# Matching within subgroups

*Ben B. Hansen, Mark Fredrickson, Josh Buckner, Josh Errickson, and Peter Solenberger, with embedded Fortran code due to Dimitri P. Bertsekas and Paul Tseng*

*2019-07-12*

When utilizing full matching, a user may want to introduce restrictions to the potential match sets. There are two main reasons to do this.

- There may be certain variables which matches should either always or never agree upon. For example, if there is a binary gender variable, you may wish to only match treatment members to control members of the same gender.
- When matching on a medium-to-large data set, speed concerns can become paramount. By splitting the matching problem into a series of smaller subproblems, we can realize substantial performance improvements.

## Combining matches

In optmatch 0.9-11 and above, `optmatch` objects can be easily combined to facilitate breaking a problem into smaller sub-problems and reconstituting a matched structure on the entire data set. To demonstrate this, let's consider the `infert` data set.

```
> data(infert)
> head(infert)
##   education age parity induced case spontaneous stratum pooled.stratum
## 1    0-5yrs  26      6       1    1           2       1              3
## 2    0-5yrs  42      1       1    1           0       2              1
## 3    0-5yrs  39      6       2    1           0       3              4
## 4    0-5yrs  34      4       2    1           0       4              2
## 5   6-11yrs  35      3       1    1           1       5             32
## 6   6-11yrs  36      4       2    1           1       6             36
```

The "case" variable indicates treatment (1) versus control (0) status. We'll want to match upon "age".

```
> table(infert$case)
##
##   0   1
## 165  83
> table(infert$education, infert$case)
##
##             0  1
##   0-5yrs    8  4
##   6-11yrs  80 40
##   12+ yrs  77 39
```

Due to the sample size, if we were to compute matches on the entire data set, the `fullmatch` call would generate a distance matrix of size $83 \times 65 = 13,695$. However, if we were instead to compute a match within each level of the "education" variable, we'd compute three different distance matrices, of total size $8 \times 4 + 80 \times 40 + 77 \times 39 = 6,235$, a reduction of 55%.

We'll do this by splitting the data within each match.

```
> f1 <- fullmatch(case ~ age, data = infert[infert$education == "0-5yrs", ])
> f2 <- fullmatch(case ~ age, data = infert[infert$education == "6-11yrs", ])
```

```
> f3 <- fullmatch(case ~ age, data = infert[infert$education == "12+ yrs", ])
> summary(f1)
## Structure of matched sets:
## 1:2
##   4
## Effective Sample Size:  5.3
## (equivalent number of matched pairs).
> summary(f2)
## Structure of matched sets:
##  1:1  1:2  1:3  1:4 1:5+
##   20    8    6    4    2
## Effective Sample Size:  49.4
## (equivalent number of matched pairs).
> summary(f3)
## Structure of matched sets:
##  1:1  1:2  1:3  1:4 1:5+
##   23    5    6    2    3
## Effective Sample Size:  47
## (equivalent number of matched pairs).
```

Some of the matched sets are quite large (1:5+) so let's put some restrictions.

```
> f2 <- fullmatch(case ~ age, data = infert[infert$education == "6-11yrs", ],
+                 max.controls = 4)
> f3 <- fullmatch(case ~ age, data = infert[infert$education == "12+ yrs", ],
+                 max.controls = 4)
> summary(f2)
## Structure of matched sets:
## 1:1 1:2 1:3 1:4
##  18  10   6   6
## Effective Sample Size:  49.9
## (equivalent number of matched pairs).
> summary(f3)
## Structure of matched sets:
## 1:1 1:2 1:3 1:4
##  20   6   7   6
## Effective Sample Size:  48.1
## (equivalent number of matched pairs).
```

Now we simply combine the three matches.

```
> fcombine <- c(f1, f2, f3)
> summary(fcombine)
## Structure of matched sets:
## 1:1 1:2 1:3 1:4
##  38  20  13  12
## Effective Sample Size:  103.4
## (equivalent number of matched pairs).
> infert$match <- fcombine
```

## Using the `within` argument

An alternative approach would be using the `within` argument and the `exactMatch` function to define subproblems.

```
> fwithin <- fullmatch(case ~ age, data = infert, max.controls = 4,
+                      within = exactMatch(case ~ education, data = infert))
> summary(fwithin)
## Structure of matched sets:
## 1:1 1:2 1:3 1:4
##  38  20  13  12
## Effective Sample Size:  103.4
## (equivalent number of matched pairs).
```

Observe that we obtain equivalent matched structure. A few notes comparing the two approaches:

1. When using the `within` argument, restrictions must be the same across subproblems. That is, `max.controls`, `min.controls` and `omit.fraction` will be equivalent. By running the subproblems separately, you can set different restrictions per subproblem. E.g.,

   ```
   > f1 <- fullmatch(z ~ x, data = d[d$group == 1, ], max.controls = 2)
   > f2 <- fullmatch(z ~ x, data = d[d$group == 2, ], min.controls = 1/3)
   > c(f1, f2)
   ```

2. While the matched structures will be equivalent between these two approaches (if the restrictions are the same across subproblems), the actual matched sets themselves may differ if you have observations of equal distance. In general this should not be considered a problem.