BE_PCLT_report

Lei Shi

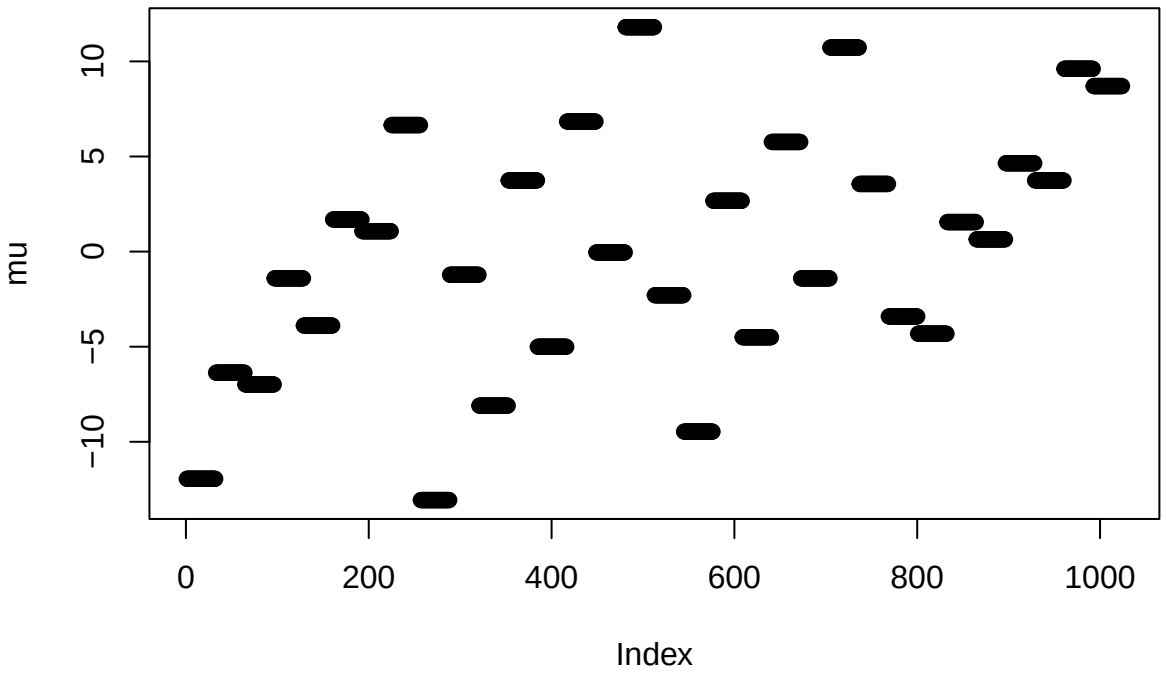
2022-08-17

Experiment setup

- K = 10, number of factors
- Q_U = 660, number of unreplicated arms, each with $N_q = 1$
- Q_R = 350, number of small replicated arms, each with $N_q = 2$
- Q_L = 14, number of large arms, each with $N_q = 30$.
- N = 1780, population

Generate data such that all the k-way ($k \geq 3$) interactions are zero. The mean for treatment arm q is a function of $F_6, \ldots, F_{10}$. The logic is that we assume $F_1, \ldots, F_5$ are less important factors and $F_6, \ldots, F_{10}$ are important ones. The pattern of the means: (the axis 'Index' is the arms ordered in lexicographical order)

```
plot(mu)
```



The factorial effects in the first two levels:

```
tau[abs(tau)>1e-4]
```

```
##        F6        F7        F8        F9       F10     F6.F9    F6.F10
##  1.167904  2.481268  4.029012  1.007448  1.639198  1.565897 -3.187514
```

The true target effects and true variance for the WLS estimator:

```
# population level:
## tau
cat("target_tau\n")
```

```
## target_tau
```

```
target_tau
```

```
##         [,1]
## F2  0.000000
## F4  0.000000
## F6  1.167904
## F8  4.029012
## F10 1.639198
```

```
cat("\n")
```

```
## variance for the estimator
cat("true variance\n")
```

```
## true variance
```

```
true_cov_tauhat <- t(as.matrix(target_design)) %*% true_cov_Yhat %*% as.matrix(target_design) / 1024^2
true_cov_tauhat
```

```
##                 F2            F4            F6            F8           F10
## F2   1.968737e-03  1.654976e-05 -4.117173e-05  1.778187e-05 -1.982575e-05
## F4   1.654976e-05  1.968551e-03 -6.562823e-05  4.423916e-05  3.853443e-05
## F6  -4.117173e-05 -6.562823e-05  1.968537e-03 -1.103128e-04  4.954102e-05
## F8   1.778187e-05  4.423916e-05 -1.103128e-04  1.968650e-03 -6.851526e-05
## F10 -1.982575e-05  3.853443e-05  4.954102e-05 -6.851526e-05  1.968592e-03
```

```
cat("\n")
```

```
## True standard deviation
cat("true sd\n")
```

```
## true sd
```

```
sqrt(diag(true_cov_tauhat))
```

```
##         F2         F4         F6         F8        F10
## 0.04437045 0.04436836 0.04436820 0.04436947 0.04436882
```

```
cat("\n")
```

Target effects we want to estimate: 'F2', 'F4', 'F6', 'F8', 'F10'

Numeric experiments

In the first experiments, we run 1000 MC trials and report:

- histogram of the point estimates
- estimated standard deviation for 5 methods (see later parts on what these methods are)
- 95%-CI coverage
- 95% Wald CI coverage

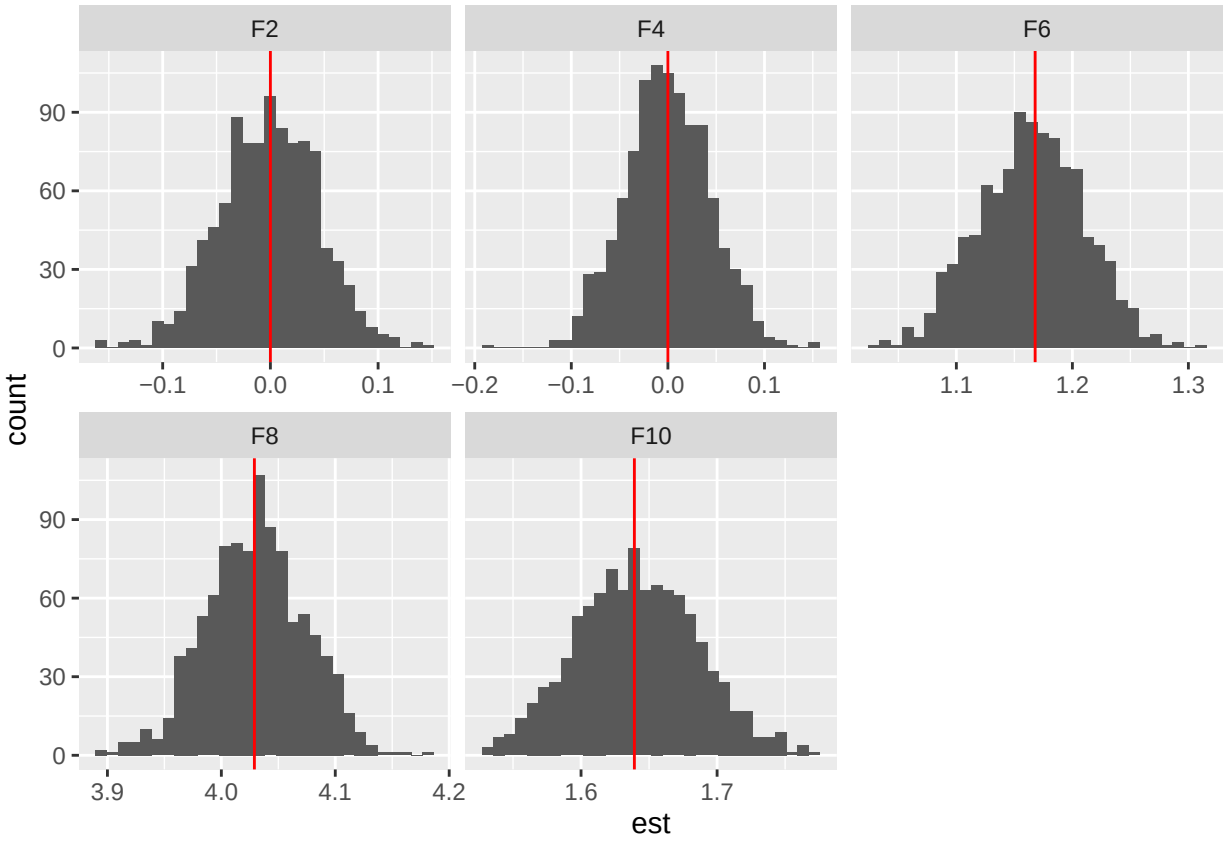
Distribution of point estimates

```
# report results
record <- readRDS("record_OUTCOME.RData")
# point estimates
hist_data <- data.frame(
  est = c(t(record$rec_point_est)),
  labs = factor(rep(target_effect, each = 1000), levels = c('F2', 'F4', 'F6', 'F8', 'F10'))
)

summary_data <- data.frame(
  target_tau = target_tau,
  labs = factor(target_effect, levels = c('F2', 'F4', 'F6', 'F8', 'F10'))
)

ggplot(hist_data, aes(x = est)) +
  facet_wrap(~labs, scales = 'free_x') +
  geom_histogram() +
  geom_vline(data = summary_data, mapping = aes(xintercept = target_tau), col = 'red') # red lines are
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Takeaways:

- CLT holds even the design is highly non-uniform.

Expectation of the sd estimators

We applied five methods for variance estimation:

- wls_0: wls + homoskedastic var, with incorrect specification: $Y \sim F2 + F4 + F6 + F8 + F10$
- ehw_0: wls + hc2 var, with incorrect specification: $Y \sim F2 + F4 + F6 + F8 + F10$
- wls_1: wls + homoskedastic var, with correct specification; $Y \sim F2 + F4 + F6 + F7 + F8 + F9 + F10 + F6.F9 + F6.F10$
- ehw_1: wls + hc2 var, with correct specification; $Y \sim F2 + F4 + F6 + F7 + F8 + F9 + F10 + F6.F9 + F6.F10$
- lex: lexicographical pairing

```
# expectation of sd estimator
print(data.frame(
  true_sd   = sqrt(diag(true_cov_tauhat)),
  wls_0_sd  = diag(apply(sqrt(record$rec_var_est_wls_0), MARGIN = c(1,2), sum))/1000,
  ehw_0_sd  = diag(apply(sqrt(record$rec_var_est_ehw_0), MARGIN = c(1,2), sum))/1000,
  wls_1_sd  = diag(apply(sqrt(record$rec_var_est_wls_1), MARGIN = c(1,2), sum))/1000,
  ehw_1_sd  = diag(apply(sqrt(record$rec_var_est_ehw_1), MARGIN = c(1,2), sum))/1000,
  lex_sd  = diag(apply(sqrt(record$rec_var_est_lex), MARGIN = c(1,2), sum))/1000
))
```

```
## Warning in sqrt(record$rec_var_est_ehw_0): NaNs produced
```

```
## Warning in sqrt(record$rec_var_est_ehw_1): NaNs produced
```

```
## Warning in sqrt(record$rec_var_est_lex): NaNs produced
```

```
##        true_sd  wls_0_sd  ehw_0_sd   wls_1_sd   ehw_1_sd     lex_sd
## F2  0.04437045 0.1119123 0.1354369 0.03706802 0.04436812 0.06308728
## F4  0.04436836 0.1119123 0.1354369 0.03706802 0.04436812 0.06308728
## F6  0.04436820 0.1119123 0.1354369 0.03706802 0.04436812 0.06308728
## F8  0.04436947 0.1119123 0.1354369 0.03706802 0.04436812 0.06308728
## F10 0.04436882 0.1119123 0.1354369 0.03706802 0.04436812 0.06308728
```

```
# true sd's:
# 0.04368610 0.04368584 0.04368498 0.04368678 0.04368549 0.04368564
```

Takeaways:

- wls + homoskedastic var: tends to underestimate the variance. `wls_0` is less conservative than `ehw_0` is not theoretically guaranteed. It is very likely to be a coincidence due to the underestimation habit of (wls + homoskedastic var).
- wls + ehw: when the correct model is included in the specification, performance is very nice. When misspecification happens, still robust but can be very conservative.
- lex: works fair in general. A little bit more conservative than ehw_1 but less conservative than ehw_0.

CI coverage

```
# CI coverage
print(data.frame(
  target_effect = target_effect,
  wls_0_coverage  = rowSums(record$rec_coverage_wls_0)/1000,
  ehw_0_coverage  = rowSums(record$rec_coverage_ehw_0)/1000,
  wls_1_coverage  = rowSums(record$rec_coverage_wls_1)/1000,
  ehw_1_coverage  = rowSums(record$rec_coverage_ehw_1)/1000,
  lex_coverage  = rowSums(record$rec_coverage_lex)/1000
))
```

```
##   target_effect wls_0_coverage ehw_0_coverage wls_1_coverage ehw_1_coverage
## 1            F2              1              1          0.894          0.947
```

```
## 2            F4              1              1          0.889          0.954
## 3            F6              1              1          0.889          0.959
## 4            F8              1              1          0.909          0.956
## 5           F10              1              1          0.895          0.957
##   lex_coverage
## 1        0.990
## 2        0.996
## 3        0.995
## 4        0.994
## 5        0.999
```

Wald inference

```
# wald inference
print(data.frame(
  method = c('wls_0', 'ehw_0', 'wls_1', 'ehw_1', 'lex'),
  wald_coverage = c(sum(record$rec_wald_wls_0)/1000,
                    sum(record$rec_wald_ehw_0)/1000,
                    sum(record$rec_wald_wls_1)/1000,
                    sum(record$rec_wald_ehw_1)/1000,
                    sum(record$rec_wald_lex)/1000)
))
```

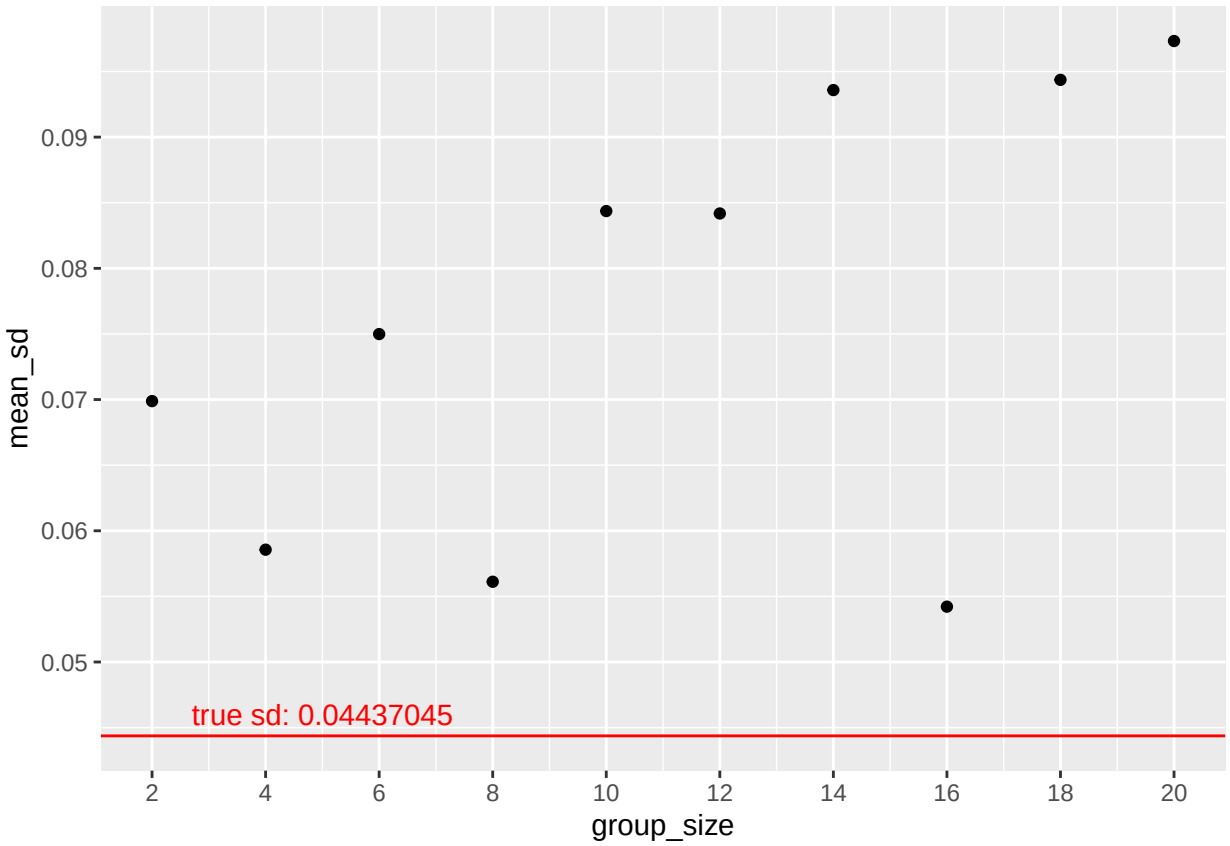
```
##   method wald_coverage
## 1  wls_0         1.000
## 2  ehw_0         1.000
## 3  wls_1         0.835
## 4  ehw_1         0.946
## 5    lex         1.000
```

How does the performance change for lex as the size of the groups vary?

In the first experiment, we tried pairing based on lexicographical order, which is inherently group size $|g| = 2$. In this experiment we vary the group size: $|g| = 2 * k, k = 1, \cdots, 10$ and make comparison over 500 MC runs.

```
record_vary_lex <- readRDS("record_OUTCOME_VARY_LEX.RData")

group_size <- 2*(1:10)
mean_sd <- apply(record_vary_lex$rec_var_est_lex, MARGIN = c(1,2,4), function(x){mean(sqrt(max(x,0)))})
mean_sd <- mean_sd[1,1,]
plot_df <- data.frame(
  mean_sd = mean_sd,
  group_size = group_size
  )
ggplot(plot_df, aes(x=group_size, y= mean_sd)) +
  geom_point() +
  geom_hline(yintercept = 0.04437045, col = 'red') +
  annotate(geom="text", x=5, y=0.046, label="true sd: 0.04437045", color="red") +
  scale_x_continuous(breaks = 2*(1:10))
```



Note that when group size equals 2^k, the performance is good. This is because the data generating process induces averages that are clustered in segments of size 32. See the plot at the beginning of this pdf file showing the pattern of mu. Therefore, grouping by 2^k will not cause any "breaks" in the means. As k get large, one can approximate the group-wise mean better, hence the variance estimation gets less conservative. However, if group size $\neq 2^k$, using smaller groups leads to better performance.

For grouping, our proposal adds some finite sample correction which targets the worst case scenario: the potential outcomes have high correlation (think about the ϱ_g we defined in the theory). Maybe in some cases smaller correction factors also suffice for robustness and can reduce the conservativeness. But we need stronger assumptions.

Overall, there is no panacea for variance estimation under unreplicated designs. One needs more information such as knowledge about DGP or additional covariates. If we know them, grouping can help a lot to handle the embarrassment in unreplicated experiments.

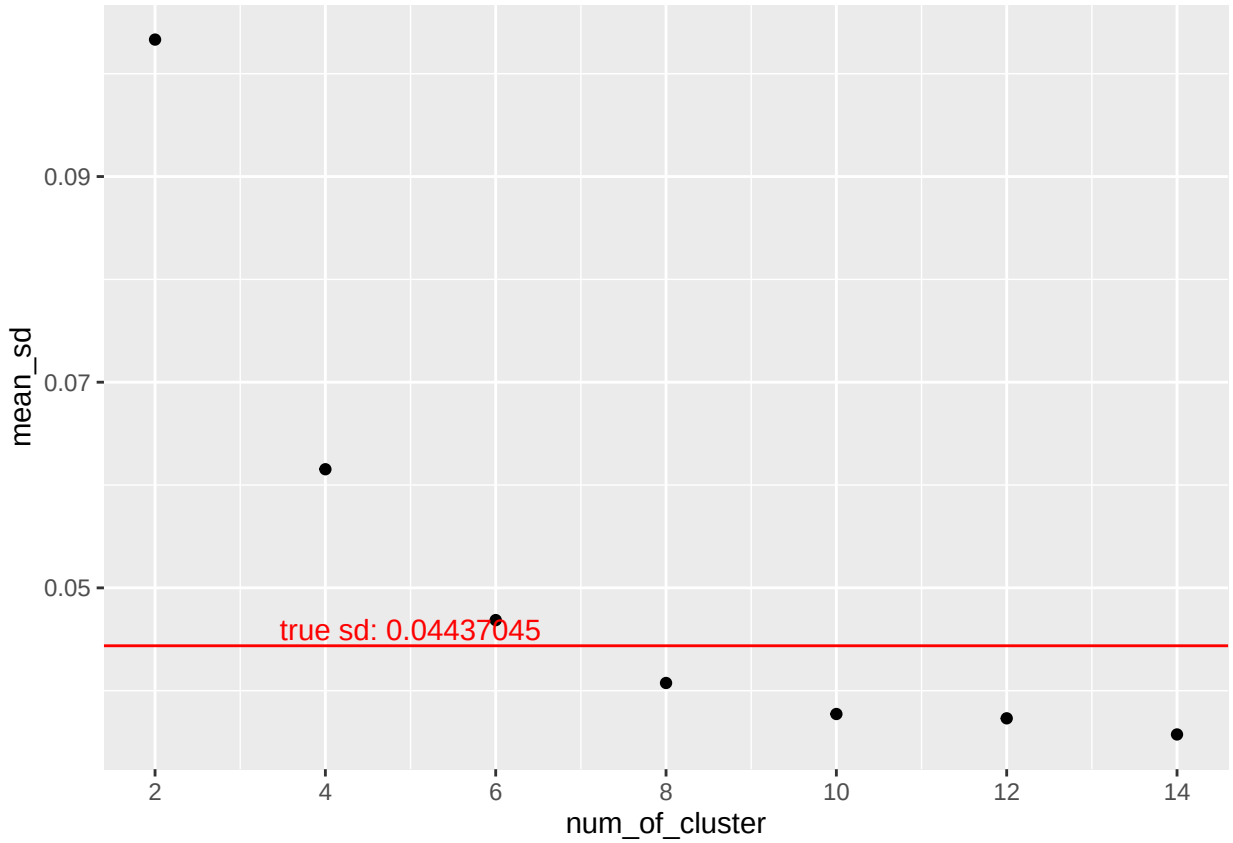
Can we do clustering?

In this part we try clustering based on outcomes. We use `kmeans` from R base library. The only tuning parameter is the number of centers. We try $number\ of\ centers = 2k, k = 1:7$, each with 500 MC runs.

```
record_vary_km <- readRDS("record_OUTCOME_VARY_KM.RData")

num_of_cluster <- 2*(1:7)
mean_sd <- apply(record_vary_km$rec_var_est_km, MARGIN = c(1,2,4), function(x){mean(sqrt(max(x,0)))})
mean_sd <- mean_sd[1,1,]
plot_df <- data.frame(
  mean_sd = mean_sd,
  num_of_cluster = num_of_cluster
  )
```

```
ggplot(plot_df, aes(x=num_of_cluster, y= mean_sd)) +
  geom_point() +
  geom_hline(yintercept = 0.04437045, col = 'red') +
  annotate(geom="text", x=5, y=0.046, label="true sd: 0.04437045", color="red") +
  scale_x_continuous(breaks = 2*(1:7))
```



Small number of clusters lead to high conservativenss, while large number of clusters under-estimate the variance. In general it is unknown to decide a proper choice. Besides the theory is not clear if one use the outcomes twice.