MTH 4230 R Notes 12

1 Generalized Linear Models in R

- A ***generalized linear model*** (not to be confused with **general linear model**) is a regression model for which the **response** variable Y follows a **non-normal** distribution.
- Two important special cases are ***logistic regression***, for which the response is **Bernoulli** (**0** or **1**), and ***Poisson regression***, for which it's **Poisson**.
- We can carry out ***logistic*** and ***Poisson regression analyses*** using the **generalized linear model** function.

```
glm()          # Fit a generalized linear model
```

The syntax for `glm()` is similar that of `lm()`, except that now we have to specify the distribution of the response variable via the argument `family`.

1.1 Logistic Regression in R

- For ***logistic regression***, the response is a ***Bernoulli*** random variable whose "success" probability $\boldsymbol{\pi = \pi(X)}$ is a function of a predictor variable $\boldsymbol{X}$.

  A **Bernoulli** random variable is a special case of a ***binomial*** random variable for which there's only one "trial".

- For example, suppose we have the following *data frame* `my.data`:

```
my.data

##    y    x
## 1  0 0.11
## 2  0 0.84
## 3  1 0.99
## 4  1 1.25
## 5  0 1.32
## 6  0 3.06
## 7  1 3.50
## 8  0 4.25
## 9  0 4.71
## 10 0 4.74
```

```
## 11 0 4.80
## 12 1 5.13
## 13 1 5.42
## 14 0 6.09
## 15 1 6.16
## 16 1 7.80
## 17 1 8.16
## 18 1 8.43
## 19 1 8.67
## 20 0 8.76
## 21 1 9.22
## 22 1 9.63
## 23 1 9.95
```

Note that `y` is ***dichotomous*** (0 or 1). Suppose we want to model `y` in terms of the predictor `x` using a ***logistic regression model***. We type:

```
my.logreg <- glm(y ~ x, family = binomial, data = my.data)
```

The object `my.logreg` belongs to the *glm* class of objects:

```
class(my.logreg)

## [1] "glm" "lm"
```

(and also belongs to the *lm* class – *glm* is a special case of *lm*). Typing:

```
names(my.logreg)

##  [1] "coefficients"      "residuals"         "fitted.values"
##  [4] "effects"           "R"                 "rank"
##  [7] "qr"                "family"            "linear.predictors"
## [10] "deviance"          "aic"               "null.deviance"
## [13] "iter"              "weights"           "prior.weights"
## [16] "df.residual"       "df.null"           "y"
## [19] "converged"         "boundary"          "model"
## [22] "call"              "formula"           "terms"
## [25] "data"              "offset"            "control"
## [28] "method"            "contrasts"         "xlevels"
```

shows that `my.logreg` contains objects `"coefficients"` and `"fitted.values"` like *lm* objects. (The `"residuals"` are so-called ***deviance residuals***, not the familiar residuals used in linear regression).

We can get a summary of the fit by typing:

```
summary(my.logreg)

##
## Call:
## glm(formula = y ~ x, family = binomial, data = my.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8012  -1.0833   0.5847   0.8529   1.6567
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.3962     0.9662  -1.445   0.1484
## x             0.3194     0.1669   1.914   0.0556
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 31.492  on 22  degrees of freedom
## Residual deviance: 27.059  on 21  degrees of freedom
## AIC: 31.059
##
## Number of Fisher Scoring iterations: 4
```

From the output, the ***estimates*** of the coefficients β_0 and β_1 in the ***logistic regression model***

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \beta_1 X_i$$

are $\boldsymbol{b_0 = -1.3962}$ and $\boldsymbol{b_1 = 0.3194}$.

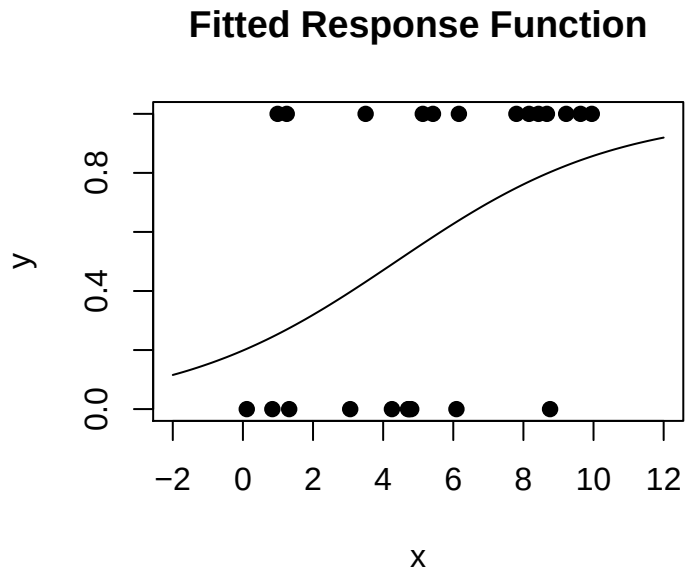
We can **estimate** probabilities

$$\pi_i = \frac{e^{\beta_0+\beta_1 X_i}}{1+e^{\beta_0+\beta_1 X_i}} \tag{1}$$

by replacing the unknown parameters β_0 and β_1 by their estimates b_0 and b_1.

We can plot the data and the fitted response function using `plot()` followed by `curve()` (with `add = TRUE`). For example:

```
plot(y ~ x, data = my.data, xlim = c(-2, 12), pch = 19,
     main = "Fitted Response Function")
curve(exp(-1.3962 + 0.3194*x)/(1 + exp(-1.3962 + 0.3194*x)),
      from = -2, to = 12, add = TRUE)
```



1.2 Poisson Regression in R

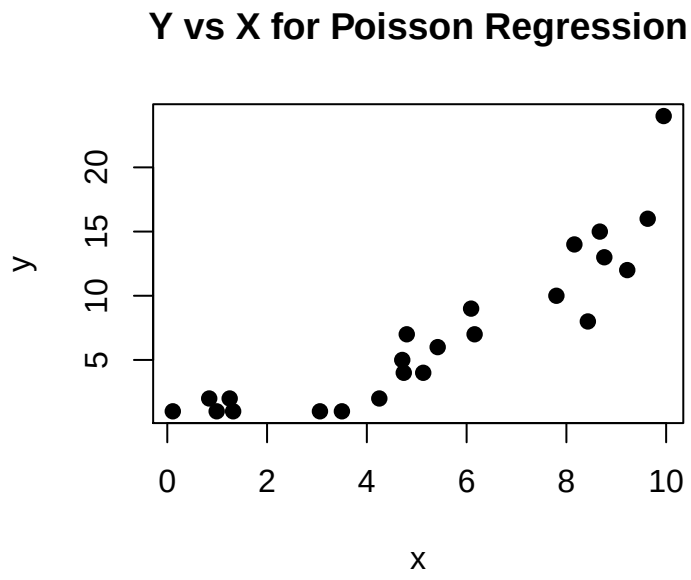
- For ***Poisson regression***, the response is a ***Poisson*** random variable whose mean $\boldsymbol{\mu} = \boldsymbol{\mu}(\boldsymbol{X})$ is a function of a predictor variable $\boldsymbol{X}$.
- For example, suppose we have the following *data frame* `my.data`:

```
my.data

##     y    x
## 1   1 0.11
## 2   2 0.84
## 3   1 0.99
## 4   2 1.25
## 5   1 1.32
## 6   1 3.06
## 7   1 3.50
## 8   2 4.25
## 9   5 4.71
## 10  4 4.74
## 11  7 4.80
## 12  4 5.13
## 13  6 5.42
## 14  9 6.09
## 15  7 6.16
## 16 10 7.80
## 17 14 8.16
```

```
## 18  8 8.43
## 19 15 8.67
## 20 13 8.76
## 21 12 9.22
## 22 16 9.63
## 23 24 9.95
```

```
plot(x, y, pch = 19, main = "Y vs X for Poisson Regression")
```



Note that `y` is a ***count***. Suppose we want to model `y` in terms of the predictor `x` using a ***Poisson regression model***. We type:

```
my.poisreg <- glm(y ~ x, family = poisson, data = my.data)
```

We can get a summary of the fit by typing:

```
summary(my.poisreg)

##
## Call:
## glm(formula = y ~ x, family = poisson, data = my.data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.35653  -0.43623  -0.06644   0.42955   1.24066
##
```

```
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 0.0008593  0.2676738   0.003    0.997
## x           0.2992558  0.0336476   8.894   <2e-16
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 113.934  on 22  degrees of freedom
## Residual deviance:  12.247  on 21  degrees of freedom
## AIC: 94.963
##
## Number of Fisher Scoring iterations: 4
```

From the output, the ***estimates*** of the coefficients β_0 and β_1 in the ***Poisson regression model***

$$\log(\mu_i) \;=\; \beta_0 + \beta_1 X_i$$

are $\boldsymbol{b_0 = 9 \times 10^{-4}}$ and $\boldsymbol{b_1 = 0.2993}$.

To add the fitted model to the plot from above, type:

```
plot(x, y, pch = 19, main = "Y vs X for Poisson Regression")
curve(exp(0.0008593 + 0.2992558*x),
      from = 0, to = 10, add = TRUE)
```