Estimate Parameters for Discrete Distributions

Fan Wang

2022-07-23

Contents

1 Binomial Probability Function Given Observed Discrete Probabilities

Go to the **RMD**, **R**, **PDF**, or **HTML** version of this file. Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

1.1 Estimate Change of Success given Fixed Number of Trials

We observe $\{P(x_i)\}_{i=0}^{N}$, where $\sum_{i=0}^{N} x_i = 1$. Note this is observing probabilities not at N outcomes but $N+1$ outcomes starting at $i = 0$. What is the binomial probability function that would fit the observed data the best?

There are two parameters of the binomial distribution, N, which is the "number of trials", and θ, which is the probability of success for each trial. In this section, we treat N as already observed, so we only need to pick the θ that provides the best fit between the binomial probabilities generated (given p), and the data.

So we fit with a likelihood based approach. First, the likelihood function is:

$$\mathcal{L}(\theta) = \Pi_{i=0}^{N} \left(\frac{N!}{(N-x_i)!x_i!} \cdot \theta^{x_i} \cdot (1-\theta)^{N-x_i} \right)^{P(x_i)}$$

Second, we write the log likelihood. Note that we observe $P(x_i)$ from the data.

$$\begin{aligned}
\ln\left(\mathcal{L}(\theta)\right) &= \sum_{i=0}^{N} P(x_i) \ln\left(\frac{N!}{(N-x_i)!x_i!} \cdot \theta^{x_i} \cdot (1-\theta)^{N-x_i} \right) \\
&= \sum_{i=0}^{N} \left(P(x_i) \ln\left(\frac{N!}{(N-x_i)!x_i!} \right) + P(x_i) x_i \ln(\theta) + P(x_i)(N-x_i) \ln(1-\theta) \right)
\end{aligned}$$

Third, we take the derivative of the log likelihood with respect to the parameter of interest θ.

$$\frac{d \ln\left(\mathcal{L}(\theta)\right)}{d\theta} = \sum_{i=0}^{N} \left(\frac{P(x_i)x_i}{\theta} - \frac{P(x_i)(N-x_i)}{1-\theta} \right)$$

Fourth, we set the derivative above to be equal to zero.

$$\frac{\sum_{i=0}^{N} P(x_i)x_i}{\theta} = \frac{\sum_{i=0}^{N} P(x_i)(N - x_i)}{1 - \theta}$$

$$\sum_{i=0}^{N} P(x_i)x_i - \theta \cdot \sum_{i=0}^{N} P(x_i)x_i = \theta \cdot \sum_{i=0}^{N} P(x_i)(N - x_i)$$

$$\theta = \frac{\sum_{i=0}^{N} P(x_i)x_i}{\sum_{i=0}^{N} (P(x_i)(N - x_i) + P(x_i)x_i)}$$

$$\theta = \frac{\sum_{i=0}^{N} P(x_i)x_i}{N \cdot \sum_{i=0}^{N} P(x_i)}$$

$$\theta = \frac{\sum_{i=0}^{N} P(x_i)x_i}{N}$$

Fifth, given the derivation above, we have found the maximizing parameter of the likelihood function. The maximizer is the ratio between the mean (probability-weighted average) of the observed discrete data, divided by the N, where $N + 1$ is the number of discrete data points observed.

$$\arg\max_{\theta} \ln(\mathcal{L}(\theta)) = \frac{\sum_{i=0}^{N} P(x_i)x_i}{N}$$

Suppose that the observe probability data is measured without error from a binomial distribution, in that case $\sum_{i=0}^{N} P(x_i)x_i = \mu_{\text{binomial}} = N \cdot \theta$, which means the mean number of "wins" is the number of "trials" times the probability of winning each trial.

1.1.1 Approximating Transition Probability

As an example, we now proceed with an example. Below we have a 5 by 5 transition matrix, where each of the row sums up to 1. We compute a binomial probability function to fit the probabilities in each of the rows of the matrix.

This matrix example below is used as the household structure kids transition matrix for a particular set of 18 year old individuals Nygaard, Sørensen, and Wang (2022) (preprint pdf). Each row corresponds to a different household size in the current period, and each column is the chance of transitioning to different household sizes in the next period.

First, generate the input matrixes.

```
# Construct transition matrix (row sums to 1, 5 by 5)
mt_pi_kids_trans <- matrix(
    data = c(
        0.8929,    0.1045,    0.0026,    0.0000,    0.0000,
        0.0547,    0.6704,    0.2708,    0.0040,    0.0001,
        0.0014,    0.0571,    0.7931,    0.1462,    0.0022,
        0.0000,    0.0009,    0.0748,    0.7885,    0.1358,
        0.0000,    0.0000,    0.0008,    0.0593,    0.9399
    ),
    nrow = 5, ncol = 5,
    byrow = TRUE
)
# check row sum
rowSums(mt_pi_kids_trans)
```

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

```
# Construct the x vector, this is (5 by 1)
# we have x_0, to x_4, for the 5 columns
# x_0 = 0, and x_4 = 4
ar_x <- matrix(c(0, 1, 2, 3, 4), nrow = 5, ncol = 1)
```

Second, estimate the θ parameters for each household size (row) this period.

```
# (5 by 5) times (5 by 1) generates (5 by 1)
# they correspond to 5 different \sum_{i=0}^N P(x_i)x_i
# Note divide by 4 not 5
ar_row_expected_x <- (mt_pi_kids_trans %*% ar_x) / (4)

# estimated thetas have been found
ar_theta_esti <- ar_row_expected_x
print(ar_theta_esti)
```

```
##          [,1]
## [1,] 0.027425
## [2,] 0.306100
## [3,] 0.522675
## [4,] 0.764800
## [5,] 0.984775
```

Third, generate the binomial probabilities and construct the 5 by 5 matrix based on that.

```
# Generate Binomial
mt_dbinom_approx <- t(sapply(
    ar_theta_esti, dbinom,
    x = seq(0, 4), size = 4
))
# Present
print(round(mt_dbinom_approx, 3))
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]
## [1,] 0.895 0.101 0.004 0.000 0.000
## [2,] 0.232 0.409 0.271 0.080 0.009
## [3,] 0.052 0.227 0.373 0.273 0.075
## [4,] 0.003 0.040 0.194 0.421 0.342
## [5,] 0.000 0.000 0.001 0.058 0.940
```

Evaluating the results, we see that the fit this provides a good fit for the first and fifth rows, but the fit for the three middle rows seem to be substantially worse. The observed transition mass is much more persistent than the fitted transition.

1.2 Estimate Change of Success and the Number of Trials

In contrast to the prior section, in this section, we will allow for both the "Number of Trials" and the "Chance of Success" parameters to vary to provide a better fit between transition probabilities and the data.

Since the Binomial distribution is approximately normal, so in principle, we should be able to use the binomial to fit a discretized normal random variable. A problem with the prior section is that the "number of trials" is only 4, which is very small. At this level, the binomial/normal approximation does not work well.

We will adjust our likelihood based estimation objective function. We assume that the observed probabilities are aggregated based on underlying subgroups. Our new likelihood function is:

$$\mathcal{L}(\theta, M) = \Pi_{i=0}^{N} \left(\sum_{j=0}^{M-1} P_{\text{binom}} \left(\underbrace{(i \times M + j)}_{\text{Number of success}} ; \overbrace{\theta}^{\text{suc. prb.}} , \underbrace{(N+1) \times M - 1}_{\text{Number of trials}} \right) \right)^{P(x_i)}$$

In the equation above, P_{binom} follows the same binomial probability formula as before, it is evaluated at the "Number of successes", given success probability θ and the "Number of Trials". To be consistent with the prior example, $N+1$ is the number of discrete observed probability categories. The new parameter $M \in \mathbb{N}$ is an integer that determines how many discrete evenly-spaced sub-categories are within each of the observed $N+1$ probability categories.

Given one set of observed probabilities to fit, the constrained maximum likelihood optimization problem is now:

$$\max_{\substack{\theta \in (0,1) \\ M \in \mathbb{N}}} \mathcal{L}(\theta, M)$$

When we have K sets of observed probabilities as in the example earlier with the five transition probabilities, we could allow for K different θ parameters, but constrain all K transitions to share the same M. In that case, the constrained optimization problem becomes:

$$\max_{\substack{\{\theta_k \in (0,1)\}_{k=1}^{K} \\ M \in \mathbb{N}}} \Pi_{k=1}^{K} \mathcal{L}(\theta_k, M)$$

We do not, however, have analytical solutions for this optimization problem. We will solve for this problem numerically.

1.2.1 Numerical Best Fit

For the numerical procedure, given M, we will find the best fitting θ, then we increment M, and we compare fit and best fitting θ across M.

First, we construct the likelihood function that considers the sum of probability between most adjacent groups. The function computes the log likelihood, given $N+1$ points of probability, θ values, and M.

```
#' Probability group aggregator
ffi_pbinom_group_theta_m <-
    function(fl_chance_success = 0.52,
             it_number_of_trials = 4,
             it_m = 8) {

        # it_n = (N+1)*M-1
        it_n <- it_number_of_trials
        # nbrtrl = number of trials
        it_nbrtrl <- (it_n + 1) * it_m - 1

        # Evaluate probability
        ar_dbinom <- dbinom(
            seq(0, it_nbrtrl),
            size = it_nbrtrl, prob = fl_chance_success
        )
        mt_dbinom <- matrix(
            ar_dbinom,
            nrow = it_n + 1, ncol = it_m,
            byrow = TRUE
```

```
        )

        # Sum probabilities each sub-groups
        ar_dbinom_grouped <- rowSums(mt_dbinom)

        return(ar_dbinom_grouped)
    }
# log-likelihood generator
# at the default value, the generated probabilities are similar to observed.
ffi_pbinom_lnlk_theta_m <-
    function(ar_drm_prob = c(0.0014, 0.0571, 0.7931, 0.1462, 0.0022),
             fl_chance_success = 0.52,
             it_m = 8) {

        # it_n = (N+1)*M-1
        it_n <- length(ar_drm_prob) - 1

        # Call the grouping function
        ar_dbinom_grouped <- ffi_pbinom_group_theta_m(
            fl_chance_success = fl_chance_success,
            it_number_of_trials = it_n,
            it_m = it_m
        )

        # Log of group-probabilities
        ar_dbinom_grouped_ln <- log(ar_dbinom_grouped)

        # Log likelihood
        fl_ln_likelihood <- ar_dbinom_grouped_ln %*% ar_drm_prob
        return(fl_ln_likelihood)
    }
```

Second, test the function above with different theta values. Note that given the input example probability array and fixing $M = 1$, choosing among 9 different θ values from 0.1 to 0.9, we find that $\theta = 0.5$ provides the highest likelihood. It is clear that the likelihood function is single-peaked and smooth in the example here as we vary over θ here.

```
# Test the function with different theta values
ar_drm_prob <- c(0.0014, 0.0571, 0.7931, 0.1462, 0.0022)
ar_fl_theta <- seq(0.1, 0.9, length.out = 9)
it_m <- 1
ar_ln_likelihood <- sapply(
    ar_fl_theta,
    ffi_pbinom_lnlk_theta_m,
    ar_drm_prob = ar_drm_prob,
    it_m = it_m
)
# Label and print
mt_theta_likelihood <- cbind(ar_fl_theta, ar_ln_likelihood)
colnames(mt_theta_likelihood) <- c("theta", "log_likelihood")
kable(as_tibble(mt_theta_likelihood) %>%
    mutate(rank = min_rank(desc(log_likelihood))), caption = paste(
    "Log likelihood evaluated at varying",
    "theta values, given some vector",
```

```
    "of observed probabilities, and",
    paste0("M=", it_m),
    separator = " "
)) %>% kable_styling_fc()
```

Log likelihood evaluated at varying theta values, given some vector of observed probabilities, and M=1

theta	log_likelihood	rank
0.1	-3.312301	9
0.2	-2.088022	7
0.3	-1.495267	5
0.4	-1.188130	3
0.5	-1.069711	1
0.6	-1.114579	2
0.7	-1.341568	4
0.8	-1.836548	6
0.9	-2.913725	8

Third, construct likelihood with different values of M. Note that we are using $\theta = 0.5$, which we found from the previous step as the likelihood maximizing parameter value given $M = 1$. The table that we generate below shows that the likelihood is maximized at $M = 7$. It is clear that the likelihood function is single-peaked and smooth in the example here as we vary over M

```
# Test the function with different theta values
ar_drm_prob <- c(0.0014, 0.0571, 0.7931, 0.1462, 0.0022)
fl_theta <- 0.50
ar_it_m <- seq(1, 10)
ar_ln_likelihood <- sapply(
    ar_it_m,
    ffi_pbinom_lnlk_theta_m,
    ar_drm_prob = ar_drm_prob,
    fl_chance_success = fl_theta
)
# Label and print
mt_m_likelihood <- cbind(ar_it_m, ar_ln_likelihood)
colnames(mt_m_likelihood) <- c("M", "log_likelihood")
kable(as_tibble(mt_m_likelihood) %>%
    mutate(rank = min_rank(desc(log_likelihood))), caption = paste(
    "Log likelihood evaluated at varying",
    "M values, given some vector",
    "of observed probabilities, and",
    paste0("theta=", fl_theta),
    separator = " "
)) %>% kable_styling_fc()
```

Fourth, we develop a grid evaluator of the function we just created. Given a vector of different θ values, we find the index that maximizes the likelihood, and we create new lower and upper bound that are around the maximizing grid point for θ.

```
# Evaluate likelihood along a grid and find the
# surrounding points around the maximizing grid point.
ffi_pbinom_lnlk_theta_m_grid <-
    function(ar_drm_prob = c(0.0014, 0.0571, 0.7931, 0.1462, 0.0022),
             it_m = 8,
             it_p_success_grid_len = 10,
```

Log likelihood evaluated at varying M values, given some vector of observed probabilities, and theta=0.5

M	log_likelihood	rank
1	-1.0697106	10
2	-0.8713480	9
3	-0.7772675	8
4	-0.7266149	7
5	-0.6988888	6
6	-0.6850268	3
7	-0.6803342	1
8	-0.6820966	2
9	-0.6886103	4
10	-0.6987371	5

```
             fl_min_p_success = 1e-5,
             fl_max_p_success = 1 - 1e-5) {

        # Theta array rebuilt
        ar_fl_theta <- seq(
            fl_min_p_success,
            fl_max_p_success,
            length.out = it_p_success_grid_len
        )

        # Evaluate likelihood
        ar_ln_likelihood <- sapply(
            ar_fl_theta,
            ffi_pbinom_lnlk_theta_m,
            ar_drm_prob = ar_drm_prob,
            it_m = it_m
        )
        # Find min grid
        it_max_idx <- which.max(ar_ln_likelihood)
        fl_max_val <- ar_ln_likelihood[it_max_idx]

        # Find lower and upper bound
        fl_min_p_succcess_new <- ar_fl_theta[
            max(it_max_idx - 1, 1)
        ]
        fl_max_p_succcess_new <- ar_fl_theta[
            min(it_max_idx + 1, it_p_success_grid_len)
        ]

        # return
        return(list(
            fl_max_val = fl_max_val,
            fl_min_p_succcess_new = fl_min_p_succcess_new,
            fl_max_p_succcess_new = fl_max_p_succcess_new
        ))
    }
# test
ffi_pbinom_lnlk_theta_m_grid()
```

```
## $fl_max_val
## [1] -0.7219323
##
## $fl_min_p_succcess_new
## [1] 0.4444456
##
## $fl_max_p_succcess_new
## [1] 0.6666633
```

Fifth, given $N+1$ points of data, and given M, we find the best fit θ value. We loop iteratively over grid several times. For an example of this iterative-grid algorithm, see Find Maximum By Iterating Over Grids.

```
# Find likelihood maximizing theta, given M
ffi_pbinom_lnlk_estitheta_fixm <-
    function(ar_drm_prob = c(0.0014, 0.0571, 0.7931, 0.1462, 0.0022),
             it_m = 1,
             verbose = FALSE) {

        # Initialize min and max and tolerance criteria
        fl_min_p_success_cur <- 1e-5
        fl_max_p_success_cur <- 1 - 1e-5
        it_p_success_grid_len <- 10
        fl_tol <- 1e-6
        it_max_iter <- 10

        # Initialize initial gaps etc
        fl_gap <- 1e5
        fl_min_ln_like_last <- 1e5
        it_iter <- 0

        # Iteratively loop over grid to find the maximum by zooming in
        while ((fl_gap > fl_tol) && it_iter <= it_max_iter) {

            # Iterator counts up
            it_iter <- it_iter + 1
            if (verbose) print(paste0("it_iter=", it_iter))

            # build array
            ls_find_max <- ffi_pbinom_lnlk_theta_m_grid(
                ar_drm_prob = ar_drm_prob,
                it_m = it_m,
                it_p_success_grid_len = it_p_success_grid_len,
                fl_min_p_success = fl_min_p_success_cur,
                fl_max_p_success = fl_max_p_success_cur
            )

            # Min objective value current
            fl_max_ln_like <- ls_find_max$fl_max_val
            # Find new lower and upper bound
            fl_min_p_success_cur <- ls_find_max$fl_min_p_succcess_new
            fl_max_p_success_cur <- ls_find_max$fl_max_p_succcess_new
            if (verbose) print(paste0("min_p_succ_cur=", fl_min_p_success_cur))
            if (verbose) print(paste0("max_p_succ_cur=", fl_max_p_success_cur))

            # Compare
```

```
            fl_gap <- abs(fl_max_ln_like - fl_min_ln_like_last)
            fl_max_ln_like_last <- fl_max_ln_like
            if (verbose) print(paste0("fl_gap=", fl_gap))
        }

        # Find estimate for theta
        fl_p_success_argmax <- (fl_min_p_success_cur + fl_max_p_success_cur) / 2

        # return
        return(list(
            fl_p_success_argmax = fl_p_success_argmax,
            fl_max_ln_like = fl_max_ln_like
        ))
    }
# Test with different M values
ar_drm_prob <- c(0.0014, 0.0571, 0.7931, 0.1462, 0.0022)
ffi_pbinom_lnlk_estitheta_fixm(ar_drm_prob = ar_drm_prob, it_m = 1)
```

```
## $fl_p_success_argmax
## [1] 0.522675
##
## $fl_max_ln_like
## [1] -1.065596
```

```
ffi_pbinom_lnlk_estitheta_fixm(ar_drm_prob = ar_drm_prob, it_m = 5)
```

```
## $fl_p_success_argmax
## [1] 0.5195473
##
## $fl_max_ln_like
## [1] -0.685463
```

```
ffi_pbinom_lnlk_estitheta_fixm(ar_drm_prob = ar_drm_prob, it_m = 9)
```

```
## $fl_p_success_argmax
## [1] 0.5211407
##
## $fl_max_ln_like
## [1] -0.665317
```

Sixth, we now proceed to find the optimum M value, assuming that multiple probability vectors can have different θ but must share M. We find the optimizing θ that differs for each of the discrete probability vectors.

```
# Find likelihood maximizing theta, given M
ffi_pbinom_lnlk_estijnt <-
    function(mt_drm_prob = matrix(
                 data = c(
                     0.8929,    0.1045,    0.0026,    0.0000,    0.0000,
                     0.0547,    0.6704,    0.2708,    0.0040,    0.0001
                 ),
                 nrow = 2, ncol = 5, byrow = TRUE
             ),
             verbose = FALSE) {

        # Initialize min and max and tolerance criteria
        it_drm_set <- dim(mt_drm_prob)[1]
```

```
        ar_it_m <- seq(1, 10)

        for (it_i in seq(1, it_drm_set)) {
            ls_estitheta_fixm_res <- sapply(
                ar_it_m,
                ffi_pbinom_lnlk_estitheta_fixm,
                ar_drm_prob = mt_drm_prob[it_i, ]
            )

            mt_estitheta_fixm_res <- t(matrix(
                as.numeric(ls_estitheta_fixm_res),
                nrow = 2, ncol = length(ls_estitheta_fixm_res) / 2
            ))
            mt_estitheta_fixm_res <- cbind(mt_estitheta_fixm_res, ar_it_m)
            colnames(mt_estitheta_fixm_res) <- c("esti_theta", "ln_like", "M")

            tb_estitheta_fixm_res <-
                as_tibble(mt_estitheta_fixm_res) %>%
                mutate(drm_group = it_i)

            if (it_i > 1) {
                tb_estitheta_fixm_res_stack <-
                    bind_rows(
                        tb_estitheta_fixm_res_stack,
                        tb_estitheta_fixm_res
                    )
            } else {
                tb_estitheta_fixm_res_stack <- tb_estitheta_fixm_res
            }
        }

        # Group max
        tb_estitheta_fixm_res_stack <- tb_estitheta_fixm_res_stack %>%
            group_by(drm_group) %>%
            mutate(drm_group_rank = min_rank(desc(ln_like))) %>%
            ungroup()

        # Overall M that maximizes sum of log likelihood
        tb_estitheta_fixm_res_stack <- tb_estitheta_fixm_res_stack %>%
            group_by(M) %>%
            mutate(ln_like_M_sum = sum(ln_like)) %>%
            ungroup() %>%
            mutate(drm_jnt_rank = dense_rank(desc(ln_like_M_sum))) %>%
            arrange(drm_group, M)

        # Return
        return(tb_estitheta_fixm_res_stack)
    }
# Test with full transition matrix
tb_rest_res <- ffi_pbinom_lnlk_estijnt(mt_drm_prob = mt_pi_kids_trans)
tb_rest_res %>% filter(drm_jnt_rank == 1)
```

Seventh, we evaluat the predicted probabilities given our optimizing parameters.

```r
# Compute predicted probabilities at maxmizing parameters
mt_dbinom_grouped <-
    apply(
        tb_rest_res %>% filter(drm_jnt_rank == 1),
        1,
        function(row) {
            ffi_pbinom_group_theta_m(
                fl_chance_success = row[["esti_theta"]],
                it_number_of_trials = (dim(mt_pi_kids_trans)[1] - 1),
                it_m = row[["M"]]
            )
        }
    )
# Show predictions
kable(round(t(mt_dbinom_grouped), 3),
    caption = "Predicted transitions given estimates"
) %>% kable_styling_fc()
```

Predicted transitions given estimates

0.889	0.111	0.000	0.000	0.000
0.038	0.702	0.258	0.002	0.000
0.000	0.092	0.724	0.183	0.001
0.000	0.000	0.081	0.777	0.141
0.000	0.000	0.000	0.061	0.939

```r
# Show data
kable(round(mt_pi_kids_trans, 3),
    caption = "Data transitions probalities"
) %>% kable_styling_fc()
```

Data transitions probalities

0.893	0.104	0.003	0.000	0.000
0.055	0.670	0.271	0.004	0.000
0.001	0.057	0.793	0.146	0.002
0.000	0.001	0.075	0.788	0.136
0.000	0.000	0.001	0.059	0.940

```r
# Show data and prediction differences
kable(round(mt_pi_kids_trans - t(mt_dbinom_grouped), 3),
    caption = "Data transitions probalities - predictions"
) %>% kable_styling_fc()
```

Data transitions probalities - predictions

0.004	-0.007	0.003	0.000	0.000
0.017	-0.032	0.013	0.002	0.000
0.001	-0.035	0.069	-0.037	0.002
0.000	0.001	-0.007	0.011	-0.006
0.000	0.000	0.001	-0.002	0.001