An Application of Preference Models and the EM Algorithm to Ranked Choice Primary Polls

Lisa Wilson

Fall 2019

Background and Data

Ranked choice voting (RCV) is a voting system where voters rank candidates by preference, rather than voting for only one candidate. Different methods exist for eliminating candidates with the fewest first place votes and redistributing second, third, etc. votes until a winner is identified. RCV is used internationally in countries such as Ireland and at the state and local levels in 10 U.S. states. Maine will be the first U.S. state to use RCV in a general presidential election in 2020.

The following data[1] comes from a poll commissioned by FairVote, an electoral reform organization that advocates for the implementation of RCV nationwide. Between Septemeber 2 and 6, 1002 likely Democratic primary voters were surveyed. One question asked respondents to rank the then-top five candidates – former Vice President Joe Biden, Sen. Elizabeth Warren, Sen. Bernie Sanders, Sen. Kamala Harris, and Mayor Pete Buttigieg – from one to five. The table below shows the number of people in the sample who ranked each candidate at each of the five ranks. As a note, the total number of rankings for each candidate do not equal the overall sample size of 1002, likely indicating that not every respondent fully ranked all five candidates.

	Biden	Buttigieg	Harris	Sanders	Warren
1	331	80	100	200	291
2	180	110	180	230	271
3	120	230	220	160	200
4	150	240	250	130	130
5	180	261	160	220	50
Total	961	921	910	940	942

Mixture model for preference data

Following D'Elia and Piccolo (2005)[2], a mixture of a discrete uniform and a shifted binomial, known as an MUB distribution, can be used to model ranked preference data. The MUB probability mass function is as follows:

$$p_r(\pi, \xi) = P(R = r) = \pi \binom{m-1}{r-1} (1-\xi)^{r-1} \xi^{m-r} + (1-\pi)\frac{1}{m} \tag{1}$$

where $\binom{m-1}{r-1}(1-\xi)^{r-1}\xi^{m-r}$ is a shifted binomial distribution (more detail in D'Elia (2000)[3]). r is the rank given by a respondent to a given option among m total options, $r = 1, 2, ..., m$, and ξ measures "positive feeling" toward the given option, $\xi \in [0, 1]$. As ξ increases, the option is more likely to be ranked higher. The shifted binomial portion incorporates the degree to which respondents like a given option in relation to the other options.

$\frac{1}{m}$ is a discrete uniform distribution where m is again the total number of options. The discrete uniform portion incorporates the uncertainty involved in the ranking process; for example, the uncertainty a respondent might feel about choosing between two rankings for an option.

Finally, π is the mixture probability, $\pi \in [0, 1]$. As π goes to 0, we have a case of total uncertainty, where respondents have no preference among the options, and as π goes to 1, there is no uncertainty, with the ranking process following the shifted binomial distribution.

EM algorithm

To estimate π and ξ from the MUB model, the EM algorithm can be used.

Letting $\theta = (\pi, \xi)^{'}$, the log-likelihood of the observed data for a particular candidate from the FairVote poll is

$$\log\ L(\theta^{(k)}) = \sum_{r=1}^{m} n_r \log(p_r(\theta^{(k)})) \tag{2}$$

with n_r being the number of respondents who gave the particular candidate a rank of r. This is the incomplete data log-likelihood because we do not know the appropriate mixture component for each respondent; i.e., we do not know how much the shifted binomial vs. the discrete uniform should be weighted for each respondent. Letting $Z_g, g = 1, 2$, be indicator variables for which distribution the respondent's preferences are from, the complete data log-likelihood for a particular candidate is

$$\log\ L(\theta) = \sum_{g=1}^{2} \sum_{r=1}^{m} z_g n_r [\log(\pi_g) + \log(p_g(r, \xi_g))] \tag{3}$$

where $g = 1$ is associated with the shifted binomial distribution (i.e., $p_1(r, \xi_1^{(k)}) = \binom{m-1}{r-1}(1 - \xi^{(k)})^{r-1}\xi^{(k)^{m-r}}$ and $\pi_1^{(k)} = \pi^{(k)}$) and $g = 2$ is associated with the discrete uniform distribution (i.e., $p_2(r, \xi_2^{(k)}) = \frac{1}{m}$ and $\pi_2^{(k)} = 1 - \pi^{(k)}$).

E-step

Z_g are Bernoulli random variables with expectation

$$E(Z_g \mid \mathbf{r}, \theta^{(k)}) = \tau_g(r, \theta^{(k)}) = \frac{\pi_g^{(k)} p_g(r, \xi_g^{(k)})}{\pi_1^{(k)} p_1(r, \xi_1^{(k)}) + \pi_2^{(k)} p_2(r)} \tag{4}$$

Then the expectation of the complete data log-likelihood at the k-th iteration is

$$E(\log\ L(\theta^{(k)})) = \sum_{g=1}^{2} \sum_{r=1}^{m} \tau_g(r, \theta^{(k)}) n_r [\log(\pi_g^{(k)}) + \log(p_g(r, \xi_g^{(k)}))] \tag{5}$$

M-step

Maximum likelihood estimates of π and ξ are obtained by maximizing the following function:

$$Q(\theta^{(k)}) = \sum_{r=1}^{m} n_r [\tau_1(r, \theta^{(k)}) \log(\pi^{(k)}) + \tau_2(r, \theta^{(k)}) \log(1 - \pi^{(k)})] + \sum_{r=1}^{m} n_r [\tau_1(r, \theta^{(k)}) \log[(p_1(r, \xi^{(k)}))] + \tau_2(r, \theta^{(k)}) \log[p_2(r)]] \tag{6}$$

This results in explicit formulas[4] for the updated estimates of π and ξ:

$$\pi^{(k+1)} = \frac{1}{N}\sum_{r=1}^{m} n_r \tau_1(r, \theta^{(k)}) = \frac{1}{N}\sum_{r=1}^{m} n_r \frac{\pi^{(k)}\binom{m-1}{r-1}(1-\xi^{(k)})^{r-1}\xi^{(k)^{m-r}}}{\pi^{(k)}\binom{m-1}{r-1}(1-\xi^{(k)})^{r-1}\xi^{(k)^{m-r}} + (1-\pi^{(k)})\frac{1}{m}} \tag{7}$$

$$\xi^{(k+1)} = \frac{m - \frac{\sum_{r=1}^{m} r\ n_r \tau_1(r,\theta^{(k)})}{\sum_{r=1}^{m} n_r \tau_1(r,\theta^{(k)})}}{m-1} = \frac{m - \bar{R}_n(\theta^{(k)})}{m-1} \tag{8}$$

This process repeats until the estimates converge. As suggested by D'Elia and Piccolo (2005), starting values of $\theta^{(0)} = (\pi^{(0)}, \xi^{(0)})' = (0.5, \frac{m-\bar{R}_n}{m-1})'$ were used, where $\bar{R}_n = \frac{1}{N}\sum_{r=1}^{m} r n_r$, $m = 5$, and $N = 1002$.

Estimates of ξ and π for each candidate

Candidate	$\hat{\xi}$	$\hat{\pi}$	$\frac{1-\hat{\pi}}{5}$	Iterations	$P(R=1)$
Biden	0.9812	0.1796	0.1641	269	0.3305
Buttigieg	0.2167	0.2918	0.1416	157	0.1423
Harris	0.3618	0.1452	0.171	296	0.1734
Sanders	0.856	0.0265	0.1947	583	0.2089
Warren	0.7946	0.414	0.1172	125	0.2822

The table above shows the estimates $\hat{\xi}$ and $\hat{\pi}$ obtained from the EM algorithm. A greater $\hat{\xi}$ indicates that the candidate is more likely to be ranked higher. Based on these results, Biden and Sanders were more likely to be ranked toward the top while Buttigieg was more likely to be ranked near the bottom of the pack. The third column shows the uncertainty share $\frac{1-\hat{\pi}}{5}$, which estimates the discrete uniform portion of the mixture model. A greater uncertainty share indicates there is more uncertainty among respondents about where to rank the candidate; Sanders has the highest uncertainty share and Warren has the lowest. The fourth column shows the number of iterations required for the EM algorithm to converge for each candidate.

The fifth column shows the predicted probability that each candidate would be ranked first. These were obtained by plugging $\hat{\xi}$ and $\hat{\pi}$ for each candidate into the MUB probability mass function (equation 1) for $r = 1$. Biden is most likely to be ranked first, with a predicted probability of 0.3305, followed by Warren, Sanders, Harris, and lastly Buttigieg. Biden's relatively high probability of being ranked first follows from his high $\hat{\xi}$ value. While Sanders's $\hat{\xi}$ is higher than Warren's (0.856 for Sanders vs. 0.7946 for Warren), Warren's uncertainty share is lower than Sanders's (0.1172 for Warren vs. 0.1947 for Sanders), which contributes to Warren's predicted probability of being ranked first being higher than for Sanders.

Bootstrap confidence intervals for ξ and π

Candidate	$\hat{\xi}$	Lower bound: ξ	Upper bound: ξ	$\hat{\pi}$	Lower bound: π	Upper bound: π
Biden	0.9812	0.9245	1.9623*	0.1796	−0.2558*	0.3592
Buttigieg	0.2167	−0.6045*	0.4334	0.2918	−0.0314*	0.5837
Harris	0.3618	−0.3142*	0.7237	0.1452	−0.3246*	0.2905
Sanders	0.856	0.6741	1.712*	0.0265	−0.562*	0.0531
Warren	0.7946	0.5513	1.5892*	0.414	0.2063	0.8281

The table above gives the 95% basic bootstrap confidence intervals for ξ and π for each candidate, as obtained from the `boot` package in `R`. Upper bounds above 1 and lower bounds below 0 are marked with an asterisk to indicate they are invalid for these proportion parameters.

Conclusion

As illustrated, the MUB model allows the different factors that contribute to a person's ranking decision to be teased apart, and the EM algorithm provides a way to estimate the probability parameters from this model. In the ranked choice poll application, this analysis shows how the "positive feeling" parameter and uncertainty share varies among the top-five Democratic primary candidates.

References

1. "FAIR0001 Toplines." FairVote. Accessed November 11, 2019. https://fairvote.app.box.com/v/2020-FV-YouGov-Toplines. [Relevant table on page 5 of pdf.]
2. D'Elia, Angela and Domenico Piccolo. "A mixture model for preferences data analysis." *Computational Statistics & Data Analysis* 49 (2005): 917-934.
3. D'Elia, Angela. "A shifted binomial model for rankings." In Nunez-Anton, V. and E. Ferreira (eds.), *Proceedings of the 15th International Workshop on Statistical Modelling: Servicio Editorial de la Universidad del Pais Vasco*, 412-416. Bilbao.
4. Piccolo, Domenico. "Observed information matrix for MUB models." *Quaderni di Statistica* 8 (2006): 33-78.

Code appendix

```
library(boot)
library(bootstrap)
library(tidyverse)
library(pander)
library(kableExtra)

# from page 5 of "FAIR0001 Toplines"
# https://fairvote.app.box.com/v/2020-FV-YouGov-Toplines
rcv_perc <- data.frame(c(.33, .18, .12, .15, .18),
                       c(.08, .11, .23, .24, .26),
                       c(.1, .18, .22, .25, .16),
                       c(.2, .23, .16, .13, .22),
                       c(.29, .27, .2, .13, .05))
colnames(rcv_perc) <- c("Biden", "Buttigieg", "Harris", "Sanders", "Warren")
# neither columns (candidates) nor rows (ranks) sum to 1
# --> people didn't use all 5 ranks
# colSums(rcv_perc)
# rowSums(rcv_perc)

# convert percentages to frequencies
rcv_freq <- round(1002*rcv_perc, 0)
# rcv_freq/1002

rcv_table <- rbind(rcv_freq, colSums(rcv_freq))
rownames(rcv_table) <- c("1", "2", "3", "4", "5", "Total")
kable(rcv_table, "latex", booktabs = TRUE, row.names = TRUE) %>%
  kable_styling(font_size = 10)

# shifted binomial pmf
pbinom.shift <- function(m, r, psi) {
    choose(m-1, r-1)*((1-psi)^(r-1))*psi^(m-r)
}

# EM algorithm function for frequency data
EM_rcv_freq <- function(current, data, m, n){
    r <- seq(1, m, 1)
    freq <- data
    pi <- current[1]
    psi <- current[2]

    # equivalent expression for tau_1 as in report
    tau <- (1 + (1-pi)/(m*pi*pbinom.shift(m, r, psi)))^(-1)

    R_n <- sum(r*freq*tau)/sum(freq*tau)

    pi_new <- (1/n)*sum(freq*tau)

    psi_new <- (m - R_n)/(m - 1)

    Out <- c(pi_new, psi_new)
}
```

```r
# MUB pmf
pMUB <- function(pi, psi, m, r){
    pi*(choose(m-1, r-1)*((1-psi)^(r-1))*psi^(m-r)) + (1-pi)*(1/m)
}

N <- 1002
tol <- 1E-10

# Joseph Biden
JB_freq <- rcv_freq[,1]

Rn_JB <- (seq(1, 5, 1)%*%JB_freq)/N
psi_JB <- (5-Rn_JB)/(5-1)

theta0 <- c(0.5, psi_JB) # starting values
theta_JB_f <- matrix(data = rep(0, 2000), nrow = 1000, ncol = 2) # store the values of theta
theta_JB_f[1,] <- theta0 # put starting value in theta vector

for(i in 2:1000){
    theta_JB_f[i,] = EM_rcv_freq(theta_JB_f[i-1, ], JB_freq, 5, N)
    if(abs(theta_JB_f[i,1]-theta_JB_f[i-1,1]) < tol & abs(theta_JB_f[i,2]-theta_JB_f[i-1,2]) < tol){
        est_JB <- theta_JB_f[i,]
        iter_JB <- i
        break
    }
}

# uncertainty share
unc_JB <- (1-est_JB[1])/5

# predicted probability of rank 1
r1_JB <- pMUB(est_JB[1], est_JB[2], 5, 1)

# good: probabilities of ranks sum to 1
# sum(pMUB(est_JB[1], est_JB[2], 5, seq(1,5,1)))

# Peter Buttigieg
PB_freq <- rcv_freq[,2]

Rn_PB <- (seq(1, 5, 1)%*%PB_freq)/N
psi_PB <- (5-Rn_PB)/(5-1)

theta0_PB <- c(0.5, psi_PB) # starting values
theta_PB_f <- matrix(data = rep(0, 2000), nrow = 1000, ncol = 2) # store the values of theta
theta_PB_f[1,] <- theta0_PB # put starting value in theta vector

for(i in 2:1000){
    theta_PB_f[i,] = EM_rcv_freq(theta_PB_f[i-1, ], PB_freq, 5, N)
    if(abs(theta_PB_f[i,1]-theta_PB_f[i-1,1]) < tol & abs(theta_PB_f[i,2]-theta_PB_f[i-1,2]) < tol){
        est_PB <- theta_PB_f[i,]
        iter_PB <- i
        break
    }
```

```
}

unc_PB <- (1-est_PB[1])/5

r1_PB <- pMUB(est_PB[1], est_PB[2], 5, 1)

# Kamala Harris
KH_freq <- rcv_freq[,3]

Rn_KH <- (seq(1, 5, 1)%*%KH_freq)/N
psi_KH <- (5-Rn_KH)/(5-1)

theta0_KH <- c(0.5, psi_KH) # starting values
theta_KH_f <- matrix(data = rep(0, 2000), nrow = 1000, ncol = 2) # store the values of theta
theta_KH_f[1,] <- theta0_KH # put starting value in theta vector

for(i in 2:1000){
    theta_KH_f[i,] = EM_rcv_freq(theta_KH_f[i-1, ], KH_freq, 5, N)
    if(abs(theta_KH_f[i,1]-theta_KH_f[i-1,1]) < tol & abs(theta_KH_f[i,2]-theta_KH_f[i-1,2]) < tol){
        est_KH <- theta_KH_f[i,]
        iter_KH <- i
        break
    }
}

unc_KH <- (1-est_KH[1])/5

r1_KH <- pMUB(est_KH[1], est_KH[2], 5, 1)

# Bernard Sanders
BS_freq <- rcv_freq[,4]

Rn_BS <- (seq(1, 5, 1)%*%BS_freq)/N
psi_BS <- (5-Rn_BS)/(5-1)

theta0_BS <- c(0.5, psi_BS) # starting values
theta_BS_f <- matrix(data = rep(0, 2000), nrow = 1000, ncol = 2) # store the values of theta
theta_BS_f[1,] <- theta0_BS # put starting value in theta vector

for(i in 2:1000){
    theta_BS_f[i,] = EM_rcv_freq(theta_BS_f[i-1, ], BS_freq, 5, N)
    if(abs(theta_BS_f[i,1]-theta_BS_f[i-1,1]) < tol & abs(theta_BS_f[i,2]-theta_BS_f[i-1,2]) < tol){
        est_BS <- theta_BS_f[i,]
        iter_BS <- i
        break
    }
}

unc_BS <- (1-est_BS[1])/5

# all ranks around 20%
# --> low psi indicates not much preference for 1st
r1_BS <- pMUB(est_BS[1], est_BS[2], 5, 1)
```

```r
# Elizabeth Warren
EW_freq <- rcv_freq[,5]

Rn_EW <- (seq(1, 5, 1)%*%EW_freq)/N
psi_EW <- (5-Rn_EW)/(5-1)

theta0_EW <- c(0.5, psi_EW) # starting values
theta_EW_f <- matrix(data = rep(0, 2000), nrow = 1000, ncol = 2) # store the values of theta
theta_EW_f[1,] <- theta0_EW # put starting value in theta vector

for(i in 2:1000){
    theta_EW_f[i,] = EM_rcv_freq(theta_EW_f[i-1, ], EW_freq, 5, N)
    if(abs(theta_EW_f[i,1]-theta_EW_f[i-1,1]) < tol & abs(theta_EW_f[i,2]-theta_EW_f[i-1,2]) < tol){
        est_EW <- theta_EW_f[i,]
        iter_EW <- i
        break
    }
}

unc_EW <- (1-est_EW[1])/5

r1_EW <- pMUB(est_EW[1], est_EW[2], 5, 1)

# rank sum is not bounded by 1
# r1_JB + r1_PB + r1_KH + r1_BS + r1_EW

### boot function
# EM algorithm function adapted for boot function
EM_rcv_bs2 <- function(data, vec, ranktot, n, indices){
  for(i in 2:1000){
    vec[i,] = EM_rcv_freq(vec[i-1,], data[indices], 5, N)
    if(abs(vec[i,1]-vec[i-1,1]) < tol & abs(vec[i,2]-vec[i-1,2]) < tol){
      est <- vec[i,]
      iter <- i
      break
    }
  }
  return(est)
}

# Joe Biden
theta0_JB <- c(0.5, psi_JB) # starting values
theta_JB_f <- matrix(data = rep(0, 2000), nrow = 1000, ncol = 2) # store the values of theta
theta_JB_f[1,] <- theta0 # put starting value in theta vector

set.seed(42)
JB_boot <- boot(JB_freq, EM_rcv_bs2, R=1000, vec=theta_JB_f, ranktot=5, n=N)

JB_basic_lpi <- boot.ci(JB_boot, index = 1)$basic[4]
JB_basic_upi <- boot.ci(JB_boot, index = 1)$basic[5]
JB_basic_lxi <- boot.ci(JB_boot, index = 2)$basic[4]
JB_basic_uxi <- boot.ci(JB_boot, index = 2)$basic[5]
# est_JB
```

```r
# bootstrap(x=JB_freq, nboot=1000, theta=EM_rcv_bs, current=theta0_JB, ranktot=5, n=N)

# Pete Buttigieg
theta0_PB <- c(0.5, psi_PB) # starting values
theta_PB_f <- matrix(data = rep(0, 2000), nrow = 1000, ncol = 2) # store the values of theta
theta_PB_f[1,] <- theta0_PB # put starting value in theta vector

set.seed(42)
PB_boot <- boot(PB_freq, EM_rcv_bs2, R=1000, vec=theta_PB_f, ranktot=5, n=N)

PB_basic_lpi <- boot.ci(PB_boot, index = 1)$basic[4]
PB_basic_upi <- boot.ci(PB_boot, index = 1)$basic[5]
PB_basic_lxi <- boot.ci(PB_boot, index = 2)$basic[4]
PB_basic_uxi <- boot.ci(PB_boot, index = 2)$basic[5]
# est_PB

# Kamala Harris
theta0_KH <- c(0.5, psi_KH) # starting values
theta_KH_f <- matrix(data = rep(0, 2000), nrow = 1000, ncol = 2) # store the values of theta
theta_KH_f[1,] <- theta0_KH # put starting value in theta vector

set.seed(42)
KH_boot <- boot(KH_freq, EM_rcv_bs2, R=1000, vec=theta_KH_f, ranktot=5, n=N)

KH_basic_lpi <- boot.ci(KH_boot, index = 1)$basic[4]
KH_basic_upi <- boot.ci(KH_boot, index = 1)$basic[5]
KH_basic_lxi <- boot.ci(KH_boot, index = 2)$basic[4]
KH_basic_uxi <- boot.ci(KH_boot, index = 2)$basic[5]
# est_KH

# Bernie Sanders
theta0_BS <- c(0.5, psi_BS) # starting values
theta_BS_f <- matrix(data = rep(0, 2000), nrow = 1000, ncol = 2) # store the values of theta
theta_BS_f[1,] <- theta0_BS # put starting value in theta vector

set.seed(42)
BS_boot <- boot(BS_freq, EM_rcv_bs2, R=1000, vec=theta_BS_f, ranktot=5, n=N)

BS_basic_lpi <- boot.ci(BS_boot, index = 1)$basic[4]
BS_basic_upi <- boot.ci(BS_boot, index = 1)$basic[5]
BS_basic_lxi <- boot.ci(BS_boot, index = 2)$basic[4]
BS_basic_uxi <- boot.ci(BS_boot, index = 2)$basic[5]
# est_BS

# Elizabeth Warren
theta0_EW <- c(0.5, psi_EW) # starting values
theta_EW_f <- matrix(data = rep(0, 2000), nrow = 1000, ncol = 2) # store the values of theta
theta_EW_f[1,] <- theta0_EW # put starting value in theta vector

set.seed(42)
EW_boot <- boot(EW_freq, EM_rcv_bs2, R=1000, vec=theta_EW_f, ranktot=5, n=N)

EW_basic_lpi <- boot.ci(EW_boot, index = 1)$basic[4]
```

```r
EW_basic_upi <- boot.ci(EW_boot, index = 1)$basic[5]
EW_basic_lxi <- boot.ci(EW_boot, index = 2)$basic[4]
EW_basic_uxi <- boot.ci(EW_boot, index = 2)$basic[5]
# est_EW

bootCI <- c(JB_basic_lpi, JB_basic_upi, JB_basic_lxi, JB_basic_uxi,
            PB_basic_lpi, PB_basic_upi, PB_basic_lxi, PB_basic_uxi,
            KH_basic_lpi, KH_basic_upi, KH_basic_lxi, KH_basic_uxi,
            BS_basic_lpi, BS_basic_upi, BS_basic_lxi, BS_basic_uxi,
            EW_basic_lpi, EW_basic_upi, EW_basic_lxi, EW_basic_uxi)
```