Optimal pair matching con R optmacht y RItools packages

Luis Maldonado

Mayo, 2018

1 Aplicación de pair matching

1. En este guía veremos como realizar matching de pares. Para ello vamos a ocupar los siguientes paquetes en R

```
> rm(list=ls())
> library(optmatch)
> library(RItools)
```

2. Vamos a trabajar con la base de datos plantas nucleares, la cual está incluida en el paquete optmatch. La variable *pr* es el tratamiento, en donde 1 indica si una planta es construida en un sitio con un planta ya existente y 0 si tenemos una planta en nuevo sitio. Para detalles de la base de datos ejecutemos las siguientes opciones

```
> data(nuclearplants)
> head(nuclearplants)
```

```
    cost  date t1 t2  cap pr ne ct bw cum.n pt
H 460.05 68.58 14 46  687  0  1  0  0    14  0
I 452.99 67.33 10 73 1065  0  0  1  0     1  0
A 443.22 67.33 10 85 1065  1  0  1  0     1  0
J 652.32 68.00 11 67 1065  0  1  1  0    12  0
B 642.23 68.00 11 78 1065  1  1  1  0    12  0
K 345.39 67.92 13 51  514  0  1  1  0     3  0
```

```
> # help("nuclearplants"), para detalles de la base de datos
```

3. El ejemplo más simple de matching is utilizar una distancia en base a solo una covariante. Por ejemplo, ocupemos la variable *cap* y veamos qué resulta si aplicamos un pairmatching.

4. Para ello, vamos a concentrarnos en un subgrupo de la muestra en la cual se excluye a las plantas construidas con especiales garantías (ojo, solo con fines didácticos, luego vamos a trabajar con la base completa)

```
> nuke.nopt <- subset(nuclearplants, pt==0)
> table(nuke.nopt$pr) #Checkear en numero de tratados y controles
```

```
 0  1
19  7
```

```
> pairmatch( pr~cap, data=nuke.nopt ) #pairmatching
```

```
   H    I    A    J    B    K    L    M    C    N    O    P    Q    R    S    T
<NA>  1.2  1.1  1.1  1.2 <NA>  1.3 <NA>  1.3  1.7 <NA> <NA>  1.4  1.5 <NA> <NA>
   U    D    V    E    W    F    X    G    Y    Z
 1.6  1.4 <NA>  1.5 <NA>  1.6 <NA>  1.7 <NA> <NA>
```

5. Note que el comando pairmatch genera una variable del mismo largo que las variables que hay en la base de datos. En esta nueva variable, los valores corresponden a las parejas que hemos creado. Además, la distancia generada es la de Mahalanobis pero solo sobre la variable cap.

6. Para ver un output más leible, podemos ver el siguiente output

```
> print( pairmatch(pr~cap, data=nuke.nopt ), grouped=T )

 Group Members
   1.1    A, J
   1.2    I, B
   1.3    L, C
   1.4    Q, D
   1.5    R, E
   1.6    U, F
   1.7    N, G
```

7. El último paso de nuestro primer ejemplo es ver información resumen sobre la calidad del matching y estimar ATE.

```
> pm <- pairmatch( pr~cap, data=nuke.nopt )
> print(pm, grouped=T)

 Group Members
   1.1    A, J
   1.2    I, B
   1.3    L, C
   1.4    Q, D
   1.5    R, E
   1.6    U, F
   1.7    N, G

> summary(pm)

Structure of matched sets:
1:1 0:1
  7  12
Effective Sample Size:  7
(equivalent number of matched pairs).

> summary(lm(cost~pr+pm, data=nuke.nopt))

Call:
lm(formula = cost ~ pr + pm, data = nuke.nopt)

Residuals:
    Min      1Q  Median      3Q     Max
-101.05  -49.69    0.00   49.69  101.05

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   554.20      80.57   6.879 0.000466 ***
pr            -12.86      56.97  -0.226 0.828936
pm1.2          -0.16     106.58  -0.002 0.998851
pm1.3        -183.03     106.58  -1.717 0.136740
pm1.4        -196.85     106.58  -1.847 0.114267
pm1.5         -54.54     106.58  -0.512 0.627121
pm1.6         141.36     106.58   1.326 0.232978
pm1.7         101.72     106.58   0.954 0.376709
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 106.6 on 6 degrees of freedom
  (12 observations deleted due to missingness)
Multiple R-squared:  0.747,        Adjusted R-squared:  0.4518
F-statistic:  2.53 on 7 and 6 DF,  p-value: 0.1391
```

8. También podemos crear matching con múltiples controles, por ejemplo, veamos tripletas

```
> tm <- pairmatch( pr~cap, controls=2, data=nuke.nopt )
> print(tm, grouped=T)

 Group Members
   1.1 I, A, Z
   1.2 J, B, V
   1.3 L, C, X
   1.4 Q, D, Y
   1.5 R, E, W
   1.6 S, U, F
   1.7 N, P, G

> summary(tm)

Structure of matched sets:
1:2 0:1
  7   5
Effective Sample Size:  9.33
(equivalent number of matched pairs).

> summary(lm(cost~pr+tm, data=nuke.nopt))

Call:
lm(formula = cost ~ pr + tm, data = nuke.nopt)

Residuals:
     Min       1Q   Median       3Q      Max
-209.140  -76.914    7.727  100.370  158.510

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   543.74      83.94   6.478 2.07e-05 ***
pr            -37.88      65.02  -0.583    0.570
tm1.2         194.15     114.68   1.693    0.114
tm1.3         -80.80     114.68  -0.705    0.493
tm1.4        -139.29     114.68  -1.215    0.246
tm1.5         -13.03     114.68  -0.114    0.911
tm1.6          59.76     114.68   0.521    0.611
tm1.7          36.08     114.68   0.315    0.758
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 140.5 on 13 degrees of freedom
  (5 observations deleted due to missingness)
Multiple R-squared:  0.4517,        Adjusted R-squared:  0.1565
F-statistic:  1.53 on 7 and 13 DF,  p-value: 0.2408
```

2 Distancias

1. Un aspecto clave de el tipo de matching que estamos viendo es la separación entre los pasos de creación de distancias y el matching mismo. En el paso de arriba, ambos pasos fueron juntados. Ahora los separamos. Primero, veamos la creación de la matriz de distancia y luego realizamos el matching sobre la base de esta matriz

```
> cap.dist <- match_on(pr~cap, data=nuke.nopt )
> cap.dist
```

```
An object of class "DenseMatrix"
         control
treatment         H          I          J          K           L         M
        A 1.8028437 0.00000000 0.00000000 2.62795475 1.158970968 2.8998121
        B 1.8028437 0.00000000 0.00000000 2.62795475 1.158970968 2.8998121
        C 0.6438728 1.15897097 1.15897097 1.46898378 0.000000000 1.7408412
        D 0.7488002 2.55164390 2.55164390 0.07631085 1.392672933 0.3481682
        E 1.7313023 0.07154142 0.07154142 2.55641333 1.087429550 2.8282707
        F 0.6724893 1.13035440 1.13035440 1.49760035 0.028616567 1.7694577
        G 0.6391033 1.16374040 1.16374040 1.46421435 0.004769428 1.7360717
         control
treatment         N         O         P         Q          R         S         T
        A 1.3020538 2.4085611 1.3115927 2.551644 0.07154142 1.0254270 1.3688258
        B 1.3020538 2.4085611 1.3115927 2.551644 0.07154142 1.0254270 1.3688258
        C 0.1430828 1.2495901 0.1526217 1.392673 1.08742955 0.1335440 0.2098548
        D 1.2495901 0.1430828 1.2400512 0.000000 2.48010248 1.5262169 1.1828181
        E 1.2305124 2.3370196 1.2400512 2.480102 0.00000000 0.9538856 1.2972844
        F 0.1716994 1.2782067 0.1812383 1.421289 1.05881298 0.1049274 0.2384714
        G 0.1383134 1.2448207 0.1478523 1.387904 1.09219898 0.1383134 0.2050854
         control
treatment          U         V         W         X          Y         Z
        A 1.04927413 0.1192357 0.7249530 1.3306704 2.51348848 0.3100128
        B 1.04927413 0.1192357 0.7249530 1.3306704 2.51348848 0.3100128
        C 0.10969684 1.2782067 0.4340179 0.1716994 1.35451751 1.4689838
        D 1.50236977 2.6708796 1.8266909 1.2209735 0.03815542 2.8616567
        E 0.97773271 0.1907771 0.6534116 1.2591290 2.44194706 0.3815542
        F 0.08108027 1.2495901 0.4054014 0.2003160 1.38313408 1.4403672
        G 0.11446627 1.2829761 0.4387874 0.1669300 1.34974808 1.4737532
Slot "call":
match_on(x = pr ~ cap, data = nuke.nopt)
```

```
> summary(pairmatch(cap.dist))

Structure of matched sets:
1:1 0:1
  7  12
Effective Sample Size:  7
(equivalent number of matched pairs).
```

2. Podemos estimar una Matriz de distancia con caliper de 2 desviaciones estándares sobre la variable cap del siguiente modo

```
> cap.dist + caliper(cap.dist,2)

       control
treated         H          I          J          K           L         M
      A 1.8028437 0.00000000 0.00000000        Inf 1.158970968       Inf
      B 1.8028437 0.00000000 0.00000000        Inf 1.158970968       Inf
      C 0.6438728 1.15897097 1.15897097 1.46898378 0.000000000 1.7408412
      D 0.7488002        Inf        Inf 0.07631085 1.392672933 0.3481682
      E 1.7313023 0.07154142 0.07154142        Inf 1.087429550       Inf
      F 0.6724893 1.13035440 1.13035440 1.49760035 0.028616567 1.7694577
      G 0.6391033 1.16374040 1.16374040 1.46421435 0.004769428 1.7360717
       control
treated         N         O         P         Q          R         S         T
      A 1.3020538       Inf 1.3115927       Inf 0.07154142 1.0254270 1.3688258
      B 1.3020538       Inf 1.3115927       Inf 0.07154142 1.0254270 1.3688258
      C 0.1430828 1.2495901 0.1526217 1.392673 1.08742955 0.1335440 0.2098548
      D 1.2495901 0.1430828 1.2400512 0.000000        Inf 1.5262169 1.1828181
```

```
      E 1.2305124       Inf 1.2400512      Inf 0.00000000 0.9538856 1.2972844
      F 0.1716994 1.2782067 0.1812383 1.421289 1.05881298 0.1049274 0.2384714
      G 0.1383134 1.2448207 0.1478523 1.387904 1.09219898 0.1383134 0.2050854
       control
treated          U         V         W         X          Y         Z
      A 1.04927413 0.1192357 0.7249530 1.3306704        Inf 0.3100128
      B 1.04927413 0.1192357 0.7249530 1.3306704        Inf 0.3100128
      C 0.10969684 1.2782067 0.4340179 0.1716994 1.35451751 1.4689838
      D 1.50236977       Inf 1.8266909 1.2209735 0.03815542       Inf
      E 0.97773271 0.1907771 0.6534116 1.2591290        Inf 0.3815542
      F 0.08108027 1.2495901 0.4054014 0.2003160 1.38313408 1.4403672
      G 0.11446627 1.2829761 0.4387874 0.1669300 1.34974808 1.4737532
```

3. Todo en un solo código

```
> pairmatch(cap.dist, within=caliper(cap.dist, 2), data=nuke.nopt)

   H    I    A    J    B    K    L    M    C    N    O    P    Q    R    S    T
<NA>  1.2  1.1  1.1  1.2 <NA>  1.3 <NA>  1.3  1.7 <NA> <NA>  1.4  1.5 <NA> <NA>
   U    D    V    E    W    F    X    G    Y    Z
 1.6  1.4 <NA>  1.5 <NA>  1.6 <NA>  1.7 <NA> <NA>
```

4. Ayuda para interpretar el output

```
> summary(pairmatch(cap.dist))

Structure of matched sets:
1:1 0:1
  7  12
Effective Sample Size:  7
(equivalent number of matched pairs).

> matched.distances(pm, match_on(pr~cap, data=nuke.nopt))

       1.1        1.2        1.3        1.4        1.5        1.6        1.7
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.08108027 0.13831341

> sum(matched.distances(pm, match_on(pr~cap, data=nuke.nopt)))

[1] 0.2193937

> summary(matched.distances(pm, match_on(pr~cap, data=nuke.nopt)))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00000 0.00000 0.00000 0.03134 0.04054 0.13830
```

5. El **tamaño muestral efectivo** es una medida que permite comparar la calidad de distintos matchings en términos del tamaño muestral efectivo de parejas equivalentes formadas por el matching y se obtiene del siguiente modo: 1) para cada match s (o cada estrato), se calcula la media armónica del número de sujetos tratados (m_{st}) y el número de controles (m_{ct}), $h(m_{st}, m_{sc}) = [(m_{st}^{-1} + (m_{sc}^{-1})/2]^{-1}$; 2) se suman las medias armónicas de los estratos para determinar el tamaño muestral efectivo. Si comparamos dos tipos machings, será mejor el que logre un tamaño muestral efectivo mayor.

3 Balance

1. La calidad de los matchs no solo se evalua en base al tamaño muestral efectivo sino también al balance logrado respecto de las covariantes. A continuación, veamos el balance pre-matching. Usaremos la función *xBalance* del paquete RItools

```
> args(xBalance)
```

```
function (fmla, strata = list(unstrat = NULL), data, report = c("std.diffs",
    "z.scores", "adj.means", "adj.mean.diffs", "adj.mean.diffs.null.sd",
    "chisquare.test", "p.values", "all")[1:2], stratum.weights = harmonic,
    na.rm = FALSE, covariate.scaling = NULL, normalize.weights = TRUE,
    impfn = median, post.alignment.transform = NULL)
NULL
```

```
> xBalance(pr ~ cap, data=nuke.nopt, report="all")

     strata unstrat
     stat      pr=0    pr=1 adj.diff adj.diff.null.sd std.diff    z
vars
cap         803.263 883.000  79.737           92.220   0.380   0.865
---Overall Test---
        chisquare df p.value
unstrat     0.748  1   0.387
---
Signif. codes:  0 '***' 0.001 '** ' 0.01 '*  ' 0.05 '.  ' 0.1 '   ' 1
```

Note las diferencias (las hay?) entre los siguientes dos análisis

```
> xBalance(pr ~ cap, data=nuke.nopt, report="adj.mean.diffs")

     strata  unstrat
     stat   adj.diff
vars
cap           79.737
```

```
> summary(lm(cap ~ pr, data=nuke.nopt))

Call:
lm(formula = cap ~ pr, data = nuke.nopt)

Residuals:
    Min      1Q  Median      3Q     Max
-353.00 -102.70  -12.26  178.25  326.74

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   803.26      48.10   16.70 1.03e-14 ***
pr             79.74      92.70    0.86    0.398
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 209.7 on 24 degrees of freedom
Multiple R-squared:  0.0299,        Adjusted R-squared:  -0.01052
F-statistic: 0.7398 on 1 and 24 DF,  p-value: 0.3982
```

2. Veamos balance post-matching.

```
> summary(lm(cap ~ pr + pm, data=nuke.nopt))

Call:
lm(formula = cap ~ pr + pm, data = nuke.nopt)

Residuals:
     Min       1Q   Median       3Q      Max
-13.6429  -0.8571   0.0000   0.8571  13.6429

Coefficients:
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.064e+03  7.268e+00 146.406 6.85e-12 ***
pr           1.714e+00  5.140e+00   0.334     0.75
pm1.2        4.884e-13  9.615e+00   0.000     1.00
pm1.3       -2.430e+02  9.615e+00 -25.272 2.53e-07 ***
pm1.4       -5.350e+02  9.615e+00 -55.641 2.26e-09 ***
pm1.5       -1.500e+01  9.615e+00  -1.560     0.17
pm1.6       -2.285e+02  9.615e+00 -23.764 3.64e-07 ***
pm1.7       -2.585e+02  9.615e+00 -26.884 1.75e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.615 on 6 degrees of freedom
  (12 observations deleted due to missingness)
Multiple R-squared:  0.9988,        Adjusted R-squared:  0.9974
F-statistic: 712.3 on 7 and 6 DF,  p-value: 2.501e-08
```

```
> xBalance(pr ~ cap, data=nuke.nopt, strata=pm,
+          report=c("adj.means","adj.mean.diffs", "p.values","chisquare.test"))
```

```
     strata   strat
     stat      pr=0    pr=1 adj.diff
vars
cap         881.286 883.000  1.714
---Overall Test---
      chisquare df p.value
strat     0.127  1   0.721
---
Signif. codes:  0 '***' 0.001 '** ' 0.01 '*  ' 0.05 '.  ' 0.1 '   ' 1
```

3. Para comparar balance pre y post

```
> xBalance(pr ~ cap, data=nuke.nopt, strata = data.frame(original = factor("none"), pm),
+          report=c("adj.means","adj.mean.diffs","p.values", "chisquare.test"))
```

```
     strata original                                  pm
     stat       pr=0    pr=1 adj.diff            pr=0    pr=1 adj.diff
vars
cap          803.263 883.000  79.737          881.286 883.000  1.714
---Overall Test---
         chisquare df p.value
original     0.748  1   0.387
pm           0.127  1   0.721
---
Signif. codes:  0 '***' 0.001 '** ' 0.01 '*  ' 0.05 '.  ' 0.1 '   ' 1
```

4. Para comparar balance pre y post en base a diferencias estandarizadas

```
> xBalance(pr ~ cap, data=nuke.nopt, strata = data.frame(original = factor("none"), pm),
+          report=c("chisquare.test", "std.diffs"))
```

```
     strata original              pm
     stat   std.diff         std.diff
vars
cap          0.38030          0.00818
---Overall Test---
         chisquare df p.value
original     0.748  1   0.387
pm           0.127  1   0.721
---
Signif. codes:  0 '***' 0.001 '** ' 0.01 '*  ' 0.05 '.  ' 0.1 '   ' 1
```