Estimating a Poisson Regression Model with R

Rémi Piatek

May 6, 2019

The R Programming Language

- `R`: Popular **open-source programming language** for statistical analysis.
- Widely used in statistics and econometrics.
- **User-friendly and powerful IDE** for `R`: RStudio.
- Basic functionalities of `R` can be extended by **packages**.
- Large number of packages available on the Comprehensive R Archive Network.
- **Goal of this presentation:** Illustrate how to use `R` for the estimation of a Poisson regression model.

Count Data Models

- **Count data** models are used to explain dependent variables that are natural numbers, i.e., positive integers such that $y_i \in \mathbb{N}$, where $\mathbb{N} = \{0, 1, 2, \ldots\}$.
- Count data models are frequently used in economics to study **countable events**: Number of years of education, number of patent applications filed by companies, number of doctor visits, number of crimes committed in a given city, etc.
- The **Poisson model** is a popular count data model.

Poisson Regression Model

- Given a parameter $\lambda_i > 0$, the **Poisson model** assumes that the probability of observing $Y_i = y_i$, where $y_i \in \mathbb{N}$, is equal to:

$$Prob(Y_i = y_i \mid \lambda_i) = \frac{\lambda_i^{y_i} \exp\{-\lambda_i\}}{y_i!},$$

  for $i = 1, \dots, N$.

- The mean and the variance of Y_i are equal to the parameter λ_i:

$$E(Y_i \mid \lambda_i) = V(Y_i \mid \lambda_i) = \lambda_i,$$

  implying *equi-dispersion* of the data.

- To control for **observed characteristics**, the parameter λ_i can be parametrized as follows (implying $\lambda_i > 0$):

$$E(Y_i|X_i, \beta) \equiv \lambda_i = \exp\{X_i' \beta\},$$

  where X_i is a vector containing the covariates.

Simulating Data

- R function simulating data from Poisson regression model:

```
simul_poisson <- function(n, beta) {
  k <- length(beta)                    # number of covariates
  x <- replicate(k - 1, rnorm(n)) # simulate covariates
  x <- cbind(1, x)                     # for intercept term
  lambda <- exp(x %*% beta)            # individual means
  y <- rpois(n, lambda)                # simulate count
  return(data.frame(y, x))             # return variables
}
```

- Using function to generate data:

```
set.seed(123)
nobs <- 1000
beta <- c(-.5, .4, -.7)
data <- simul_poisson(nobs, beta)
```

Data Description

- Descriptive statistics:

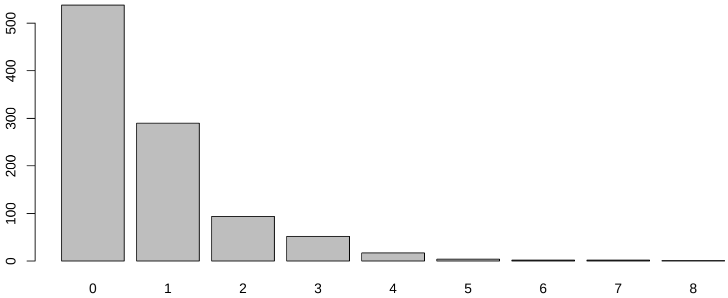
```r
# extract variables of interest from data set
y <- data[, 1]
x <- as.matrix(data[, 2:4])

# descriptive statistics
library(psych)
describe(data)
```

```
##    vars    n mean   sd median trimmed  mad   min  max ra
## y     1 1000 0.76 1.08   0.00    0.54 0.00  0.00 8.00  8
## X1    2 1000 1.00 0.00   1.00    1.00 0.00  1.00 1.00  0
## X2    3 1000 0.02 0.99   0.01    0.01 0.96 -2.81 3.24  6
## X3    4 1000 0.04 1.01   0.05    0.05 1.05 -3.05 3.39  6
##      se
## y  0.03
## X1 0.00
```

Data Description

- Histogram of count variable:

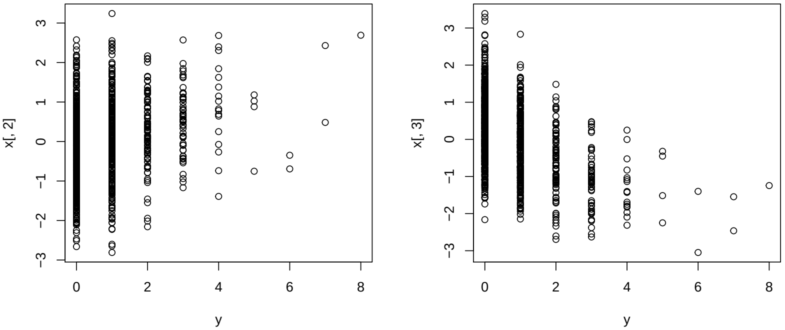
```
barplot(table(y))
```



Data Description

- Relationship between count variable and covariates:

```
par(mfrow = c(1, 2))
plot(y, x[, 2])
plot(y, x[, 3])
```



Likelihood Function and ML Estimator

- Individual contribution to the likelihood function:

$$L_i(\beta; y_i, x_i) = \frac{\exp\{y_i x_i \beta\} \exp\{-\exp\{x_i \beta\}\}}{y_i!}$$

- Individual log-Likelihood function:

$$\ell_i(\beta; y_i, x_i) = \log L_i(\beta; y_i, x_i) = y_i x_i \beta - \exp\{x_i \beta\} - \log(y_i!)$$

- Maximum Likelihood Estimator:

$$\hat{\beta}_{\text{MLE}} = \arg\max_{\beta} \sum_{i=1}^{N} \ell(\beta; y, X)$$

- Optimization (using *minimization* of objective function):

$$\hat{\beta}_{\text{MLE}} = \arg\min_{\beta} Q(\beta; y, X) \qquad Q(\beta; y, X) = -\frac{1}{N} \sum_{i=1}^{N} \ell_i(\beta; y_i, x_i)$$

Coding the Objective Function

```r
# Objective function of Poisson regression model
obj_poisson <- function(beta, y, x) {
  lambda <- x %*% beta
  llik <- y*lambda - exp(lambda) - lfactorial(y)
  return(-mean(llik))
}

# Evaluating objective function
beta0 <- c(1, 2, 3)
obj_poisson(beta0, y, x)
```

```
## [1] 1757.113
```

Maximizing the Objective Function

- Set starting values:

```
beta0 <- rep(0, length(beta))
```

- Optimize using quasi-Newton method (BFGS algorithm):

```
opt <- optim(beta0, obj_poisson, method = "BFGS",
             y = y, x = x)
```

- Show results:

```
cat("ML estimates:", opt$par,
    "\nObjective function:", opt$value, "\n")
```

```
## ML estimates: -0.5740286 0.3921569 -0.7231029
## Objective function: 0.9998689
```

Comparing Results to Built-in Function

```r
opt_glm <- glm(y ~ 0 + x, family = poisson)
summary(opt_glm)
```

```
##
## Call:
## glm(formula = y ~ 0 + x, family = poisson)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1707  -0.9185  -0.5915   0.5133   3.6768
##
## Coefficients:
##     Estimate Std. Error z value Pr(>|z|)
## xX1 -0.57403    0.04602  -12.47   <2e-16 ***
## xX2  0.39216    0.03690   10.63   <2e-16 ***
## xX3 -0.72310    0.03662  -19.74   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.
```

Comparing Results to Built-in Function

- Collect results from the two approaches to compare them:

```
res <- cbind("True" = beta, "MLE" = opt$par,
             "GLM" = opt_glm$coefficients)
row.names(res) <- c("constant", "x1", "x2")
res
```

```
##          True        MLE        GLM
## constant -0.5 -0.5740286 -0.5740289
## x1        0.4  0.3921569  0.3921561
## x2       -0.7 -0.7231029 -0.7231039
```

- **Question:** Our results (`MLE`) are virtually the same as those obtained with the built-in function `GLM`, but not identical. Where do the small differences come from?

Empirical Illustration

- Goal: Investigate the determinants of fertility.
- Poisson regression model used to estimate the relationship between explanatory variables and count outcome variable.
- Both our estimator coded from scratch and `R` built-in function will be used.

Data

- Source: Botswana's 1988 Demographic and Health Survey.
- Data set borrowed from Wooldridge:

```
library(wooldridge)
data(fertil2)
```

- Outcome variable: Total number of living children:

```
y_lab <- "children"
```

- Explanatory variables: Education, age, marital status, living in urban area, having electricity/TV at home:

```
x_lab <- c("educ", "age", "agesq", "evermarr", "urban",
           "electric", "tv")
```

Loading data

- Selecting variables and removing missing values:

```
data <- fertil2[, c(y_lab, x_lab)]
data <- na.omit(data)
```

- Show first 6 observations on first 8 variables:

```
head(data[, 1:8], n = 6)
```

```
##   children educ age agesq evermarr urban electric tv
## 1        0   12  24   576        0     1        1  1
## 2        3   13  32  1024        1     1        1  1
## 3        1    5  30   900        1     1        1  0
## 4        2    4  42  1764        1     1        1  1
## 5        2   11  43  1849        1     1        1  1
## 6        1    7  36  1296        1     1        1  0
```

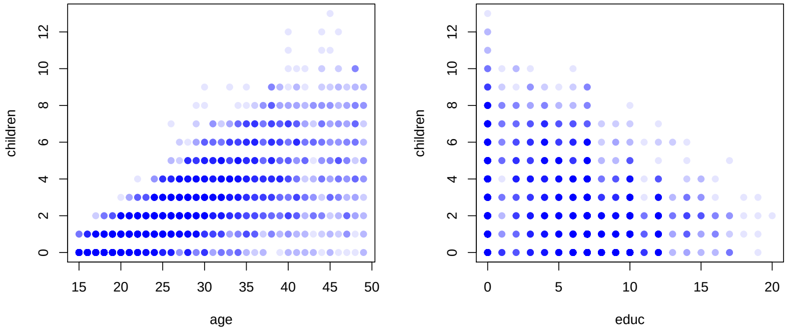
Descriptive Statitics

```r
library(psych)
describe(data)
```

```
##          vars    n   mean     sd median trimmed    mad m
## children    1 4358   2.27   2.22      2    1.95   2.97
## educ        2 4358   5.86   3.93      7    5.79   4.45
## age         3 4358  27.40   8.69     26   26.71   8.90
## agesq       4 4358 826.42 526.96    676  752.87 467.02 2
## evermarr    5 4358   0.48   0.50      0    0.47   0.00
## urban       6 4358   0.52   0.50      1    0.52   0.00
## electric    7 4358   0.14   0.35      0    0.05   0.00
## tv          8 4358   0.09   0.29      0    0.00   0.00
##           skew kurtosis   se
## children  1.07     0.75 0.03
## educ     -0.03    -0.50 0.06
## age       0.59    -0.49 0.13
## agesq     1.10     0.52 7.98
## evermarr  0.09    -1.99 0.01
```

Plot

```
attach(data)
par(mfrow = c(1, 2))
blue_transp <- adjustcolor("blue", alpha.f = 0.1)
plot(age, children, pch = 19, col = blue_transp)
plot(educ, children, pch = 19, col = blue_transp)
```



MLE of the Poisson Model

- Maximum likelihood function using built-in function `glm()`:

```
mle <- glm(children ~ educ + age + agesq + evermarr +
             urban + electric + tv,
           family = "poisson", data = data)
```

- Maximum likelihood function using our own function:

```
y <- data[, y_lab]
x <- as.matrix(data[, x_lab])
x <- cbind(1, x)           # for intercept term
beta0 <- rep(0, ncol(x))  # starting values
opt <- optim(beta0, obj_poisson, method = "BFGS",
             y = y, x = x)
```

MLE of the Poisson Model

- Results different from `glm()`?
- Optimization algorithms are iterative methods that rely on different criteria to dertermine if/when the optimum has been reached.
- For example: Change in the objective function, change in the parameter values, change in the gradient, step size, etc.
- *[More in Advanced Microeconometrics course].*
- Try to adjust tuning parameters, for example add `control = list(ndeps = rep(1e-8, ncol(x)))` to `optim()` to change step size of gradient approximation.

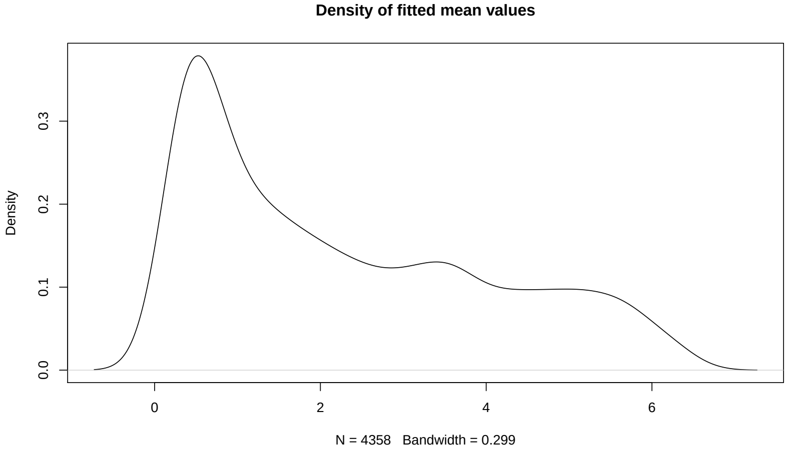
Summarizing the Empirical Results

```r
summary(mle)
```

```
##
## Call:
## glm(formula = children ~ educ + age + agesq + evermarr +
##     electric + tv, family = "poisson", data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5620  -0.8116  -0.1091   0.5439   2.8893
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.3748294  0.1628671 -33.001  < 2e-16 ***
## educ        -0.0216645  0.0029131  -7.437 1.03e-13 ***
## age          0.3373308  0.0099365  33.949  < 2e-16 ***
## agesq       -0.0041158  0.0001453 -28.331  < 2e-16 ***
## evermarr     0.3147510  0.0244473  12.875  < 2e-16 ***
```

Fitted Values

```
plot(density(mle$fitted.values),
     main = "Density of fitted mean values")
```



Formatting the results

```r
library(xtable)
xtable(mle)
```

```
## % latex table generated in R 3.6.0 by xtable 1.8-4 packa
## % Mon May  6 11:06:08 2019
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrrr}
##   \hline
##  & Estimate & Std. Error & z value & Pr($>$$|$z$|$) \\
##   \hline
## (Intercept) & -5.3748 & 0.1629 & -33.00 & 0.0000 \\
##   educ & -0.0217 & 0.0029 & -7.44 & 0.0000 \\
##   age & 0.3373 & 0.0099 & 33.95 & 0.0000 \\
##   agesq & -0.0041 & 0.0001 & -28.33 & 0.0000 \\
##   evermarr & 0.3148 & 0.0244 & 12.87 & 0.0000 \\
##   urban & -0.0861 & 0.0216 & -3.98 & 0.0001 \\
##   electric & -0.1205 & 0.0388 & -3.10 & 0.0019 \\
```