Dark Rate Estimation - Simulation with multiple different input variables

Nicole Bellert

2024-02-09

Simulation of collusive and non-collusive behaviour and detection

Stigler ICC: delta > 1 - 1/nfirms
delta = c_value
C if delta >= 1 - 1/nfirms = ICC
same as delta <= 1/nfirms

Z is policy shock and happens after several time periods.

```
# Set random seed op <- options(digits.secs = 6) sim_seed <-
# as.numeric(Sys.time()) set.seed(sim_seed)

# Seed used for graphs and tables to get reproducible code
sim_seed <- 1673465635  # seed for enforcement
set.seed(sim_seed)
```

Set basic parameters

```
n <- 99  # number of industries
periods_in <- 4
share_no_z_in <- 0.5  # share of time periods without policy shock
nf <- 2:10  # number of firms in industry
n_firms_in <- rep(nf, n/length(nf))
sigma1_in <- seq(0.1, 0.4, 0.1)
sigma2_in <- seq(0.3, 0.7, 0.1)
sigma_re_in <- seq(0.4, 0.9, 0.1)
n_sim <- length(periods_in) * length(share_no_z_in) * length(sigma1_in) * length(sigma2_in) *
    length(sigma_re_in)
input <- tibble(input_id = 1:n_sim, seed = sim_seed, periods = rep(periods_in, each = n_sim/length(periods_in)),
    share_no_z = rep(share_no_z_in, times = length(periods_in), each = n_sim/(length(periods_in) *
        length(share_no_z_in))), sigma1 = rep(sigma1_in, times = length(periods_in) *
        length(share_no_z_in), each = n_sim/(length(periods_in) * length(share_no_z_in) *
        length(sigma1_in))), sigma2 = rep(sigma2_in, times = length(periods_in) *
        length(share_no_z_in) * length(sigma1_in), each = n_sim/(length(periods_in) *
        length(share_no_z_in) * length(sigma1_in) * length(sigma2_in))), sigma_re = rep(sigma_re_in,
        times = length(periods_in) * length(share_no_z_in) * length(sigma1_in) *
            length(sigma2_in), each = n_sim/(length(periods_in) * length(share_no_z_in) *
            length(sigma1_in) * length(sigma2_in) * length(sigma_re_in))), )
```

```
# delete if not sigma1 < sigma2 < sigma_re
input <- filter(input, (sigma1 < sigma2) & (sigma2 < sigma_re))
output <- input %>%
    mutate(D = 0, C = 0, ARE = 0, LARE = 0, E_D1D2Z = 0, E_D1D2 = 0, C_hat1 = 0,
        C_hat = 0, compliers = 0)
```

build paneldata

```
build_panel <- function(ip, n) {
    ind_id <- 1:n
    t_in <- 1:ip$periods
    periods_no_z <- round(ip$periods * ip$share_no_z, 0)
    periods_z <- ip$periods - periods_no_z
```

```
    Z_in <- c(rep(0, periods_no_z), rep(1, periods_z))
    c_value_in <- runif(n)
    d_value_in = runif(n * ip$periods)

    df <- tibble(input_id = ip$input_id, t = rep(t_in, each = n), ind_id = rep(ind_id,
        length(t_in)), n_firms = rep(n_firms_in, length(t_in)), Z = rep(Z_in, each = n),
        prob_c = 1/n_firms, prob_c_rep = 1/(2 * n_firms), c_value = rep(c_value_in,
            length(t_in)), prob_d = ifelse(Z == 0, ip$sigma1, ip$sigma2), prob_d_rep = ip$sigma_re,
        d_value = d_value_in, C = ifelse(t == 1, as.numeric(c_value <= prob_c), 0),
        D1 = 0, D2 = C * ifelse(t == 1, as.numeric(d_value <= prob_d), 0))
    df <- arrange(df, ind_id, t)
}
simulate_detection <- function(df) {
    for (j in 1:length(df$t)) {
        if (df$t[j] > 1) {
            df$D1[j] <- ifelse(df$D2[j - 1] == 1, 1, 0)  # 1 for detected yesterday.
            df$C[j] <- ifelse(df$D1[j] == 1, as.numeric(df$c_value[j] <= df$prob_c_rep[j]),
                as.numeric(df$c_value[j] <= df$prob_c[j]))
            df$D2[j] <- df$C[j] * ifelse(df$D1[j] == 1, as.numeric(df$d_value[j] <=
                df$prob_d_rep[j]), as.numeric(df$d_value[j] <= df$prob_d[j]))
        }
    }
    return(df)
}
add_complier <- function(df) {
    df <- df %>%
        group_by(ind_id) %>%
        mutate(complier1 = as.numeric((lead(D2 == 1) & (Z == 0)))) %>%
        mutate(complier1 = ifelse(is.na(complier1), 0, complier1)) %>%
        mutate(complier2 = as.numeric(lead(complier1 == 1))) %>%
        mutate(complier2 = ifelse(is.na(complier2), 0, complier2)) %>%
        mutate(complier3 = ifelse((complier1 + complier2) > 0, 1, 0)) %>%
        mutate(complier = max(complier3)) %>%
        mutate(D1D2 = D1 * D2)
}
```

Dark Rate Estimation

$ARE = E[D2 - D1D2] = E[D2 - D1D2|C2] * Pr(C2)$

$\text{LARE} = E[D2 - D1D2|complier]$

$\hat{E}(C) = ARE/LARE$

```
estimate_output <- function(df, ip) {
    df_comp = df[df$complier == 1, ]
    op <- ip %>%
        mutate(D = mean(df$D2), C = mean(df$C), ARE = mean(df$D2 - df$D1D2), LARE = mean(df_comp$D2 -
            df_comp$D1D2), E_D1D2Z = mean(df$D1D2Z), E_D1D2 = mean(df$D1D2), C_hat1 = ARE/LARE,
            C_hat = ifelse(D > C_hat1, D, C_hat1), compliers = mean(df$complier))
    op
}
```

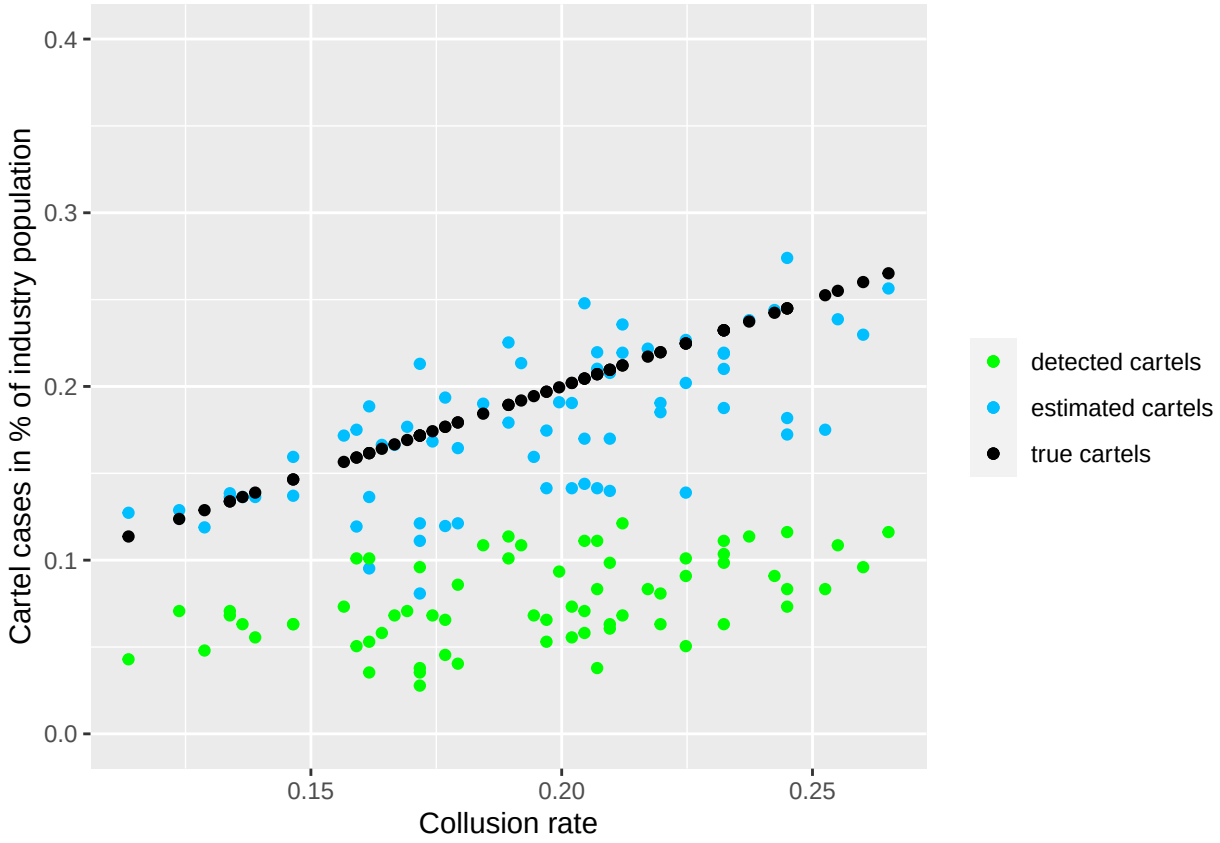
Simulate for all rows of input values

```
for (i in 1:length(input$seed)) {
    ip <- input[i, ]
    pd <- build_panel(ip, n)
    pd <- simulate_detection(pd)
    pd <- add_complier(pd)
    op <- estimate_output(pd, ip)
```

```
    output[i, ] <- op
}
```

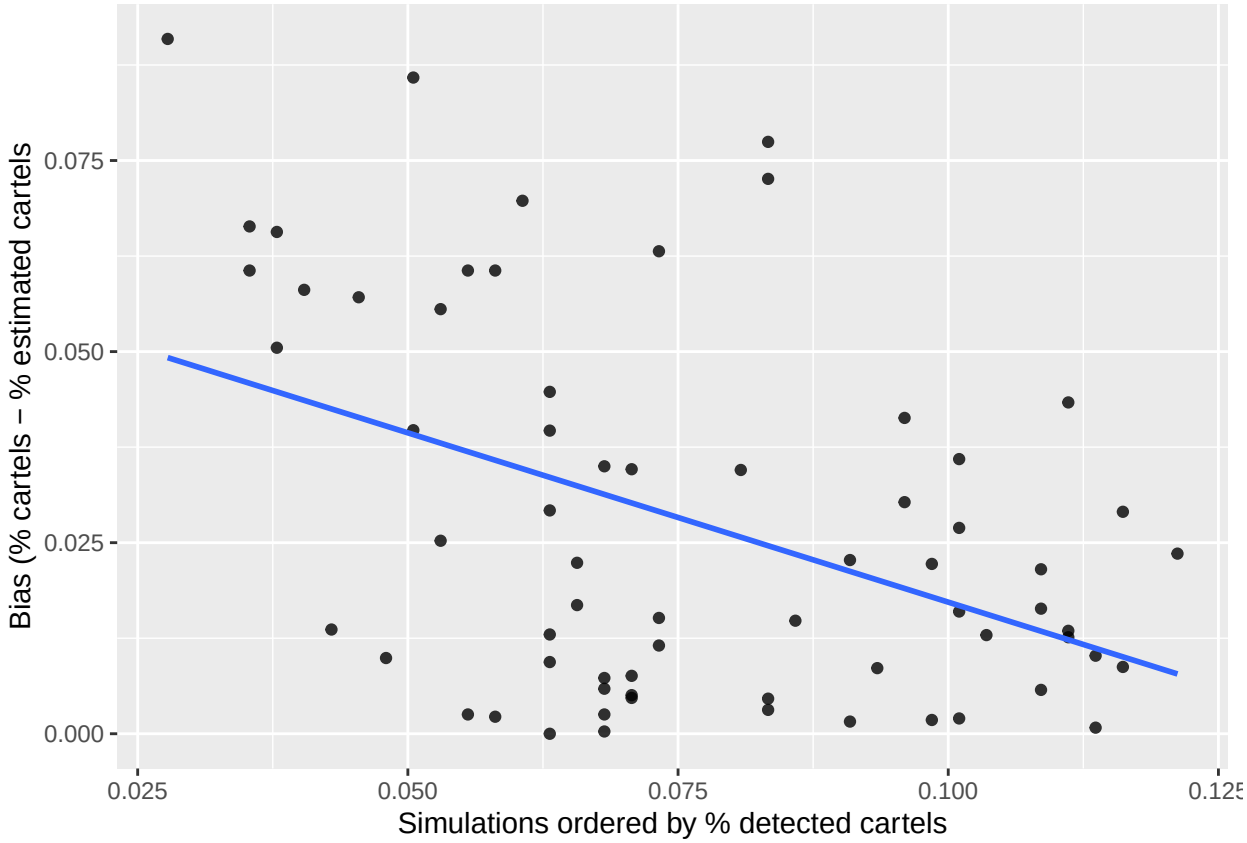
```
output <- output %>%
    replace(is.na(.), 0) %>%
    mutate(bias = abs(C - C_hat)) %>%
    arrange(C, D)
```

```
ggplot(output, aes(x = C, y = C_hat)) + geom_point(aes(x = C, y = output$C_hat, colour = "estimated cartels"),
    alpha = 1, size = 1.5) + geom_point(aes(x = C, y = output$C, colour = "true cartels"),
    alpha = 1, size = 1.5) + geom_point(aes(x = C, y = output$D, colour = "detected cartels"),
    alpha = 1, size = 1.5) + scale_color_manual(name = "", values = c("green", "deepskyblue",
    "black")) + ylim(0, 0.4) + ylab("Cartel cases in % of industry population") +
    xlab("Collusion rate")
```




```
ggplot(output, aes(x = D, y = bias)) + geom_point(alpha = 0.8, size = 1.5) + geom_smooth(method = lm,
    se = FALSE) + ylab("Bias (% cartels - % estimated cartels") + xlab("Simulations ordered by % detected cartel
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
df1 <- output %>%
    select(sigma1, sigma2, sigma_re, D, C, C_hat, bias) %>%
    arrange(sigma1, sigma2, sigma_re)
df <- round(df1, 3)
k <- kbl(df, "latex", booktabs = T, linesep = "")
```

```
pd1 <- pd %>%
    select(ind_id, t, Z, C, D2, D1, complier) %>%
    filter(ind_id == 1)
k <- kbl(pd1, "latex", booktabs = T, linesep = "")
```