Comment on Itai Sher's Perspective-Based Preference Aggregation

Vlad Tarko

6/1/2021

This is a comment on Itai Sher, "How perspective-based aggregation undermines the Pareto principle", *Politics, Philosophy & Economics*, Vol 19, Issue 2, 2020.

See also PeaSoup discussion.

Preference formation

Each agent i judges the issue based on a number of criteria, $C_1, C_2, ..., C_K$. Each individual i has beliefs about the probability of whether each criterion k is satisfied, p_k^i. The individual preference is Boolean (yes/no; approve/disapprove).

All the criteria are assumed to be necessary, i.e. the preference of individual i is:

$$\phi^i = f(p_1^i, p_2^i, ..., p_K^i) = (p_1^i > T) \ \& \ (p_2^i > T) \ \&...\& \ (p_K^i > T)$$

where T is a threshold of acceptance, e.g. $T = 0.5$.

Group decision

Group decision can be taken in two ways:

- by preference aggregation
- by knowledge aggregation

Do these two methods agree? How often can we expect them to disagree?

If we use preference aggregation by unanimity rule, the group decision will be:

$$g_{pref} = \phi^1 \ \& \ \phi^2 \ \&...\& \ \phi^N$$

The group will take decision 'yes' only if everyone in the group agrees.

Group decision by knowledge aggregation involves two steps:

1. Aggregating the beliefs about whether each criterion is satisfied, using majority rule, i.e. the group belief about whether the criterion k is satisfied is

$$P_k = \sum_i \gamma_k^i p_k^i$$

where γ_k^i is the credibility of agent i about criterion k. I'm using

$$\gamma_k^i = \frac{1}{N}$$

2. Applying function f to the group beliefs, i.e.

$$g_{belief} = f(P_1, P_2, ..., P_K) = (P_1 > T) \ \& \ (P_2 > T) \ \& \ ... \ \& \ (P_K > T)$$

Simulation

We can create N agents with different beliefs (probability estimates) about the K criteria.

Consider three probability distributions for generating these beliefs:

- uniform probability distribution between $0.5 - range$ and $0.5 + range$
- binomial probability distribution with a probability of success between 0 and 1
- normal probability distribution with a mean between 0 and 1, and a standard devition of 0.2

The main function

```
# how many agents (N) and how many criteria (K)
Itai <- function(N, K, type = "binomial",
                 size = 100, prob = 0.5, # for binomial distribution
                 mean = 0.5, sd = 0.2,   # for normal distribution
                 range = 0.5,            # for uniform distribution
                 threshold = 0.5) {

  # assign random probabilities that the criteria are satisfied
  # these represent individual beliefs that each criterion is satisfied
  switch (type,
    binomial = { M = matrix(rbinom(N*K, size = size, prob = prob), ncol = K, nrow = N)/size },
    uniform  = { M = matrix(runif(N*K, min = 0.5 - range, max = 0.5 + range), ncol = K, nrow = N) },
    normal   = { M = matrix(rnorm(N*K, mean = mean, sd = sd), ncol = K, nrow = N) }
  )

  # function that calculates an agent's preference
  # x is a vector of probabilities, each probability corresponding to a different criterion
  # if probability is above threshold, the criterion is assumed to be satisfied
  # all criteria are assumed to be necessary
  f <- function(x, t = threshold) { Reduce(`&`, x > t) }

  # calculate all agents' preferences
  pref <- vector(length = N)
  for (i in 1:N) { pref[i] <- f(M[i,]) }

  # aggregate individual preferences
  group_pref_1 <- Reduce(`&`, pref)

  # aggregate beliefs
  group_beliefs <- colMeans(M)
  group_pref_2 <- f(group_beliefs)

  # return whether the two aggregation procedures agree
  return(group_pref_1 == group_pref_2)

}
```

Testing the effect of different probability distributions

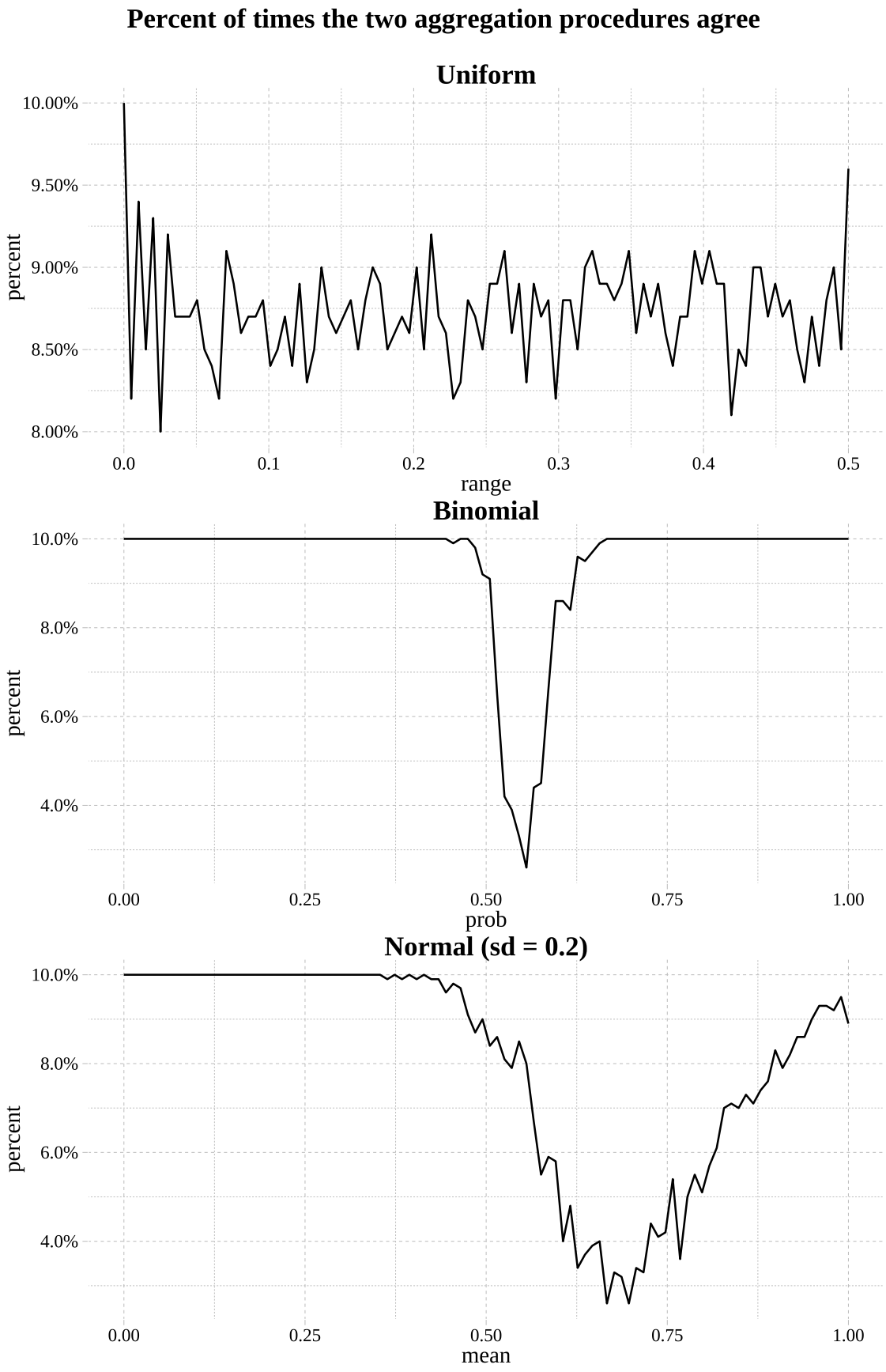
```
# calculate the percent of times that the two aggregation methods agree

df_normal <- tibble(
  mean = seq(0, 1, length.out = 100),
  percent  = map_dbl(mean,
                function(m) {
                  map_lgl(1:100,
                          ~Itai(3, 3, "normal", mean = m, sd = 0.2)) %>%
                             sum()/1000
                })
)

df_binom <- tibble(
  prob = seq(0, 1, length.out = 100),
  percent  = map_dbl(prob,
                function(p) {
                  map_lgl(1:100,
                          ~Itai(3, 3, "binomial", prob = p)) %>% sum()/1000
                })
)

df_unif <- tibble(
  range = seq(0, 0.5, length.out = 100),
  percent  = map_dbl(range,
                function(r) {
                  map_lgl(1:100,
                          ~Itai(3, 3, "uniform", range = r)) %>% sum()/1000
                })
)
```

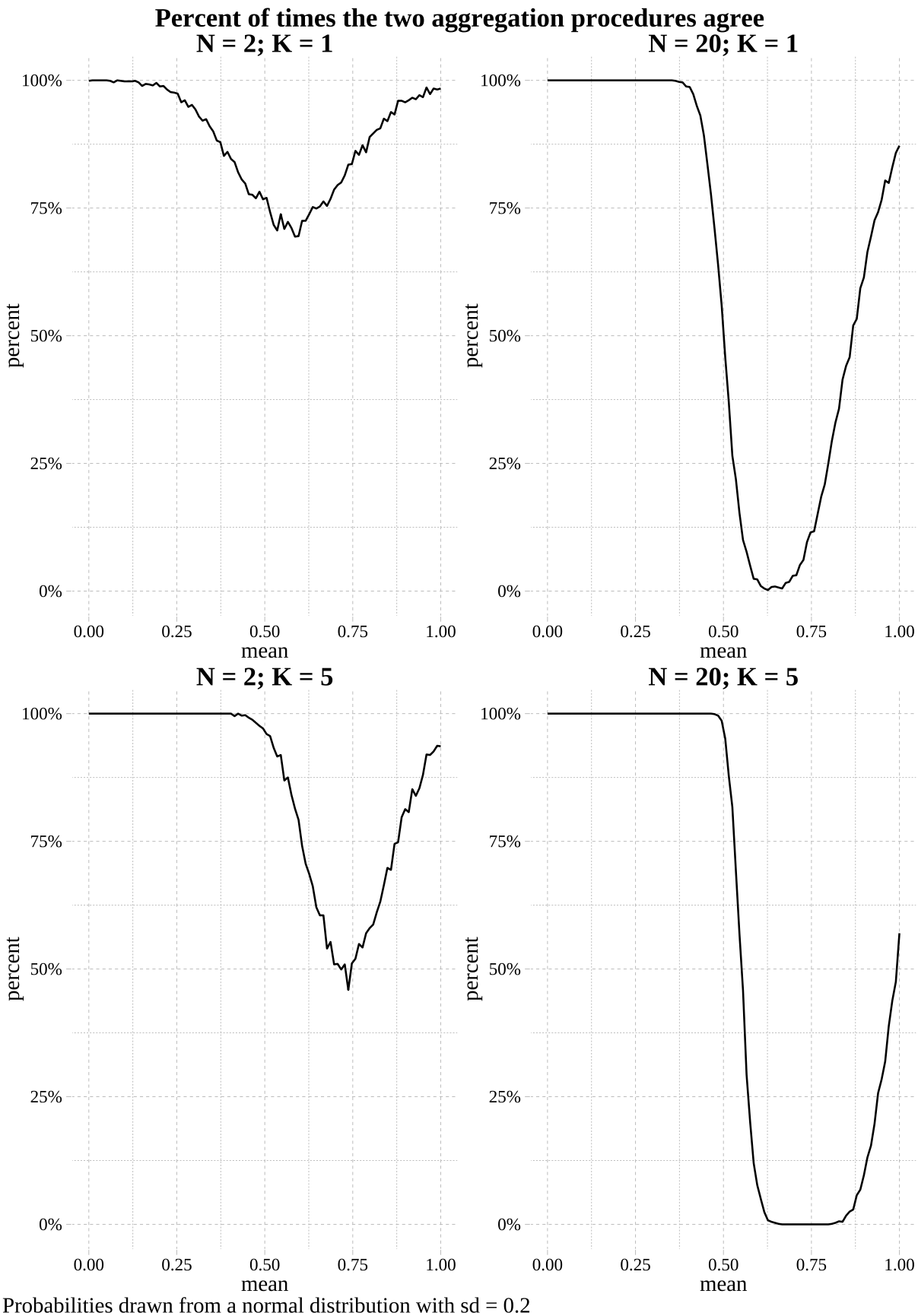
```
{ ggplot(df_unif, aes(x = range, y = percent)) +
    geom_line() +
    scale_y_continuous(labels = scales::percent) +
    labs(title = "Uniform") } /
{ ggplot(df_binom, aes(x = prob, y = percent)) +
    geom_line() +
    scale_y_continuous(labels = scales::percent) +
    labs(title = "Binomial") } /
{ ggplot(df_normal, aes(x = mean, y = percent)) +
    geom_line() +
    scale_y_continuous(labels = scales::percent) +
    labs(title = "Normal (sd = 0.2)") } +
  plot_annotation(title = "Percent of times the two aggregation procedures agree\n")
```


Percent of times the two aggregation procedures agree
Uniform
10.00%
9.50%
9.00%
8.50%
8.00%
percent
0.0
0.1
0.2
0.3
0.4
0.5
range
Binomial
10.0%
8.0%
6.0%
4.0%
percent
0.00
0.25
0.50
0.75
1.00
prob
Normal (sd = 0.2)
10.0%
8.0%
6.0%
4.0%
percent
0.00
0.25
0.50
0.75
1.00
mean

Testing the effect of different N and K

```
MonteCarlo <- function(N, K){
  tibble(
    mean = seq(0, 1, length.out = 100),
    percent  = map_dbl(mean,
                  function(m) {
                    map_lgl(1:1000,
                            ~Itai(N, K, "normal", mean = m, sd = 0.2)) %>%
                               sum()/1000
                  })
  ) %>%
  ggplot(aes(x = mean, y = percent)) +
    geom_line() +
    scale_y_continuous(labels = scales::percent, limits = c(0,1)) +
    labs(title = paste0("N = ", N, "; K = ", K))
}
```

```
( MonteCarlo(2, 1) +
  MonteCarlo(20, 1) )/
( MonteCarlo(2, 5) +
  MonteCarlo(20, 5) ) +
plot_annotation(title = "Percent of times the two aggregation procedures agree",
                caption = "Probabilities drawn from a normal distribution with sd = 0.2")
```



Probabilities drawn from a normal distribution with sd = 0.2