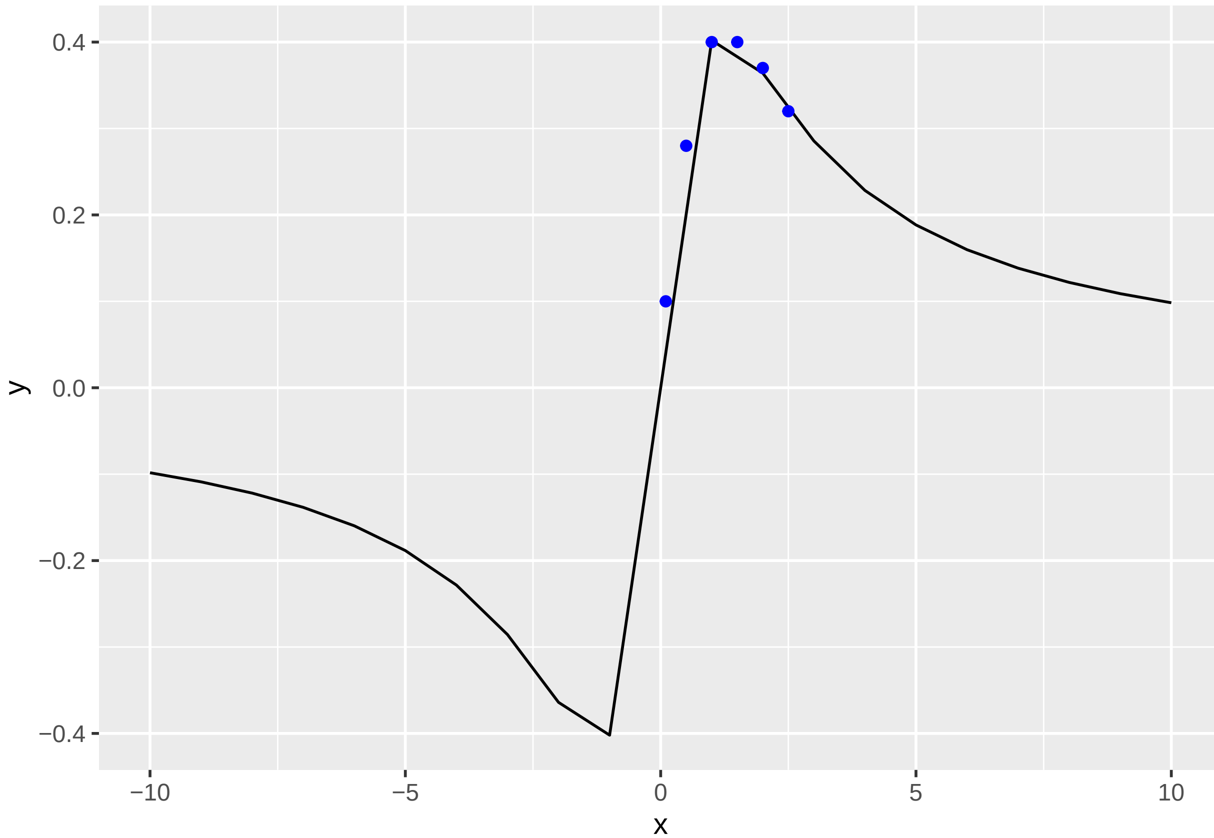
So the non-linear curve that fits these data points is:

$$y = \frac{x}{1.48544 + 1.00212x^2}$$

We can verify by plotting the fitted function and the original data points.

```r
data_fun <- data.frame(x = - 10:10,
                       y = c(fx(- 10:10, 1.48544, 1.00212)))

ggplot(data_fun, aes(x, y)) +
  geom_line() +
  geom_point(data = df, aes(x = x, y = y), color = "blue")
```



The points are reasonably close to the fitted non linear function.

Ex. 3

For the data with binary y values, try to fit the following data to the nonlinear function

$$y = \frac{1}{1 + e^{a+bx}}$$

starting with a = 1 and b = 1.

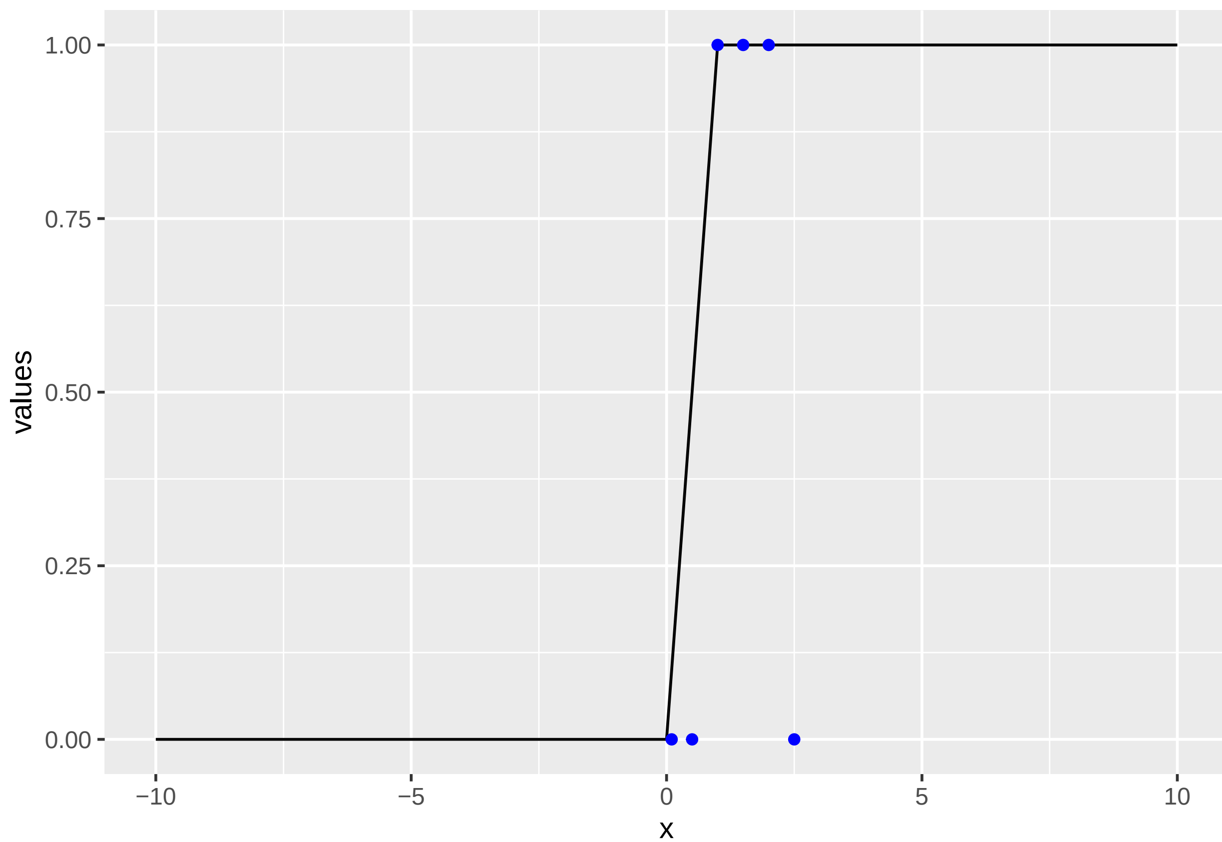
```
x <- c(.1, .5, 1, 1.5, 2, 2.5)
y <- c(0, 0, 1, 1, 1, 0)
df2 <- data.frame(x, y)
fx2 <- function(x, a, b){
  1 / (1 + exp(a + b * x))}
nlm2 <- nls(y ~ fx2(x, a, b), data = df2, start = list(a = 1, b = 1))
summary(nlm2)
```

```
##
## Formula: y ~ fx2(x, a, b)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## a     36.42  211289.32       0        1
## b    -48.51  261890.12       0        1
```

```
##
## Residual standard error: 0.5 on 4 degrees of freedom
##
## Number of iterations to convergence: 12
## Achieved convergence tolerance: 7.656e-06
```

```r
data_fun <- data.frame(x = - 10:10,
                       values = c(fx2(- 10:10, 36.42, -48.51)))

ggplot(data_fun, aes(x, values)) +
  geom_line() +
  geom_point(data = df2, aes(x = x, y = y), color = "blue")
```



It does not appear that the binary variables given can possibly be fit to the function $y = \frac{1}{1+e^{a+bx}}$