`blockTools`: Blocking, Assignment, and Diagnosing Interference in Randomized Experiments

Ryan T. Moore
rtmoore@fas.harvard.edu

Department of Government and Institute for Quantitative Social Science
Harvard University (Cambridge, Massachusetts)

Ames, IA
10 August 2007

What is *blocking*?

Blocking sorts experimental units into (homogeneous) sets prior to randomization to treatment conditions. Consider an experiment with 6 units and 3 treatments; a "randomized complete block" design sorts units into 2 blocks of 3 units each, then assigns one unit per block to each of the 3 treatments:

A					Tr 1	Tr 2	Tr 3
	B	*B*		$\Rightarrow$	*A*	*A*	*A*
B		*A*	*A*		*B*	*B*	*B*

Why block?

- Improve causal estimate efficiency
- Reduce causal estimate error from covariate imbalance
- Calculate or weight block-level causal estimates
- Define *ex ante* procedures for robustness to non-compliance

Why randomize?

Randomizing units to experimental conditions implies that all confounders, measured and unmeasured, observable and unobservable, are distributed identically in different treatment conditions. Blocking protects against "bad randomizations" on measured confounders.

`blockTools` at Work: Application to Simulated Data

Chained together, `blockTools`' three primary functions perform the stages of experimental design. I illustrate `block()`, `assignment()`, and `diagnose()` using simulated data included in `blockTools`. Variables `id` and `id2` identify units, `b1` and `b2` are substantive blocking variables, and `g` represents the unit's group. For a matched pair design within groups,

```
> bl.out <- block(data = x, groups = "g", id.vars = "id",
+   block.vars = c("b1", "b2"))
> bl.out
   Unit 1 Unit 2  Distance
1    1084   1058     0.108
2    1076   1039     0.163
3    1065   1061     0.176
...
```

Another example of `block()`, changing some arguments:

```
> bl.out <- block(data = x, groups = "g", n.tr = 3, id.vars =
+   c("id", "id2"), block.vars = c("b1", "b2"), algorithm =
+   "naiveGreedy", distance = "mve", level.two = T,
+   valid.var = "b1", valid.range = c(100, 300))
> bl.out
  Unit 1 Subunit 1 Unit 2 Subunit 2 Unit 3 Subunit 3 Max Dist
    1076       176   1024       124   1068       168    0.839
    1081       181   1032       132   1091       191    0.941
    1059       159   1016       116   1046       146    1.263
...
```

Other optional arguments to `block()` include

- `n.tr`, the number of treatment conditions
- `algorithm`, blocking proceeds as `optGreedy`, `naiveGreedy`, `sortGreedy`, or `randGreedy`
- `distance`, between-unit distance defined as `mahalanobis`, `mcd`, or `mve`
- `vcov.data`, a user-defined covariance matrix for the blocking variables
- `level.two`, a logical allowing units to be matched by best subunits
- `valid.var`, a variable to define valid range of possible matches, to prevent within-block interference

Assignment proceeds after blocking:

```
> assg.out <- assignment(bl.out, seed = 123)
> assg.out
  Tr 1 Tr 1 Tr 2 Tr 2 Tr 3 Tr 3  Max Dist
  1076  176 1024  124 1068  168     0.839
  1091  191 1032  132 1081  181     0.941
  1016  116 1046  146 1059  159     1.263
...
```

Two `blockTools` functions write `block()` and `assignment()` output objects to .tex and .csv files,creating one file for each group:

```
> outTex(assg.out)
> outCSV(assg.out)
```

Taking Interference Seriously

Social science field experiments evaluate interventions such as universal health insurance (King et al., 2007), national party platforms (Wantchekon, 2003), get-out-the-vote drives (Gerber and Green, 2000), and more.

Interference is a concern in all these cases. Interference occurs when the potential outcomes of unit i under control and treatment, (y_{i0}, y_{i1}), are affected by the treatment assignment of at least one other unit j, t_j. Formally,

$$[(y_{i0}, y_{i1})|t_j = 0] \neq [(y_{i0}, y_{i1})|t_j = 1] \qquad \text{for some } i,\ j$$

Experimenters often want units to be physically near one another to encourage similarity of background covariates, but not *too near* such that interference occurs. Sobel (2006) shows that ignoring interference results in interpreting non-causal quantities as causal effects.

Rosenbaum (2007) describes valid inference under interference, but experimenters often prefer avoiding interference via constraints on the selection, blocking, or assignment of proximate units.

Diagnosing Interference

After assignment, `diagnose()` identifies possible interference, also called "contamination", "diffusion", or "unit non-compliance":

```
> diagnose(assg.out, data = x, id.vars=c("id", "id2"),
+   suspect.var = "b2", suspect.range = c(0,50))
Tr 1  Tr 1  Tr 2  Tr 2 Difference
1026   126  1002   102         40
1005   105  1004   104         22
1030   130  1004   104         13
...
```


Gerber, Alan S. and Donald P. Green. 2000. "The Effects of Canvassing, Telephone Calls, and Direct Mail on Voter Turnout: A Field Experiment." *American Political Science Review* 94(3):653–663.

King, Gary, Emmanuela Gakidou, Nirmala Ravishankar, Ryan T. Moore, Jason Lakin, Manett Vargas, Martha María Téllez-Rojo, Juan Eugenio Hernández Ávila, Mauricio Hernández Ávila and Héctor Hernández Llamas. 2007. ""A 'Politically Robust' Experimental Design for Public Policy Evaluation, with Application to the Mexican Universal Health Insurance Program"." *Journal of Policy Analysis and Management* 26(3):479–509.

Rosenbaum, Paul R. 2007. "Interference Between Units in Randomized Experiments." *Journal of the American Statistical Association* 102(477):191–200.

Sobel, Michael E. 2006. "What Do Randomized Studies of Housing Mobility Demonstrate?: Causal Inference in the Face of Interference." *Journal of the American Statistical Association* 101(476):1398–1407.

Wantchekon, Leonard. 2003. "Clientelism and Voting Behavior: Evidence from a Field Experiment in Benin." *World Politics* 55 (3):399–422.