Andrew J. Spieker, PhD
BIOS 6342 - Contemporary Statistical Inference
Lab #3: Method of moments and maximum likelihood

Method of moments: The method of moments is a conceptually simple approach whereby sample moments (possibly bias-corrected but possibly not, and possibly central but possibly non-central) are equated to the theoretical moments, inducing an equation (or possibly a system of equations) through which an estimator can be formed.

Maximum likelihood: The method of maximum likelihood defines the estimator as the value of the parameter for which the observed data are "most probable," and I use quotes to signify that we may be maximizing a *density* function, in which the probability of any observation is zero. Typically in this class it happens that the estimate exists and is unique.

Setup: Suppose $X_1, \ldots, X_n \overset{\text{i.i.d.}}{\sim} \text{Beta}(\alpha, \beta)$, where $\alpha, \beta > 0$:

$$f_X(x; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} I_{(0,1)}(x).$$

Exercise: Consider estimation of α and β by the method of moments. Specifically, consider equating the sample mean, $\overline{X}_n$, to the theoretical first moment and equating the unbiased sample variance, S_n^2, to the second central moment. Write down the equations you would want to solve to determine $\overline{\alpha}_n$ and $\overline{\beta}_n$. Spend about three minutes trying to solve them before retching (proverbially), and then take for granted that the solution is given to you below (do the algebra later as an informal exercise if you're looking to flex your algebra muscles):

$$\overline{\alpha}_n = \overline{X}_n \left(\frac{\overline{X}_n(1-\overline{X}_n)}{S_n^2} - 1 \right) \quad \text{and} \quad \overline{\beta}_n = (1-\overline{X}_n)\left(\frac{\overline{X}_n(1-\overline{X}_n)}{S_n^2} - 1 \right).$$

Exercise: Now we consider the method of maximum likelihood. Determine the log-likelihood, and then present the score equations that could be used to solve for the MLEs, $\widehat{\alpha}_n$ and $\widehat{\beta}_n$. Now is probably the right time to present some (potentially new) notation for the *digamma* function:

$$\psi(\alpha) = \frac{d}{d\alpha} \log \Gamma(\alpha) = \frac{d\Gamma(\alpha)/d\alpha}{\Gamma(\alpha)}.$$

Do not attempt to reduce this expression down any further.

Exercise: The score equations have no closed-form solution. Blast! However, we can use computational tools. The `optim()` function in R can search for the minimum (the reason why we code the *negative* log-likelihood function), but requires an initializer. Well, we have our method of moments estimators! Code up functions for $\overline{\alpha}_n$ and $\overline{\beta}_n$.

```
## Use this space to code up functions for the method of moments estimators:
alphabar <- function(x) { }
betabar <- function(x) { }
```

The next code chunk is an R function for the (salient component of the) negative log-likelihood (note the use of `lgamma()` for convenience).

```r
## Log-likelihood for Beta(alpha, beta)
negloglik <- function(theta, x) {
  ## Parameters (must be in one vector as the initial argument)
  alpha <- theta[1]
  beta <- theta[2]

  ## Penalize if alpha or beta are out of bounds
  if (alpha <= 0 || beta <= 0) return(Inf)

  ## Sample size
  n <- length(x)

  ## Tems for the log-likelihood
  term1 <- n * lgamma(alpha + beta) - n * lgamma(alpha) - n * lgamma(beta)
  term2 <- (alpha - 1) * sum(log(x))
  term3 <- (beta - 1) * sum(log(1 - x))

  ## Return negative log-likelihood for minimization
  return( - (term1 + term2 + term3) )
}
```

Exercise: The next code chunk is the code to initialize based on method of moments and subsequently maximize the log-likelihood using the L-BFGS-B method (you can look up details if you're interested, but this algorithm in particular is designed to prevent the parameters from escaping a user-specified range; the algorithm is beyond the scope of this course).

```r
## Initialize
alpha0 <- alphabar(x)
beta0 <- betabar(x)

## Maximum likelihood
fit <- optim(par = c(alpha0, beta0), fn = negloglik, x = x,
             method = "L-BFGS-B", lower = c(1e-6, 1e-6))
```

Conduct a simulation to compare the variance of the MLEs to the MoM estimators. Use a sample size of $n = 1000$, use parameter values $\alpha = 3$ and $\beta = 4$, and conduct $M = 10{,}000$ simulations.

Fisher scoring: If you want the code to run faster, you could numerically solve the score equations using the known score and information rather than relying on an optimizer. The Fisher scoring update is as follows:

$$\boldsymbol{\theta}^{(j+1)} = \boldsymbol{\theta}^{(j)} + \frac{1}{n}\mathcal{I}^{-1}(\boldsymbol{\theta}^{(j)}) \cdot \left.\frac{\partial \ell(\boldsymbol{\theta}; \mathbf{x})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(j)}}.$$

You've already determined the score; here is the Fisher information:

$$\mathcal{I}(\boldsymbol{\theta}) = \begin{bmatrix} \psi_1(\alpha) - \psi_1(\alpha+\beta) & -\psi_1(\alpha+\beta) \\ -\psi_1(\alpha+\beta) & \psi_1(\beta) - \psi_1(\alpha+\beta) \end{bmatrix},$$

where $\psi_1(\alpha) = d^2 \log\Gamma(\alpha)/d\alpha^2$ is the *trigamma* function. *Note*: The `digamma()` and `trigamma()` functions are available in R.

Exercise: Code the Fisher scoring algorithm; again initialize using the method of moments estimators, and use the criterion of $||\boldsymbol{\theta}^{(j)} - \boldsymbol{\theta}^{(j-1)}||^2 < 10^{-6}$ to declare convergence. Compare the computation time to that of the `optim()` function by recreating the simulation study that you already completed and running it in conjunction with the `system.time()` function.

Exercise: I'm going to let you in a little secret (it's not actually a secret). When you initialize the Fisher scoring algorithm with the right kind of estimator, it is usually good enough to do *one* Fisher scoring update. This is called the one-step estimator. Although I haven't elaborated on what I mean by "the right kind of estimator" and what it means to be "good enough." try running the simulation again with this version of the algorithm; determine how much you improve the computation time. Is there a cost? I will elaborate in our unit on large sample theory.