Blocking

Ryan T. Moore

American University

The Lab @ DC

2024-06-11

Table of contents I

Why?

Why do we block?

- Covariate **balance**

Why do we block?

- Covariate **balance**
- Estimate **closer to truth**

Why do we block?

- Covariate **balance**
- Estimate **closer to truth**
- Increased **efficiency**

Why do we block?

- Covariate **balance**
- Estimate **closer to truth**
- Increased **efficiency**
- Triply-robust estimates: block, randomize, adjust

Why do we block?

- Covariate **balance**
- Estimate **closer to truth**
- Increased **efficiency**
- Triply-robust estimates: block, randomize, adjust
- Block-level effects
  $\rightsquigarrow$ different actors interested in different effects

Why do we block?

- Covariate **balance**
- Estimate **closer to truth**
- Increased **efficiency**
- Triply-robust estimates: block, randomize, adjust
- Block-level effects

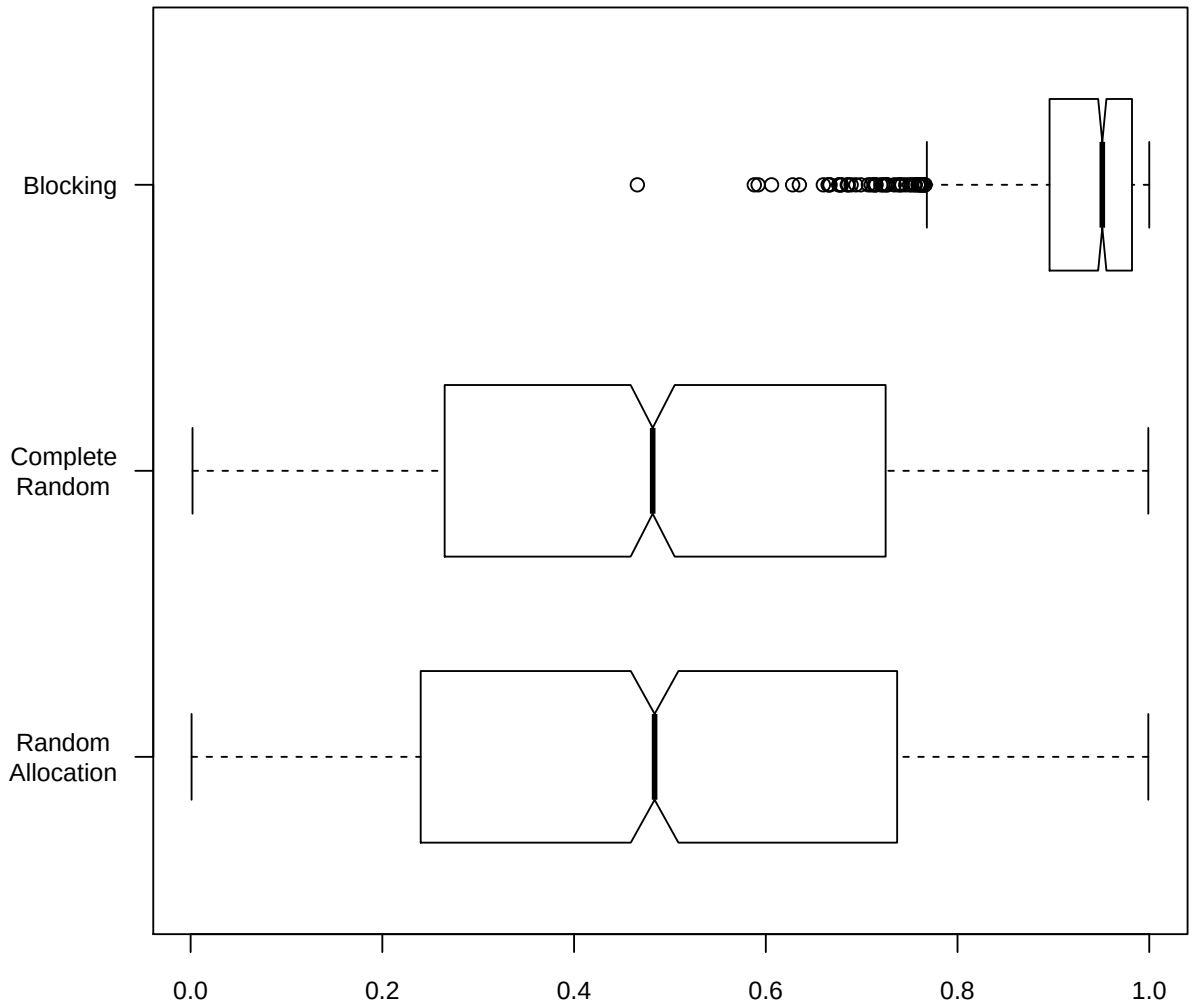
  $\rightsquigarrow$ different actors interested in different effects
- Guidelines for limited/uncertain resources

Why Block: Balance

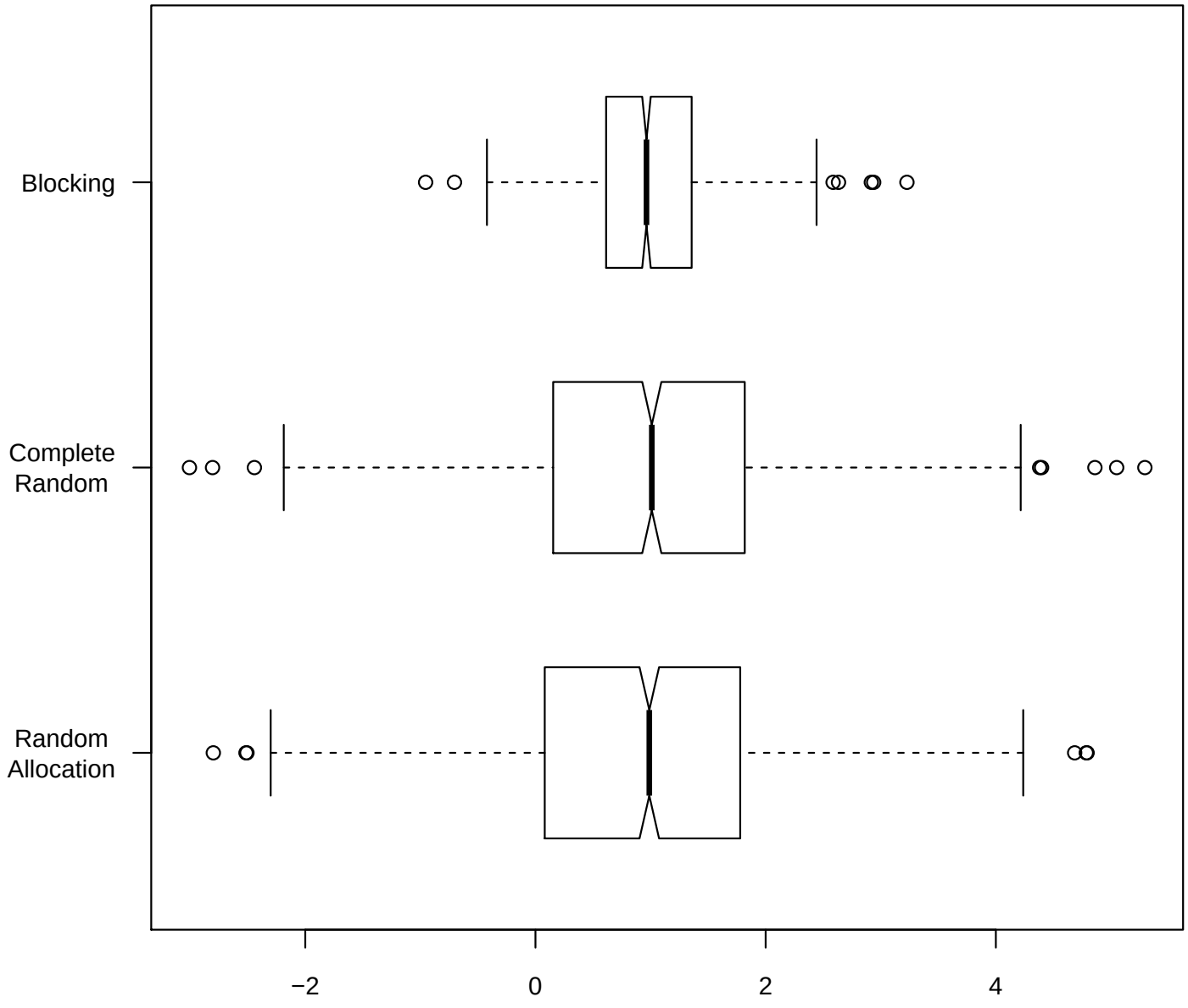
Simulation study: 100 units, $X_1 \sim N(0,1)$, $X_2 \sim \text{Unif}(0,1)$, $X_3 \sim \chi^2_2$; 1000 such experiments. Assg treatmnt in 3 ways.

Why Block: Balance

Simulation study: 100 units, $X_1 \sim N(0,1)$, $X_2 \sim \text{Unif}(0,1)$, $X_3 \sim \chi^2_2$; 1000 such experiments. Assg treatmnt in 3 ways.



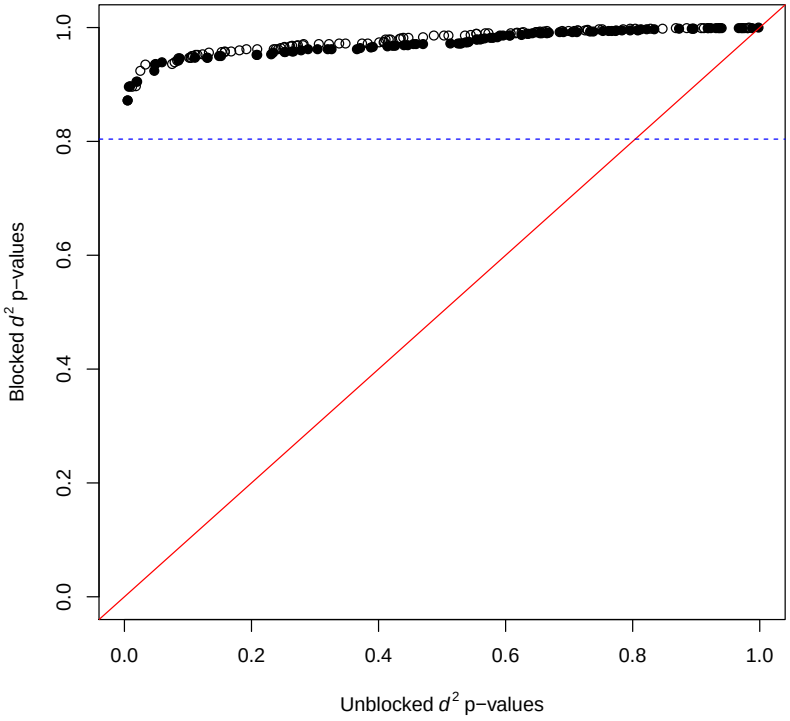
Why Block: Efficiency

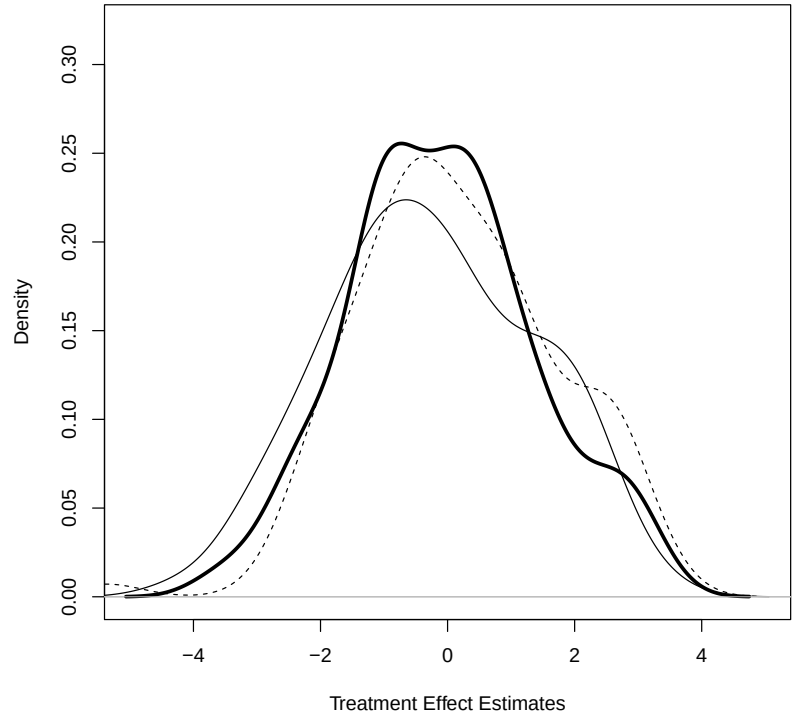


Blocking in Applications: Balance and Efficiency

Moore (2012): Perry Preschool Experiment

Left: QQ plot of balance (100 blocked vs. unblocked)
Right: Est TE under sharp null (100 blocked vs. unblocked)

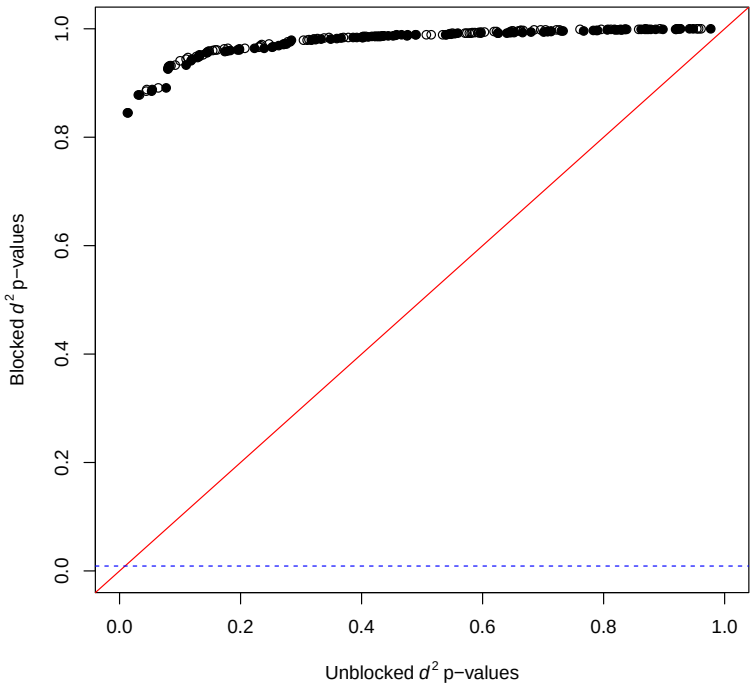


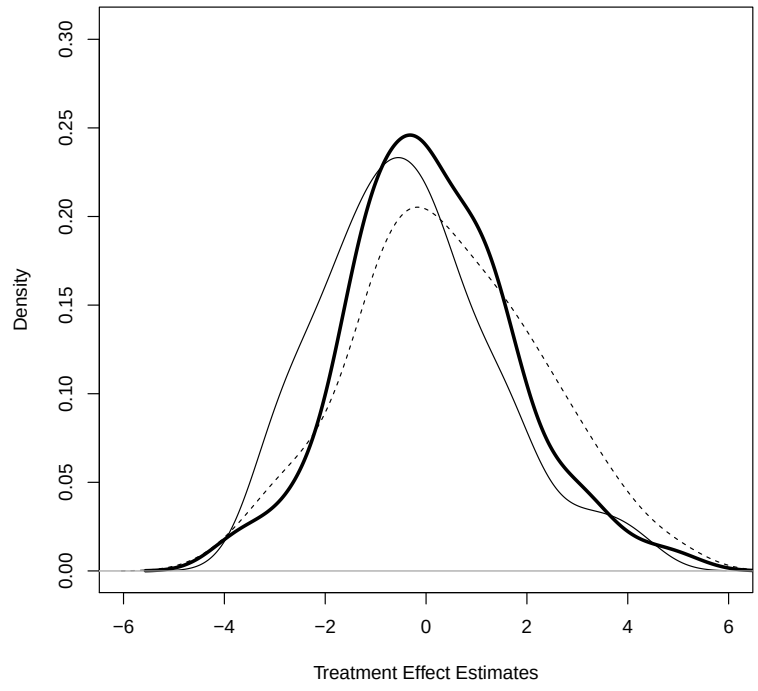


(SES, sex, IQ)

Balance in Applications: Balance and Efficiency

Considering more variables ...





(+ siblings, AFDC, mom empl, educ, father, ...)

How: `blockTools`

Blocking with `blockTools`

Start with some sample data:

```
library(blockTools)
data(x100)

x100 |> head()
```

```
    id id2  b1  b2 g  ig
1 1001 101 156 795 b 729
2 1002 102 813 469 a 627
3 1003 103 950 978 a 959
4 1004 104 991 781 a 661
5 1005 105 613 759 a 819
6 1006 106 654 838 b 643
```

(Moore 2012; Moore and Schnakenberg 2023)

Blocking with blockTools

```
b <- block(x100, id.vars = "id",
           block.vars = c("b1", "b2"))
```

```
bl <- b$blocks$`1`

bl |> head()
```

```
  Unit 1 Unit 2   Distance
1   1043   1040 0.01240000
2   1100   1020 0.02259275
3   1065   1027 0.02912651
4   1085   1081 0.03498815
5   1088   1061 0.04789253
6   1064   1014 0.07985116
```

Blocking with `blockTools`

Why all this?

```
bl <- b$blocks$`1`
```

We are extracting just the blocked pairs themselves.

- Why `b$blocks`? Since `b` has 3 components:

```
names(b)
```

```
[1] "blocks"    "level.two" "call"
```

- Why `blocks$1`‘? Since this is (default-named) first (and only) “group”:

```
names(b$blocks)
```

```
[1] "1"
```

Blocking with blockTools

What else could we do?

```
b_3groups_3conditions <- block(
  x100,
  groups = "g",                    # (Factor variable in data)
  n.tr = 3,
  id.vars = "id",
  block.vars = c("b1", "b2"),
  distance = "mve"
)
```

```
b_3groups_3conditions$blocks
```

```
$a
   Unit 1 Unit 2 Unit 3 Max Distance
1    1076   1039   1056    0.2443719
2    1084   1058   1017    0.4073681
3    1073   1029   1098    0.4211638
4    1059   1046   1081    0.4302601
```

Blocking with `blockTools`

Some rows from each "group":

```
rows_a <- b_3groups_3conditions$blocks$a |> slice(1:2) |> mutate
rows_b <- b_3groups_3conditions$blocks$b |> slice(1:2) |> mutate
rows_c <- b_3groups_3conditions$blocks$c |> slice(1:2) |> mutate
bind_rows(rows_a, rows_b, rows_c)
```

```
  Unit 1 Unit 2 Unit 3 Max Distance group
1   1076   1039   1056    0.2443719     a
2   1084   1058   1017    0.4073681     a
3   1043   1040   1009    0.1744377     b
4   1048   1031   1062    0.2444493     b
5   1095   1092   1049    0.3473709     c
6   1088   1027   1066    0.3565855     c
```

Blocking with `blockTools`

Other arguments to `block()`

- `vcov.data`
- `groups`: for exact-blocks
- `n.tr`
- `id.vars`
- `block.vars`
- `algorithm`: `optGreedy`, `optimal`, `naiveGreedy`, `randGreedy`, `sortGreedy`
- `distance`: `mahalanobis`, `mcd`, `mve`, `euclidean`, $k \times k$ matx
- `weight`
- `level.two`: block states by most similar cities
- `valid.var`, `valid.range`: Goldilocks
- `seed.dist`: (for `mcd` and `mve`)

Assign

```
a <- assignment(b, seed = 71573706)
a
```

```
Assignments:

    Treatment 1  Treatment 2  Distance
1    1040         1043        0.01240000
2    1100         1020        0.02259275
3    1065         1027        0.02912651
4    1081         1085        0.03498815
5    1088         1061        0.04789253
6    1014         1064        0.07985116
7    1032         1070        0.08279625
8    1097         1098        0.08882421
9    1038         1018        0.09316331
10   1031         1048        0.10391953
11   1084         1058        0.10835825
```

Get Assignments

```
a |> extract_conditions(x100, id.var = "id")
```

```
  [1] 2 1 2 2 2 2 1 2 2 1 1 1 1 1 1 1 2 2 1 2 2 2 1 2 1 1 2
 [38] 1 1 1 2 2 2 2 2 2 1 2 1 2 2 2 2 1 1 2 1 2 1 1 2 2 1 2
 [75] 1 2 2 2 1 2 1 1 1 1 2 1 2 1 1 2 1 2 1 1 2 2 1 2 1 1
```

```
x100 |> mutate(
  condition = extract_conditions(a, x100, id.var = "id"))
```

```
      id id2  b1   b2 g  ig condition
1   1001 101 156  795 b 729         2
2   1002 102 813  469 a 627         1
3   1003 103 950  978 a 959         2
4   1004 104 991  781 a 661         2
5   1005 105 613  759 a 819         2
6   1006 106 654  838 b 643         2
7   1007 107 640  645 c  12         1
8   1008 108 681  404 a 221         2
```

Assign 3 Conditions, within Groups

```
a3 <- assignment(b_3groups_3conditions, seed = 979677744)
a3
```

```
Assignments:

Group: a
    Treatment 1  Treatment 2  Treatment 3  Max Distance
1    1056         1076         1039         0.2443719
2    1017         1058         1084         0.4073681
3    1029         1073         1098         0.4211638
4    1046         1081         1059         0.4302601
5    1065         1002         1061         0.4417152
6    1060         1067         1004         0.6252877
7    1054         1052         1030         0.8214195
8    1026         1068         1024         1.0455063
9    1089         1008         1091         1.2872340
10   1075         1016         1036         1.3282637
```

How: `randomizr`

Blocking with `randomizr::block_ra()`

```r
library(randomizr)

tr <- block_ra(x100$g)

# Better:

x100 |> mutate(tr = block_ra(x100$g))
```

```
      id id2  b1   b2 g  ig tr
1   1001 101 156  795 b 729  1
2   1002 102 813  469 a 627  1
3   1003 103 950  978 a 959  1
4   1004 104 991  781 a 661  1
5   1005 105 613  759 a 819  0
6   1006 106 654  838 b 643  1
7   1007 107 640  645 c  12  1
8   1008 108 681  404 a 221  0
9   1009 109 530  823 b 321  1
```

Then what?

`blockTools`: diagnose, get block IDs, check balance

Diagnose:

```
diagnose(a, data = x100, id.vars = "id",
         suspect.var = "b1", suspect.range = c(0, 5))
```

`blockTools`: diagnose, get block IDs, check balance

Diagnose:

```
diagnose(a, data = x100, id.vars = "id",
         suspect.var = "b1", suspect.range = c(0, 5))
```

Get block IDs

```
createBlockIDs(a, data = x100, id.var = "id")
```

`blockTools`: diagnose, get block IDs, check balance

Diagnose:

```
diagnose(a, data = x100, id.vars = "id",
         suspect.var = "b1", suspect.range = c(0, 5))
```

Get block IDs

```
createBlockIDs(a, data = x100, id.var = "id")
```

Get balance:

```
assg2xBalance(a, x100, id.var = "id",
              bal.vars = c("b1", "b2"))
```

Analysis

- Generally, use Lin or Blocked Diff-in-Means

Analysis

- Generally, use Lin or Blocked Diff-in-Means
- LSDV (block indicators) weights w/in block TE's by

$$p_j(1-p_j)n_j$$

Analysis

- Generally, use Lin or Blocked Diff-in-Means
- LSDV (block indicators) weights w/in block TE's by

$$p_j(1-p_j)n_j$$

Analysis

- Generally, use Lin or Blocked Diff-in-Means
- LSDV (block indicators) weights w/in block TE's by

$$p_j(1-p_j)n_j$$

where

- p_j = share of block j treated
- n_j = size of block j
- (I.e., $p_j(1-p_j) = \text{var}(TE)$ in block j)

Analysis

Can I just ignore blocks and pool?

Analysis

Can I just ignore blocks and pool?

- If p_j varies, no

Thanks!

rtm@american.edu
www.ryantmoore.org

References I

Coppock, Alexander. 2023. *randomizr: Easy-to-Use Tools for Common Forms of Random Assignment and Sampling.* https://CRAN.R-project.org/package=randomizr.

Moore, Ryan T. 2012. "Multivariate Continuous Blocking to Improve Political Science Experiments." *Political Analysis* 20 (4): 460–79. https://doi.org/10.1093/pan/mps025.

Moore, Ryan T., and Keith Schnakenberg. 2023. *blockTools: Blocking, Assignment, and Diagnosing Interference in Randomized Experiments.* http://www.ryantmoore.org/html/software.blockTools.html.