Test Permutacions 17-18. Randomisation en un disseny en blocs aleatoritzats.

Jordi Ocaña, Sergi Civit

7 de març de 2018

1 Randomisation en un disseny en blocs aleatoritzats

1.1 Enunciat de la Situació Experimental

Los cigarrillos producen cantidades apreciables de monóxido de carbono. Cuando se inhala el humo del cigarrillo, el monóxido de carbono se combina con la hemoglobina para formar carboxihemoglobina. En un estudio sobre este tema **(Calvery, M.A. y otros, 1981. Carbon monoxide and exercise tolerance in chronic bronchitis and emphysema. Brit. Med. J. 283, 877-880)** los investigadores deseaban determinar si una concentración apreciable de carboxihemoglobina reduce la tolerancia al ejercicio en aquellos pacientes que sufren de bronquitis crónica y enfisema. Se seleccionaron 7 pacientes y, en un ambiente controlado, se les pidió que caminaran durante 12 minutos respirando cada una de las siguientes combinaciones gaseosas: aire, oxígeno, aire más monóxido de carbono y oxígeno más monóxido de carbono (respectivamente A, B, C, D). La cantidad de monóxido de carbono respirado fue suficiente para elevar la concentración de carboxihemoglobina de cada sujeto en 9%.

Para controlar el nivel de monóxido de carbono, **se pidió a los siete fumadores que dejaran de fumar 12 horas antes del experimento**. Los datos representan **las distancias caminadas por los sujetos (en metros) en los 12 minutos** para cada condición experimental. Para cada individuo, la secuencia de **combinaciones gaseosas se asignó al azar** (aunque en los datos siempre se indica primero A, luego B, etc. en realidad podría ser que el sujeto 1 hubiese realizado los ejercicios en el orden alatorio CADB, el sujeto 2 en orden DABC, etc.) y se **dejó un tiempo prudencial de recuperación entre ejercicio y ejercicio**.

Aquestes dades corresponen a un exercici d'un curs de Disseny d'experiments. La situació presentada correspon al que normalment s'anomena **disseny en blocs aleatoritzats**. En certa manera es pot considerar que és una generalització del "disseny per dades aparellades"que ja hem tractat al nostre curs; fixeu-vos que cada subjecte es sotmet a tots els tractaments (són només 7 subjectes, no 28 assignats 7 al tractament A, 7 al tractament B, etc.).

Disseny balancejat, d'un total de 7 individus **assignats a l'atzar a CADASCUNA de les combinacions gasoses (un total de 4)**.

La hipòtesi a demostrar (alternativa) més lògica és que hi ha almneys alguna diferencia entre els diferents nivells del tractament (combinacions gasoses que anomenarem atmosfera).

La idea intuïtiva és que com més gran sigui F més evidència tindrem contra H0. Per rebutjar-la sota un nivell de significació 0.05, DUREM A TERME una PROVA DE PERMUTACIONS que ens permetrà obtenir i calcular el p-valor d'acord amb LA DISTRIBUCIÓ DE L'ESTADÍSTIC OBTINGUT A PARTIR DE LES DADES RESMOSTREJADES. De partida, com a estadístic a calcular repetidament per cada permutació EMPRAREM el l'estadístic F que definirem en la propera secció. A l'script R ("proves esforç.R") al qual es llegeixen i es preparen les dades ("proves esforç.txt") trobareu com es calcularia aquest estadístic F i el corresponent p-valor.

1.2 DADES

Disseny balancejat, d'un total de 7 individus **assignats a l'atzar a CADASCUNA de les combinacions gasoses (un total de 4)**.

```
> # Lectura de les dades
> # Disseny balancejat, d'un total de 7 individus, ASSIGNATS ALEATORIAMENT
> # a cadascuna de les combinacions gasoses (atmofera)
> dat = read.table("proves esforç.txt", header = TRUE)

> # Les mateixes dades en una presentació més adequada per la funció 'aov':
> dades = data.frame(
+   subjecte = rep(factor(1:nrow(dat)), each = 4),
+   atmosfera = rep(factor(c("A","B","C","D")), nrow(dat)),
+   distancia = as.vector(t(dat[,-1]))
+ )
```

1.3 Que CAL PERMUTAR?

En general s'ha vist clar que calia permutar **els 4 tractaments o atmosferes respirades o combinacions gasoses, per separat dins cadascun dels 7 blocs o subjectes** -una cosa diferent- és si s'ha sabut portar a la pràctica, tal com es comentarà més endavant. El propi disseny experimental i l'estructura de les dades impliquen la necessitat d'això, fixem-nos que, per exemple, permutar els 28 valors observats lliurement (o els 28 codis de tractament) produiria conjunts de dades als quals hi hauria subjectes que no haurien rebut tots els tractaments i que presentarien repeticions de tractaments.

1.4 Quantes PERMUTACIONS?

En general també s'ha vist clar que, d'acord amb la **forma de permutació requerida, dins cada subjecte** hi ha $4! = 24$ permutacions possibles dels

4 tractaments (atmosfera) i, **atès que hi ha 7 blocs**, $(4!)^7$ configuracions en total, un nombre molt elevat

```
> factorial(4)^7
```

```
[1] 4586471424
```

1.5 Estadistic ESCOLLIT

Elecció de l'ESTADISTIC. Aqui COMENÇAREM PER UN POSSIBLE ESTADÍSTIC COM ÉS EL CLÀSSIC QUE ENS PROPORCIONA LA METODOLOGIA PARAMÈTRICA **Estadístic F)**

```
> # Preparatius inicials:
> # Les mateixes dades en una presentació més adequada per la funció 'aov':
> dades = data.frame(
+   subjecte = rep(factor(1:nrow(dat)), each = 4),
+   atmosfera = rep(factor(c("A","B","C","D")), nrow(dat)),
+   distancia = as.vector(t(dat[,-1]))
+ )
> taula.ANOVA = anova(aov(distancia ~ subjecte + atmosfera, data = dades))
> taula.ANOVA
```

```
Analysis of Variance Table

Response: distancia
          Df  Sum Sq Mean Sq F value    Pr(>F)
subjecte   6 1471772  245295 270.606 < 2.2e-16 ***
atmosfera  3   44827   14942  16.484 2.108e-05 ***
Residuals 18   16316     906
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> # Estadístic F per "atmosfera":
> taula.ANOVA["atmosfera","F value"]
```

```
[1] 16.48396
```

```
>  # o bé
> f.obs = taula.ANOVA[2,4]
> f.obs
```

```
[1] 16.48396
```

1.6 Test de PERMUTACIONS DE MONTE CARLO i Estadístic F

Veiem com es podria implementar una prova de permutacions de Monte Carlo similar. ENUMERAR TOTES LES POSSIBLES PERMUTACIONS de les dades ÉS computacionalment MOLT complex:

```
> factorial(4)^7

[1] 4586471424
```

Està clar que cal un ENFOC de TEST de PERMUTACIONS de Monte Carlo.

La primera i potser principal dificultat pot ser veure com es pot **generar una permutació aleatòria respectant els blocs**, és a dir, per separat dins cada subjecte. Una solució correcta encara que excessivament dependent d'aquest cas particular seria codificar per separat dins cadascun dels 7 blocs la generació d'una permutació aleatòria.

```
> # data.frame amb la mateixa estructura que les dades originals
> # però que emmagatzemarà unes dades permutades:
> dades.perm <- dades
> # Una permutació aleatòria de la columna 'atmosfera':
> dades.perm[1:4,2] <- sample(dades[1:4,2], replace = FALSE)
> dades.perm[5:8,2] <- sample(dades[5:8,2], replace = FALSE)
> dades.perm[9:12,2] <- sample(dades[9:12,2], replace = FALSE)
> dades.perm[13:16,2] <- sample(dades[13:16,2], replace = FALSE)
> dades.perm[17:20,2] <- sample(dades[17:20,2], replace = FALSE)
> dades.perm[21:24,2] <- sample(dades[21:24,2], replace = FALSE)
> dades.perm[25:28,2] <- sample(dades[25:28,2], replace = FALSE)
> dades.perm
```

	subjecte	atmosfera	distancia
1	1	D	835
2	1	A	874
3	1	B	750
4	1	C	854
5	2	C	787
6	2	D	827
7	2	A	755
8	2	B	829
9	3	C	724
10	3	A	738
11	3	D	698
12	3	B	726
13	4	B	336
14	4	C	378
15	4	D	210
16	4	A	279
17	5	B	252
18	5	D	315
19	5	C	168
20	5	A	336
21	6	C	560

```
22        6         D       672
23        6         B       558
24        6         A       642
25        7         D       336
26        7         B       341
27        7         A       260
28        7         C       336

>
```

Una solució bastant eficient i general es basaria en utilitzar la funció **tapply**. Els seus tres primers arguments són: **X, INDEX, FUN = NULL**. FUN correspon a una funció. Aquesta funció s'avaluarà sobre els valors d'X però per separat dins cada nivell o valor d'INDEX, que típicament serà un factor. Per tant, per fer sample dels valors de atmosfera per separat dins cada nivell de subjecte podem fer:

```
> dades.perm<-tapply(dades$atmosfera, dades$subjecte, sample)
> dades.perm

$`1`
[1] B D C A
Levels: A B C D

$`2`
[1] B D A C
Levels: A B C D

$`3`
[1] D A C B
Levels: A B C D

$`4`
[1] C D B A
Levels: A B C D

$`5`
[1] A D C B
Levels: A B C D

$`6`
[1] A D C B
Levels: A B C D

$`7`
[1] B A C D
Levels: A B C D
```

```
> # o equivalentment:
> # tapply( dades[,2], dades[,1], sample)
```

Per reconvertir la llista anterior en un vector, podem aplicar la funció **unlist**

```
> # Estatístic F sobre aquestes dades permutades:
> nperm = 9999
> set.seed(123)
> f.perms = replicate(nperm, {
+   atmos.perm = unlist(tapply(dades$atmosfera, dades$subjecte, sample, replace = FALSE))
+   anova(aov(dades$distancia ~ dades$subjecte + atmos.perm))[2,4]
+ }
+ )
```

1.7 Test de PERMUTACIONS DE MONTE CARLO i Estadístic F: P-Valor

```
> (sum(f.perms >= f.obs) + 1) / (nperm + 1)

[1] 1e-04
```

Exercici: Podrieu crear una funció bastant general (que per simplicitat només generi el p-valor final)que implementi el procediment anterior?

1.8 Test de PERMUTACIONS DE MONTE CARLO i Estadístic F SIMPLIFICAT. P-Valor

```
> # En canvi aquest enfoc donaria exactament el mateix resultat que l'estadístic F:
>
> f.simpli = function(x) {
+   s = tapply(x$distancia, x$atmosfera, sum)
+   sum(s * s)
+ }
> set.seed(123)
> dades.perm = dades
> f.obs = f.simpli(dades)
> f.perms = replicate(nperm, {
+   dades.perm$atmosfera = unlist(tapply(dades$atmosfera, dades$subjecte, sample))
+   f.simpli(dades.perm)
+ }
+ )
> (sum(f.perms >= f.obs) + 1) / (nperm + 1)

[1] 1e-04
```