Finding the MLEs for Model 2

This document covers multiple ways to find the MLE for Model 2 the conditional model from Lecture 04: Using likelihoods and Lecture 05: Using likelihoods to compare models. See Lecture 04 for more details about the set up of the model.

```r
library(tidyverse)
refs <- read_csv("data/04-refs.csv")
```

The likelihood is

$$\begin{aligned} Lik(p_{H|N}, p_{H|HBias}, p_{H|VBias}) = [(p_{H|N})^{25}(1-p_{H|N})^{23}(p_{H|HBias})^{8} \\ (1-p_{H|HBias})^{12}(p_{H|VBias})^{13}(1-p_{H|VBias})^{9}] \end{aligned}$$

The log-likelihood is

$$\begin{aligned} \log(Lik(p_{H|N}, p_{H|HBias}, p_{H|VBias})) &= 25\log(p_{H|N}) + 23\log(1-p_{H|N}) \\ &+ 8\log(p_{H|HBias}) + 12\log(1-p_{H|HBias}) \\ &+ 13\log(p_{H|VBias}) + 9\log(1-p_{H|VBias}) \end{aligned}$$

We would like to find the MLEs for $p_{H|N}, p_{H|HBias},$ and $p_{H|VBias}$.

Finding MLEs using graphs

We need to find the MLEs for three parameters, therefore we would need to visualize a 4-dimensional object to find the MLEs from a graph. Given the difficulty of this task and the lack of precision in the estimates from this approach, we should rely on other approaches to find the MLEs in this instance.

Finding MLEs using calculus

We can find the MLE for each parameter using the partial derivative of the log-likelihood with respect to each parameter.

To find the MLE for $p_{H|N}$:

$$\begin{aligned} \frac{\partial \log(Lik(p_{H|N}, p_{H|HBias}, p_{H|VBias}))}{\partial p_{H|N}} &= \frac{25}{p_{H|N}} - \frac{23}{1-p_{H|N}} = 0 \\ &\Rightarrow \frac{25}{p_{H|N}} = \frac{23}{1-p_{H|N}} \\ &\Rightarrow 23p_{H|N} = 25(1-p_{H|N}) \\ &\Rightarrow 48p_{H|N} = 25 \\ &\Rightarrow \hat{p}_{H|N} = \frac{25}{48} = 0.521 \end{aligned}$$

We can use a similar approach to find the MLEs for $p_{H|HBias}$ and $p_{H|VBias}$.

$$\hat{p}_{H|HBias} = \frac{8}{20} = 0.4$$

$$\hat{p}_{H|VBias} = \frac{13}{22} = 0.591$$

Finding the MLEs using R

We can write a function and do a grid search to find the values that maximize the log-likelihood.

```
maxloglik<- function(nvals){
  #nvals specifies the number of values
  phn <- seq(0, 1, length = nvals)
  phh <- seq(0, 1, length = nvals)
  phv <- seq(0, 1, length = nvals)

  loglik <- expand.grid(phn, phh, phv)
  colnames(loglik) <- c("phn", "phh", "phv")

  loglik <- loglik %>%
    mutate(loglik  = log(phn^25 * (1 - phn)^23 * phh^8 * (1 - phh)^12 *
                           phv^13 * (1 - phv)^9))

  loglik %>%
    arrange(desc(loglik)) %>%
    slice(1)
}
```

```
maxloglik(100)
```

```
##         phn       phh       phv    loglik
## 1 0.5252525 0.4040404 0.5858586 -61.57691
```

Depending on the number of parameters, it may be hard to conduct a granular enough search to find the exact values of the MLEs. Therefore, one could use the function above to conduct a crude search to find starting values for R's `optim` function. The function `optim` differs from `optimize` in that it can optimize over multiple parameter values (The `optimize` function can only optimize over a single parameter value).

```
# Function to calculate log-likelihood that will be used in the optim function
loglik <- function(params){
  phn <- params[1]
  phh <- params[2]
  phv <- params[3]

  log(phn^25 * (1 - phn)^23 * phh^8 * (1 - phh)^12 *
                           phv^13 * (1 - phv)^9)
}
```

```
# use manual search to get starting values
start_vals <- maxloglik(50) %>% select(-loglik)
```

```
# Use optim function in R to find the values to maximize the log-likelihood
#set fnscale = -1 to maximize (the default is minimize)
optim(par = start_vals, fn = loglik, control=list(fnscale=-1))
```

```
## $par
##       phn       phh       phv
## 0.5208272 0.4000361 0.5909793
```

```
##
## $value
## [1] -61.57319
##
## $counts
## function gradient
##       66       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```