Exercises: Week 2

Econometrics Prof. Conlon

Ulrich Atz

2021-02-09

```
library(tidyverse)
set.seed(202102)
# Hashtag NOLOOPS
```

1. Let's load the Boston HMDA data (silently).

The function should take the following arguments:

- dir : debt to income ratio
- hir : housing to income ratio
- single : dummy for single borrower
- self : dummy for self-employed

```
library("Ecdat")
data("Hmda")

probit <- glm(deny ~ dir + hir + single + self, data = Hmda,   family = binomial(link = "probit"))
logit <- glm(deny ~ dir + hir + single + self, data = Hmda, family = binomial(link = "logit"))
# Bonus: linear probability model
lpm <- glm(as.numeric(deny) ~ dir + hir + single + self, data = Hmda)

models <- list(probit, logit, lpm)

map(models, margins::margins_summary)
```

```
## [[1]]
##     factor     AME     SE       z      p   lower  upper
##        dir  0.6013 0.0931  6.4616 0.0000  0.4189 0.7838
##        hir -0.0924 0.1081 -0.8547 0.3927 -0.3043 0.1195
##    selfyes  0.0405 0.0228  1.7820 0.0747 -0.0040 0.0851
##  singleyes  0.0468 0.0138  3.3959 0.0007  0.0198 0.0737
##
## [[2]]
##     factor     AME     SE       z      p   lower  upper
##        dir  0.6091 0.0911  6.6886 0.0000  0.4306 0.7876
##        hir -0.0747 0.1037 -0.7200 0.4715 -0.2780 0.1286
##    selfyes  0.0397 0.0227  1.7460 0.0808 -0.0049 0.0843
##  singleyes  0.0462 0.0138  3.3581 0.0008  0.0192 0.0732
##
## [[3]]
##     factor     AME     SE       z      p   lower  upper
##        dir  0.7344 0.0974  7.5365 0.0000  0.5434 0.9254
```

```
##        hir -0.2025 0.1082 -1.8711 0.0613 -0.4146 0.0096
##    selfyes  0.0445 0.0203  2.1869 0.0287  0.0046 0.0843
##  singleyes  0.0490 0.0133  3.6749 0.0002  0.0229 0.0751
```

2. Consider the regression model of the logit regression:

For a single observation compute the contribution to the log-likelihood (analytically)

$$deny_i = F(\beta_1 \cdot dir_i + \beta_2 \cdot hir_i + \beta_3 \cdot single_i + \beta_4 \cdot self_i)$$

The link for the logit is the logistic function: $F(X, \beta) = \frac{e^{X'\beta}}{1+e^{X'\beta}} = \frac{1}{1+e^{-X'\beta}}$

The log-likelihood of a single observation for the logit model is:

$$\ell_i(y_i|\beta) = y_i \ln(F(X_i, \beta)) + (1 - y_i)\ln(1 - F(X_i, \beta))$$

```r
# Get beta hats from regression output
vars <- c("dir", "hir", "single", "self")
# Get first observation and add constant
x_i <- matrix(c(1, as.numeric(Hmda[1, vars])), ncol = 1)
dv_i <- as.numeric(Hdma[1, "deny"])

# Model parameters
betas <- logit$coefficients
n <- nrow(Hmda)

# Logistic function
lgc <- function(x){ as.numeric(1 / (1 + exp(-x))) }
Fn <- log(lgc(t(x_i) %*% betas))

# Log-likelihood
llik_i <- function(dv_i, x_i, betas) {
  l = (dv_i * Fn) + (1 - dv_i) * log(1 - Fn)
  cat("The log-likelihood value for individual 1 is:", l)
  }

llik_i(dv_i, x_i, betas)
```

```
## The log-likelihood value for individual 1 is: -2.265814
```

3. For a single observation compute the Score (analytically).

The contribution of a single observation to the log-likelihood for the logit is the score i, where $f(X_i, \beta) = f(Z_i)$ is the derivative:

$$\mathcal{S}_i(X_i, \beta) = S_i(Z_i) = \frac{\partial \ln f(Z_i)}{\partial \beta} = \frac{y_i}{F(Z_i)}\frac{dF(Z_i)}{d\beta} - \frac{1 - y_i}{1 - F(Z_i)}\frac{dF(Z_i)}{d\beta}$$

This simplifies to:

$$\mathcal{S}_i(X_i, \beta) = (y_i - F(X_i, \beta))X_i = (y_i - \frac{1}{1 + e^{-X_i'\beta}})X_i$$

```r
score_i <- function(dv_i, x_i, .betas) {
  s = (dv_i - Fn) * x_i
  cat("The marginal log-likelihood value for individual 1 is:\n")
  matrix(s, dimnames = list(paste0("beta_", 0:4), "Score")) %>%
    round(3)
}

score_i(dv_i, x_i, betas)
```

```
## The marginal log-likelihood value for individual 1 is:
```

```
##        Score
## beta_0 3.266
## beta_1 0.722
## beta_2 0.722
## beta_3 3.266
## beta_4 3.266
```

4. Compute the Hessian Matrix and Fisher information (analytically).

For a single observation, we can take the derivative of the above score again to get to the Hessian: $\mathcal{H}_i = \frac{\partial \ell_i^2}{\partial \beta \partial \beta'} = -f(Z_i) X_i X_i^T$

We don't even need the derivative $f(Z_i)$ in this case because of a convenient relationship:

$$\mathcal{H}_i = \frac{\partial \ell_i^2}{\partial \beta \partial \beta'} = -[F(Z_i)(1 - F(Z_i))] \cdot X_i X_i^T$$

And the Fisher information:

$$\mathcal{I}(X_i, \beta) = \mathbb{E}_X[-\mathcal{H}_i(X_i, \beta)] = \mathbb{E}_X[\mathcal{S}_i(X_i, \beta) \cdot \mathcal{S}_i(X_i, \beta)^T]$$

```r
hessian_i <- function(x_i, .betas = betas) {
  Fn = lgc(x_i %*% .betas)
  h_i = (-1) * (Fn * (1 - Fn)) * x_i %*% t(x_i)
  cat("The Hessian matrix for individual 1 is:\n")
  h_i %>% signif(3)
}

hessian_i(x_i, betas)
```

```
## The Hessian matrix for individual 1 is:
```

```
##         [,1]      [,2]    [,3]    [,4]    [,5]
## [1,] -0.0152 -0.000485 -0.0482 -0.2380 -0.2420
## [2,] -0.0451 -0.007970 -0.0121 -0.0551 -0.0552
## [3,] -0.0451 -0.007970 -0.0121 -0.0551 -0.0552
## [4,] -0.0152 -0.000485 -0.0482 -0.2380 -0.2420
## [5,] -0.0152 -0.000485 -0.0482 -0.2380 -0.2420
```

5. Code up the Fisher Information for the logit model above $I(\widehat{\beta})$ using the Hessian Matrix.

```r
# Let's rewrite the function more for functional use
hessian_i <- function(..., .betas = betas) {
  x_i <- matrix(c(...), ncol = 1) # vector 5x1
```

```r
  Fn <- lgc(t(x_i) %*% .betas) # scalar
  h_i <- (-1) * (Fn * (1 - Fn)) * x_i %*% t(x_i)  # matrix 5x5
  h_i
}

fisher_info_hessian <- function(data = Hdma, .vars = vars){
  data <- tibble(constant = 1, select(data, all_of(.vars)))
  # That one missing value really is a pain
  data <- drop_na(data)
  n <- nrow(data)
  h_is <- pmap(data, hessian_i) # Compute each H_i
  fisher <- (-1) * reduce(h_is, `+`)
  fisher
}

fisher_info_hessian()
```

```
##          [,1]      [,2]      [,3]     [,4]     [,5]
## [1,] 392.5879 135.08701 102.64497 569.3615 442.8432
## [2,] 135.0870  48.89879  36.61466 196.0166 152.2277
## [3,] 102.6450  36.61466  28.90925 149.8937 115.0043
## [4,] 569.3615 196.01663 149.89372 922.9087 639.9783
## [5,] 442.8432 152.22769 115.00427 639.9783 543.3538
```

6. Code up the Fisher Information for the logit model above $I(\widehat{\beta})$ using the score method.

```r
score2_i <- function(..., .betas = betas) {
  dv_i <- as.numeric(c(...)[[1]]) # scalar
  x_i <- matrix(c(...)[-1], ncol = 1) # vector 5x1
  Fn <- lgc(t(x_i) %*% .betas) # scalar
  s_i = (dv_i - Fn) * x_i # vector 5x1
  s2_i = s_i %*% t(s_i) # matrix 5x5
  s2_i
}

fisher_info_score <- function(data = Hdma, .vars = vars){
  data <- tibble(dv = pull(data, deny), # NOT select!
                 constant = 1,
                 select(data, all_of(.vars)))
  data <- drop_na(data)
  n <- nrow(data)
  score2_is <- pmap(data, score2_i)
  fisher <- (1) * reduce(score2_is, `+`)
  fisher
}

fisher_info_score()
```

```
##           [,1]     [,2]     [,3]      [,4]      [,5]
## [1,] 2136.7789 704.7637 544.4629 2963.9428 2386.3504
## [2,]  704.7637 259.4052 199.1054  981.1720  788.0343
## [3,]  544.4629 199.1054 160.5987  759.5761  607.6166
## [4,] 2963.9428 981.1720 759.5761 4618.2705 3312.8256
## [5,] 2386.3504 788.0343 607.6166 3312.8256 2885.4932
```

Did we achieve the same?

```
identical(fisher_info_hessian(), fisher_info_score())
```

```
## [1] FALSE
```

```
# And from the R routine
solve(vcov(logit))
```

```
##               (Intercept)      dir       hir singleyes   selfyes
## (Intercept)    236.94072 84.01123 63.108246 113.03136 33.029905
## dir             84.01123 31.66041 23.371931  40.37221 11.891088
## hir             63.10825 23.37193 18.273705  30.88919  8.414693
## singleyes      113.03136 40.37221 30.889191 113.03136 14.324912
## selfyes         33.02991 11.89109  8.414693  14.32491 33.029905
```

7. Compute the standard errors from the Fisher information and compare them to the standard errors reported from the regression. How do they compare?

Standard errors from regression

```
model_se <- sqrt(diag(vcov(logit)))
model_se
```

```
## (Intercept)         dir         hir   singleyes     selfyes
##   0.2845441   0.9173672   1.0418215   0.1306350   0.1885064
```

Standard errors from the Fisher information matrix

```
fisher_se <- sqrt(diag(solve(fisher_info_hessian())))
fisher_se %>% set_names(names(model_se))
```

```
## (Intercept)         dir         hir   singleyes     selfyes
##   0.3310026   0.7902885   0.8582877   0.1018446   0.1517494
```

Are they the same?

```
identical(model_se, fisher_se)
```

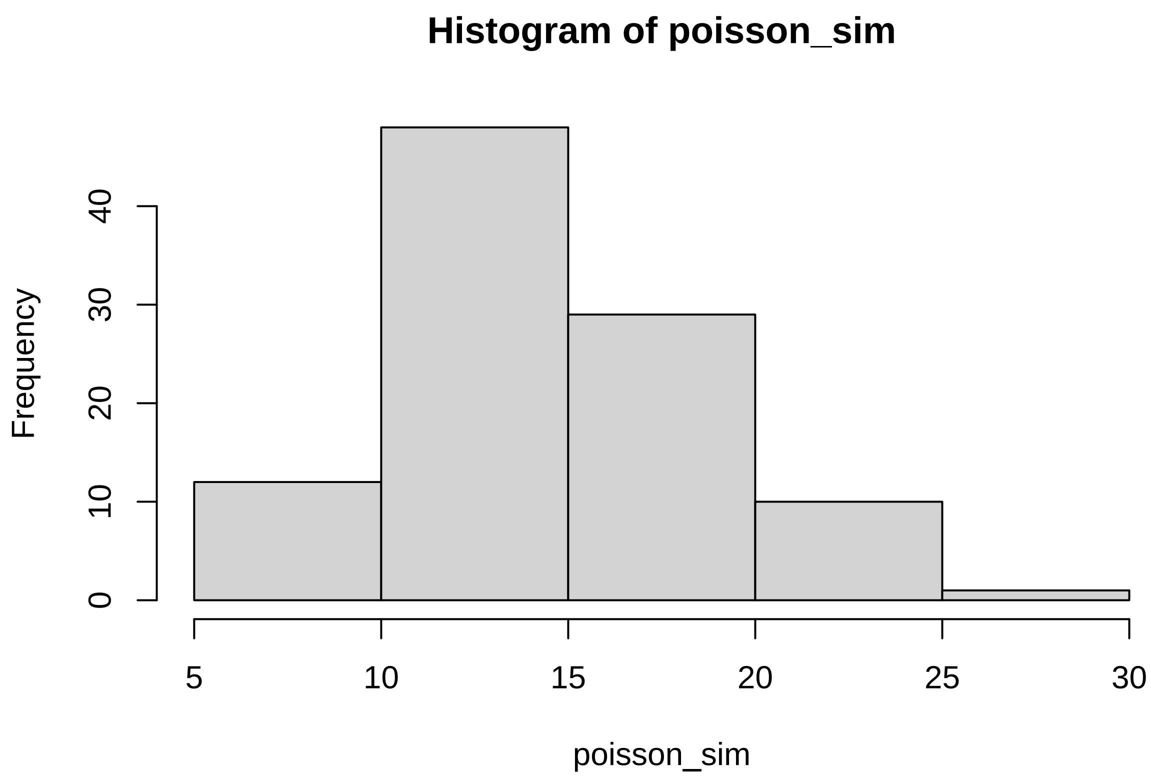
```
## [1] FALSE
```

8. Generate $n = 100$ observations where $\lambda = 15$ from a poisson model:

$$Y_i \sim Pois(\lambda)$$

```
poisson_sim <- function(n = 100, lambda = 15){
  x <- rpois(n, lambda)
}

poisson_sim <- poisson_sim()
hist(poisson_sim)
```



9. The poisson distribution is a discrete distribution for count data where the p.m.f. is given by:

$$Pr(Y_i = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

10. Write the log-likelihood $\ell(y_1, \dots, y_n; \lambda)$ (analytically).

The likelihood function is the product of the i.i.d. observations (here $n = 100$):

$$\mathcal{L}(\lambda|Y) = \prod_{i=1}^{n} \frac{\lambda^{y_i} e^{-\lambda}}{y_i!}$$

The log-likelihood function is the sum of the i.i.d. observations:

$$\ell(\lambda|Y) = \sum_{i=1}^{n} ln \frac{\lambda^{y_i} e^{-\lambda}}{y_i!} = \sum_{i=1}^{n} [-\lambda + y_i \cdot ln(\lambda) - ln(y_i!)]$$

11. Write the Score contribution $\mathcal{S}_i(y_i; \lambda)$ (analytically).

The score function is the first derivative of the log-likelihood:

$$\frac{\partial \ell_i(\lambda|y_i)}{\partial \lambda} = -1 + \frac{1}{\lambda} y_i$$

12. Write the Hessian Contribution $\mathcal{H}_i(y_i; \lambda)$ (analytically).

$$\mathcal{H}_i(y_i; \lambda) = \frac{\partial^2 \ell_i(\lambda|y_i)}{\partial^2 \lambda} = \frac{-1}{\lambda^2} y_i$$

13. Code up the log-likelihood function

```
pois_log_lik <- function(lambda, y = poisson_sim){
  s <- -lambda + y * log(lambda) - lfactorial(y)
  l <- sum(s)
  l
}

lambda = 15

pois_log_lik(lambda, poisson_sim)
```

```
## [1] -280.2924
```

14. Find the value of λ that maximizes your log likelihood using `optim` in R.

```
# Produce a parameter starting point
par <- runif(1) * 10 + 1

# Default minimizes
# SANN is best for integer problems?
out <- optim(par, pois_log_lik,
             method = "SANN",
             control = list(fnscale = -1))

# optimx::optimx(par, pois_log_lik,
#                 method = "BFGS",
#                 control = list(maximize = TRUE))


out
```

```
## $par
## [1] 15.1602
##
## $value
## [1] -280.2074
##
## $counts
## function gradient
##    10000       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
# Compare
mean(poisson_sim)
```

```
## [1] 15.16
```

15. Write a function that returns the standard error of $\hat{\lambda}$:

Note that $I(\widehat{\beta}) = nI_i(\widehat{\beta})$

```r
pois_se <- function(lambda_hat, y = poisson_sim){
  n <- length(y)
  info <- (-1) * sum(-1 / (lambda_hat)^2 * y)
  se <- sqrt(solve(info)) # matrix inversion for scalar
  se
}

lambda_hat <- out$par

pois_se(lambda_hat)
```

```
##           [,1]
## [1,] 0.3893635
```

```r
# Check with R functions
test <- glm(y ~ x,
            data = tibble(x = 1, y = poisson_sim),
            family = poisson(link = "identity"))

summary(test)$coef
```

```
##             Estimate Std. Error  z value Pr(>|z|)
## (Intercept)    15.16  0.3893584 38.93584        0
```