R Programming: Worksheet 7

By the end of today you should feel comfortable working with numerical optimization methods in `R`:

`optim()`, `nlm()`, `uniroot()`

1. **Maximum Likelihood**

   Suppose we have non-negative integer valued data $Y_1, \ldots, Y_n$, and an observed covariate $x_1, \ldots, x_n$. In a Poisson valued generalized linear model, we assume that

   $$Y_i \sim \text{Poisson}(\lambda_i)$$

   independently, where

   $$\log \lambda_i = \beta_0 + \beta_1 x_i.$$

   (a) Write down the log-likelihood for $\boldsymbol{\beta} = (\beta_0, \beta_1)^T$ given these data.

   (b) Write a function of three numeric vector arguments: `beta`, `y`, `x`, which returns **minus** the log-likelihood for the above model evaluated at $\boldsymbol{\beta} = (\beta_0, \beta_1)$. [Don't use `dpois()` for this.]

   (c) Let $n = 100$. Generate a covariate vector `x` as independent standard normal random variables. Use this to generate data from the above model with $\beta_0 = 1$, $\beta_1 = \frac{1}{2}$.

   The `R` function `optim()` performs generic minimization of functions. Its arguments are `par`, a vector of starting parameters (so in this case some starting value for $\boldsymbol{\beta}$), and `fn`, a function with first argument to be minimized over.

   (d) Use the function `optim()` to find the MLE for your dataset.

   (e) Check your answer by running the command

   ```
   > glm(y ~ x, family = poisson)
   ```

   [If you haven't seen GLMs before, you will do soon.]

   (f) Try doing the same thing as in (d) but using the function `nlm()` (which is similar to `optim()`).

   (g) * Give the ouput of `optim()` from (d) the class `optimum`. Write a print method for objects of this class which neatly displays (i) the optimal parameters, (ii) the value of the function at the optimum, and (iii) a suitable explanation of the error code (see `?optim`).

2. **Estimating Equations**

Consider a time series model

$$X_t = \phi X_{t-1} + \phi^2 X_{t-2} + \epsilon_t, \qquad t = 2, \ldots, T$$

where $X_0 = X_1 = 0$, $|\phi| < \frac{1}{2}$, and ϵ_t are independent, identically distributed random varibles with mean 0 and finite variance.

(a) Write a function with arguments T and ϕ, which generates a time series of the form above; have the errors be t_5-distributed.

(b) Generate some data with your function, using $\phi = 0.4$ and $T = 100$.

(c) Let

$$g(\eta, \boldsymbol{X}) = \frac{1}{T} \sum_{t=2}^{T} X_{t-1} \left(X_t - \eta X_{t-1} - \eta^2 X_{t-2} \right)$$

Prove that

$$\mathbb{E} X_t X_{t-1} = \phi \mathbb{E} X_{t-1}^2 + \phi^2 \mathbb{E} X_{t-1} X_{t-2}.$$

and deduce that $\mathbb{E} g(\phi, \boldsymbol{X}) = 0$.

If we don't know the distribution of the ϵ_ts (let's pretend we don't), we can't write down a likelihood for ϕ. However, we can find a **root** of the equation g: that is choose $\hat{\phi}_T$ such that $g(\hat{\phi}_T, \boldsymbol{X}) = 0$. This is called the method of *estimating equations*. Under reasonable conditions on the choice of g we find that $\hat{\phi}_T \to \phi$ as T grows.

(d) Write an `R` version of the function g, with first argument `phi`, and second `y`.

(e) Now solve the estimating equation: that is, find $\hat{\phi}$ such that $g(\hat{\phi}) = 0$. Use the function `uniroot()` .

(f) Observe that $\hat{\phi}_T$ is just the solution to a quadratic equation, and write a function to solve it exactly. [But note that it would be easy to construct an example without such an exact solution.]

(g) Generate a single large data set (sample size $n = 10^4$) and use the function from the previous part to find solutions to the estimating equation using the first 100, 300, 1000, 3000, and 10^4 observations.

Repeat this a large number of times (say 100), and comment on the accuracy of the estimates (of course the estimates improve, but how quickly?)

3. * **Violation of Modelling Assumptions**

(a) Write a function which takes a single integer `n`, and returns a list with entries `x` and `y`, where `x` is a vector of `n` independent uniform random variables on $[-1, 1]$,

$$y_i = x_i^2 + \varepsilon_i, \qquad i = 1, \ldots, n,$$

and $\varepsilon_i \overset{\text{i.i.d.}}{\sim} N(0, 1)$.

(b) Generate a sample of size 1,000 using the function from (a), and fit a linear model using the command `lm1 = lm(y ~ x)`. Look at the summary of your model output, as well as the diagnostic plots with `plot(lm1)`. What do you notice?

(c) Write a second function which generates `x` as before, but

$$y_i = x_i + \varepsilon_i, \qquad i = 1, \ldots, n,$$

where $\varepsilon_i \overset{\text{i.i.d.}}{\sim} t_3$. (Use the `rt()` function.)

(d) Repeat (b) with your new function.

(e) Write a function with argument `n` which generates a sample using the function from (c), fits a linear model, and then reports a 95% confidence interval for the coefficient of `x` (the slope).

(f) For a sample size $n = 10$, use the function from the previous part to generate $N = 1,000$ confidence intervals for different data sets. How many of them contain the 'true' value of the slope?

(g) Try increasing the sample size and repeating the previous part.