Assignment 2: Production Function Estimation

Kohei Kawaguchi

2019/1/29

Simulate data

Consider the following production and investment process for $j = 1, \cdots, 1000$ firms across $t = 1, \cdots, 10$ periods.

The log production function is of the form:

$$y_{jt} = \beta_0 + \beta_l l_{jt} + \beta_k k_{jt} + \omega_{jt} + \eta_{jt},$$

where ω_{jt} is an anticipated shock and η_{jt} is an ex post shock.

The anticipated shocks evolve as:

$$\omega_{jt} = \alpha \omega_{j,t-1} + \nu_{jt},$$

where ν_{jt} is an i.i.d. normal random variable with mean 0 and standard deviation σ_ν. The ex post shock is an i.i.d. normal random vavriable with mean 0 and standard deviation σ_η.

The product price the same across firms and normalized at 1. The price is normalized at 1. The wage w_t is an i.i.d. exponential of normal random variable with mean 0 and standard deviation σ_w.

Finally, the capital accumulate according to:

$$K_{j,t+1} = (1 - \delta) K_{jt} + I_{jt}.$$

We set the parameters as follows:

parameter	variable	value
β_0	`beta_0`	1
β_l	`beta_l`	0.2
β_k	`beta_k`	0.7
α	`alpha`	0.7
σ_η	`sigma_eta`	0.2
σ_ν	`sigma_nu`	0.1
σ_w	`sigma_w`	0.1
δ	`delta`	0.05

1. Define the parameter variables as above.

```
# size
J <- 1000
T <- 10

# production function
beta_0 <- 1
beta_l <- 0.2
beta_k <- 0.7
sigma_eta <- 0.2

# anticipated shocks
```

```
sigma_nu <- 0.1
alpha <- 0.7

# wage
sigma_w <- 0.1

# capital
delta <- 0.05
```

2. Write a function that returns the log output given l_{jt}, k_{jt}, ω_{jt}, and l_{jt} under the given parameter values according to the production function and name it `log_production`.

```
log_production <-
  function(l, k, omega, eta, beta_0, beta_l, beta_k) {
    y <- beta_0 + beta_l * l + beta_k * k + omega + eta
    return(y)
  }
```

Suppose that the labor is determined after ω_{jt} is observed, but before η_{jt} is observed, given the log capital level k_{jt}.

3. Derive the optimal log labor as a function of ω_{jt}, η_{jt}, k_{jt}, and w_t. Write a function to return the optimal log labor given the variables and parameters and name it `log_labor_choice`.

```
log_labor_choice <-
  function(k, wage, omega, beta_0, beta_l, beta_k) {
    L <-
      ((1/wage) *
         exp(beta_0 + omega) *
         beta_l *
         exp(k)^beta_k)^(1/(1 - beta_l))
    l <- log(L)
  return(L)
  }
```

As discussed in the class, if there is no additional variation in labor, the coefficient on the labor β_l is not identified. Thus, if we generate labor choice from the previous function, β_l will not be identified from the simulated data. Too see this, we write a modified version of the previous function in which ω_{jt} is replaced with $\omega_{jt} + \iota_{jt}$, where ι_{jt} is an optimaization error that follows an i.i.d. normal distribution with mean 0 and standard deviation 0.05. That is, the manager of the firm percceives as if the shock is $\omega_{jt} + \iota_{jt}$, even though the true shock is ω_{jt}.

4. Modify the previous function by including ι_{jt} as an additional input and name it `log_labor_choice_error`.

```
log_labor_choice_error <-
  function(k, wage, omega, beta_0, beta_l, beta_k, iota) {
    L <-
      ((1/wage) *
         exp(beta_0 + omega + iota) *
         beta_l *
         exp(k)^beta_k)^(1/(1 - beta_l))
    l <- log(L)
  return(L)
  }
```

Consider an investment process such that:

$$I_{jt} = (\delta + \gamma w_{jt}) K_{jt},$$

where I_{jt} and K_{jt} are investment and capital in level. Set $\gamma = 0.1$, i.e., the investment is strictly increasing in ω_{jt}. The investment function should be devied by solving the dynamic problem of a firm. But here, we just specify it in a reduced-form.

5. Define variable γ and assign it the value. Write a function that returns the investment given K_{jt}, ω_{jt}, and parameter values, according to the previous equation, and name it `investment_choice`.

```
gamma <- 0.1
investment_choice <-
  function(k, omega, gamma, delta) {
    I <- (delta + gamma * omega) * exp(k)
    return(I)
  }
```

Now simulate the data first using the labor choice without optimization error and second using the labor choice with optimization error. To do so, we specify the initial values for the state variables k_{jt} and ω_{jt}.

6. Draw k_{j0} from i.i.d. normal distribution with mean 1 and standard deviation 0.5. Draw ω_{j0} from its stationary distribution (check the stationary distribution of AR(1) process). Draw a wage.

```
# optimization error
sigma_iota <- 0.05

# parameters for the initial state distribution
mean_k_initial <- 1
sigma_k_initial <- 0.5
sigma_v_initial <- sigma_nu / sqrt(1 - alpha^2)

# set seed
set.seed(1)

# draw initial state values
wage <- exp(rnorm(1, sigma_w))
df <- expand.grid(j = 1:J, t = 1) %>%
  tibble::as.tibble() %>%
  dplyr::mutate(
    k = rnorm(J, mean_k_initial, sigma_k_initial),
    omega = rnorm(J, 0, sigma_v_initial),
    wage = wage
  )
df
```

```
## # A tibble: 1,000 x 5
##        j     t     k    omega  wage
##    <int> <dbl> <dbl>    <dbl> <dbl>
##  1     1     1 1.09   0.156   0.591
##  2     2     1 0.582 -0.122   0.591
##  3     3     1 1.80   0.0295  0.591
##  4     4     1 1.16   0.00972 0.591
##  5     5     1 0.590 -0.233   0.591
##  6     6     1 1.24   0.114   0.591
##  7     7     1 1.37  -0.268   0.591
##  8     8     1 1.29  -0.175   0.591
##  9     9     1 0.847  0.140   0.591
## 10    10     1 1.76  -0.0757  0.591
## # ... with 990 more rows
```

7. Compute the labor and investment choice of period 1. For labor choice, compute both types of labor choices.

```
# compute labor and investment
df <- df %>%
  dplyr::mutate(iota = rnorm(J, 0, sigma_iota)) %>%
  dplyr::rowwise() %>%
  dplyr::mutate(
    l = log_labor_choice(k, wage, omega, beta_0, beta_l, beta_k),
    l_error = log_labor_choice_error(k, wage, omega, beta_0, beta_l, beta_k, iota),
    I = investment_choice(k, omega, gamma, delta)
  ) %>%
  dplyr::ungroup()
df
```

```
## # A tibble: 1,000 x 9
##        j     t     k    omega  wage     iota     l l_error      I
##    <int> <dbl> <dbl>    <dbl> <dbl>    <dbl> <dbl>   <dbl>  <dbl>
##  1     1     1     1 1.09    0.156   0.591 -0.0961   2.85    2.52 0.195
##  2     2     1     1 0.582 -0.122    0.591  0.0810   1.29    1.43 0.0677
##  3     3     1     1 1.80    0.0295  0.591  0.0260   4.51    4.66 0.320
##  4     4     1     1 1.16    0.00972 0.591 -0.00279  2.53    2.52 0.163
##  5     5     1     1 0.590 -0.233    0.591  0.0348   1.13    1.18 0.0482
##  6     6     1     1 1.24    0.114   0.591  0.00268  3.08    3.10 0.213
##  7     7     1     1 1.37  -0.268    0.591 -0.0655   2.14    1.97 0.0913
##  8     8     1     1 1.29  -0.175    0.591 -0.106    2.24    1.96 0.118
##  9     9     1     1 0.847  0.140    0.591 -0.0104   2.25    2.22 0.149
## 10    10     1     1 1.76  -0.0757   0.591 -0.0156   3.81    3.74 0.246
## # ... with 990 more rows
```

8. Draw ex post shock and compute the output according to the production function for both labor without optimization error and with optimization error.

```
# compute output
df <- df %>%
  dplyr::mutate(eta = rnorm(J, 0, sigma_eta)) %>%
  dplyr::rowwise() %>%
  dplyr::mutate(
    y = log_production(l, k, omega, eta, beta_0, beta_l, beta_k),
    y_error = log_production(l_error, k, omega, eta, beta_0, beta_l, beta_k)
  ) %>%
  dplyr::ungroup()
df
```

```
## # A tibble: 1,000 x 12
##        j     t     k    omega  wage     iota     l l_error      I     eta
##    <int> <dbl> <dbl>    <dbl> <dbl>    <dbl> <dbl>   <dbl>  <dbl>   <dbl>
##  1     1     1     1 1.09    0.156   0.591 -0.0961   2.85    2.52 0.195   0.0773
##  2     2     1     1 0.582 -0.122    0.591  0.0810   1.29    1.43 0.0677  0.259
##  3     3     1     1 1.80    0.0295  0.591  0.0260   4.51    4.66 0.320  -0.161
##  4     4     1     1 1.16    0.00972 0.591 -0.00279  2.53    2.52 0.163  -0.321
##  5     5     1     1 0.590 -0.233    0.591  0.0348   1.13    1.18 0.0482  0.187
##  6     6     1     1 1.24    0.114   0.591  0.00268  3.08    3.10 0.213   0.361
##  7     7     1     1 1.37  -0.268    0.591 -0.0655   2.14    1.97 0.0913 -0.0113
##  8     8     1     1 1.29  -0.175    0.591 -0.106    2.24    1.96 0.118   0.377
##  9     9     1     1 0.847  0.140    0.591 -0.0104   2.25    2.22 0.149   0.316
```

```
## 10    10     1 1.76  -0.0757  0.591 -0.0156   3.81    3.74 0.246   0.100
## # ... with 990 more rows, and 2 more variables: y <dbl>, y_error <dbl>
```

9. Repeat this procedure for $t = 1, \cdots 10$ by updating the capital and anticipated shocks.

```
df_T <- df
for (t in 2:T) {
  # change time index
  df$t <- t

  # draw wage
  wage <- exp(rnorm(1, sigma_w))
  df$wage <- wage

  # update capital
  df <- df %>%
    dplyr::mutate(
      k = log((1 - delta) * exp(k) + I)
    )

  # update omega
  df <- df %>%
    dplyr::mutate(
      nu = rnorm(J, 0, sigma_nu),
      omega = alpha * omega + nu
    )

  # compute labor and investment
  df <- df %>%
    dplyr::mutate(iota = rnorm(J, 0, sigma_iota)) %>%
    dplyr::rowwise() %>%
    dplyr::mutate(
      l = log_labor_choice(k, wage, omega, beta_0, beta_l, beta_k),
      l_error = log_labor_choice_error(k, wage, omega, beta_0, beta_l, beta_k, iota),
      I = investment_choice(k, omega, gamma, delta)
    ) %>%
    dplyr::ungroup()
  # compute output
  df <- df %>%
    dplyr::mutate(eta = rnorm(J, 0, sigma_eta)) %>%
    dplyr::rowwise() %>%
    dplyr::mutate(
      y = log_production(l, k, omega, eta, beta_0, beta_l, beta_k),
      y_error = log_production(l_error, k, omega, eta, beta_0, beta_l, beta_k)
    ) %>%
    dplyr::ungroup()

  # append
  df_T <- dplyr::bind_rows(df_T, df)
}
df_T
```

```
## # A tibble: 10,000 x 13
##        j     t     k    omega  wage     iota     l l_error      I     eta
##    <int> <dbl> <dbl>    <dbl> <dbl>    <dbl> <dbl>   <dbl>  <dbl>   <dbl>
```

```
##  1     1     1 1.09   0.156   0.591 -0.0961   2.85    2.52 0.195   0.0773
##  2     2     1 0.582 -0.122   0.591  0.0810   1.29    1.43 0.0677  0.259
##  3     3     1 1.80   0.0295  0.591  0.0260   4.51    4.66 0.320  -0.161
##  4     4     1 1.16   0.00972 0.591 -0.00279  2.53    2.52 0.163  -0.321
##  5     5     1 0.590 -0.233   0.591  0.0348   1.13    1.18 0.0482  0.187
##  6     6     1 1.24   0.114   0.591  0.00268  3.08    3.10 0.213   0.361
##  7     7     1 1.37  -0.268   0.591 -0.0655   2.14    1.97 0.0913 -0.0113
##  8     8     1 1.29  -0.175   0.591 -0.106    2.24    1.96 0.118   0.377
##  9     9     1 0.847  0.140   0.591 -0.0104   2.25    2.22 0.149   0.316
## 10    10     1 1.76  -0.0757  0.591 -0.0156   3.81    3.74 0.246   0.100
## # ... with 9,990 more rows, and 3 more variables: y <dbl>, y_error <dbl>,
## #   nu <dbl>
```

10. Check the simulated data by making summary table.

```
df_T_summary <-
  foreach (i = 1:dim(df_T)[2], .combine = "rbind") %do% {
    df_T_i <- as.data.frame(df_T)[, i]
    row_i <- data.frame(
      N = length(df_T_i),
      Mean = mean(df_T_i, na.rm = TRUE),
      Sd = sd(df_T_i, na.rm = TRUE),
      Min = min(df_T_i, na.rm = TRUE),
      Max = max(df_T_i, na.rm = TRUE))
    rownames(row_i) <- colnames(df_T)[i]
    return(row_i)
  }
kable(df_T_summary)
```

	N	Mean	Sd	Min	Max
j	10000	500.5000000	288.6894251	1.0000000	1000.0000000
t	10000	5.5000000	2.8724249	1.0000000	10.0000000
k	10000	0.9940171	0.5199006	-0.6486048	3.0230352
omega	10000	-0.0008504	0.1389409	-0.4944505	0.5096422
wage	10000	1.2999738	1.0034241	0.2412463	2.9228370
iota	10000	-0.0000634	0.0502874	-0.1841453	0.1715419
l	10000	2.5571681	3.0000325	0.0566083	46.2630759
l_error	10000	2.5641428	3.0262212	0.0571485	51.1413043
I	10000	0.1552468	0.1028476	0.0010667	1.4612200
eta	10000	0.0012772	0.1999334	-0.7650371	0.7455922
y	10000	2.2076724	0.8688350	0.1989304	12.5394398
y_error	10000	2.2090673	0.8727397	0.1968913	13.5150855
nu	10000	-0.0002718	0.0998171	-0.4302781	0.3650776