Fiche TD avec le logiciel  : tdr331

Arrachage de clous et sous-espaces vectoriels

D.Chessel & A.-B. Dufour

La fiche donne quelques éléments de réflexions sur la manipulation des modèles linéaires à partir d'un exemple de P. Dagnelie. L'emploi des facteurs à modalités ordonnées dans une analyse de variance ne pose pas de problèmes particuliers aux utilisateurs.

Table des matières

1 Trois critères de classification

L'exemple est à la page 321 du tome 2 du livre 'Statistique Apliquée et Théorique' de P. Dagnélie[1]. Les données illustrant ces livres sont disponibles à :

`http://www.dagnelie.be/stdonn.html`

Récupérer le fichier `s2e11021.txt` (Tome 2, exemple 11.2.1) dans le dossier `st2donn.zip`. On y trouvera quatre variables.

- La variable mesurée est la résistance à l'arrachage des clous de panneaux de particules en kg. Elle est dans la quatrième colonne.
- La première variable contrôlée est le diamètre des clous qui prend deux valeurs : 1 pour 6.5 mm et 2 pour 8 mm.
- La seconde variable contrôlée est le diamètre des anneaux de support qui prend deux valeurs : 1 pour 22 mm et 2 pour 30 mm.
- La troisième variable contrôlée est la vitesse d'arrachage qui prend trois valeurs : 1 pour 22 mm/minute, 2 pour 45 mm/minute et 3 pour 90 mm/minute.

Notez que ce jeu de données particulier se trouve également sur le site pédagogique avec un 'S' majuscule : `S2e11021.txt`
Installer les données.

```
exos2 <- read.table("http://pbil.univ-lyon1.fr/R/donnees/S2e11021.txt")
names(exos2) <- c("clou", "anne", "vite", "resi")
exos2$clou <- as.factor(exos2$clou)
exos2$anne <- as.factor(exos2$anne)
exos2$vite <- as.factor(exos2$vite)
summary(exos2)
```

```
clou   anne   vite        resi
1:30   1:30   1:20   Min.   :47.0
2:30   2:30   2:20   1st Qu.:57.0
               3:20   Median :66.5
                      Mean   :65.7
                      3rd Qu.:75.0
                      Max.   :85.0
```

Faire l'analyse de variance à trois facteurs et toutes les interactions : on retrouve *exactement* le tableau 11.2.2 de l'ouvrage cité.

```
lm1 <- lm(resi ~ clou * anne * vite, data = exos2)
anova(lm1)
```

```
Analysis of Variance Table
Response: resi
              Df Sum Sq Mean Sq F value  Pr(>F)
clou           1   4403    4403  344.23 < 2e-16 ***
anne           1    355     355   27.77 3.2e-06 ***
vite           2    632     316   24.71 4.2e-08 ***
clou:anne      1     29      29    2.30   0.136
clou:vite      2     86      43    3.37   0.043 *
anne:vite      2     54      27    2.11   0.132
clou:anne:vite 2     10       5    0.40   0.671
Residuals     48    614      13
---
Signif. codes:  0
```

On retient le modèle :

```
lm2 <- lm(resi ~ clou + vite + anne + clou * vite, data = exos2)
anova(lm2)
```

[1] Dagnelie, P. 1998. Statistique théorique et appliquée. Tome 1 - Statistique descriptive et bases de l'inférence statistique. 508 pp. Tome 2 - Inférence statistique à une et à deux dimensions. 659 pp. De Boeck & Larcier, Département De Boeck Université, Bruxelles.

```
Analysis of Variance Table
Response: resi
          Df Sum Sq Mean Sq F value  Pr(>F)
clou       1   4403    4403  329.75 < 2e-16 ***
vite       2    632     316   23.67 4.5e-08 ***
anne       1    355     355   26.60 3.8e-06 ***
clou:vite  2     86      43    3.23   0.048 *
Residuals 53    708      13
---
Signif. codes:  0
```

Les coefficients sont :

```
coefficients(lm1)
     (Intercept)             clou2             anne2             vite2
            55.2              17.0              -2.8               2.8
           vite3       clou2:anne2       clou2:vite2       clou2:vite3
             8.2              -2.4               6.0              -1.4
      anne2:vite2       anne2:vite3 clou2:anne2:vite2 clou2:anne2:vite3
            -2.2               0.2              -2.6               1.4
coefficients(lm2)
(Intercept)       clou2       vite2       vite3       anne2 clou2:vite2 clou2:vite3
     56.233      15.800       1.700       8.300      -4.867       4.700      -0.700
```

Que signifie le nom de ces coefficients ? Comment les calculs sont-ils effectués ? **La première chose à retenir est que les calculs ne sont pas faits, dans un logiciel, par des sommes de carrés d'écarts, des moyennes ou des sommes de produits.** On écrit, traditionnellement, la donnée de base sous la forme x_{ijkl} où i est une modalité du facteur a, j une modalité du facteur b, k une modalité du facteur c et l le numéro de la répétition. Ceci donne les moyennes $m_{ijk.}$, moyenne des répétitions d'une même combinaison des trois facteurs, et les moyennes associées à toutes les combinaisons $m_{ij..}, m_{i.k.}, m_{.jk.}, m_{i...}, m_{.j..}, m_{..k.}, m_{....}$. On a alors le modèle observé :

$$\begin{aligned} x_{ijkl} - m_{....} &= (m_{i...} - m_{....}) + (m_{.j..} - m_{....}) + (m_{..k.} - m_{....}) \\ &+ (m_{ij..} - m_{i...} - m_{.j..} + m_{....}) + (m_{i.k.} - m_{i...} - m_{..k.} + m_{....}) \\ &+ (m_{.jk.} - m_{.j..} - m_{..k.} + m_{....}) \\ &+ (m_{ijk.} - m_{ij..} - m_{i.k.} + m_{.jk.} + m_{i...} + m_{.j..} + m_{..k.} - m_{....}) \\ &+ (x_{ijkl} - m_{ijk.}) \end{aligned} \tag{1}$$

Et l'équation :

$$\begin{aligned} \sum (x_{ijkl} - m_{....})^2 &= \sum (m_{i...} - m_{....})^2 + \sum (m_{.j..} - m_{....})^2 + \sum (m_{..k.} - m_{....})^2 \\ &+ \sum (m_{ij..} - m_{i...} - m_{.j..} + m_{....})^2 \\ &+ \sum (m_{i.k.} - m_{i...} - m_{..k.} + m_{....})^2 \\ &+ \sum (m_{.jk.} - m_{.j..} - m_{..k.} + m_{....})^2 \\ &+ \sum (m_{ijk.} - m_{ij..} - m_{i.k.} + m_{.jk.} + m_{i...} + m_{.j..} + m_{..k.} - m_{....})^2 \\ &+ \sum (x_{ijkl} - m_{ijk.})^2 \end{aligned} \tag{2}$$

ou :

$$\mathrm{SCE}_t = \mathrm{SCE}_a + \mathrm{SCE}_b + \mathrm{SCE}_c + \mathrm{SCE}_{ab} + \mathrm{SCE}_{ac} + \mathrm{SCE}_{bc} + \mathrm{SCE}_{abc} + \mathrm{SCE}_r$$

Ceci ne sert pas à simplifier la réalité, d'autant plus qu'il suffit d'enlever au hasard une donnée pour les rendre complètement inutiles, ce qui ne gène pas le programme :

```
lm3 <- lm(resi ~ clou + vite + anne + clou * vite, data = exos2[-27,
    ])
anova(lm3)
```

```
Analysis of Variance Table
Response: resi
          Df Sum Sq Mean Sq F value  Pr(>F)
clou       1   4274    4274  331.05 < 2e-16 ***
vite       2    676     338   26.17 1.4e-08 ***
anne       1    325     325   25.18 6.5e-06 ***
clou:vite  2     99      49    3.83   0.028 *
Residuals 52    671      13
---
Signif. codes:  0
```

Le calcul à la main est possible, lorsque le plan d'expérience est complet à nombre égal de répétitions (bien que pénible et fort long), il est impossible dans tous les autres cas. Il vaut mieux comprendre comment il est géré en général.

2 La fonction `model.matrix`

La fonction `gl` permet de générer des plans d'expériences :

```
gl(2, 6)
```

```
 [1] 1 1 1 1 1 1 2 2 2 2 2 2
Levels: 1 2
```

```
gl(2, 3, 12)
```

```
 [1] 1 1 1 2 2 2 1 1 1 2 2 2
Levels: 1 2
```

```
f1 <- gl(2, 6)
f2 <- gl(3, 2, 12)
table(f1, f2)
```

```
   f2
f1  1 2 3
  1 2 2 2
  2 2 2 2
```

```
cbind(f1, f2)
```

```
      f1 f2
 [1,]  1  1
 [2,]  1  1
 [3,]  1  2
 [4,]  1  2
 [5,]  1  3
 [6,]  1  3
 [7,]  2  1
 [8,]  2  1
 [9,]  2  2
[10,]  2  2
[11,]  2  3
[12,]  2  3
```

Nous avons 2 facteurs. Le premier a 2 modalités, le second en a 3, le plan est complet à 2 répétitions partout. La fonction `model.matrix` gère les indicatrices des classes des facteurs pour transformer un problème d'analyse de variance en problème de régression multiple. Elle le fait indépendamment de la présence de l'observation. Pour un facteur seul :

```
model.matrix(~f1)
```

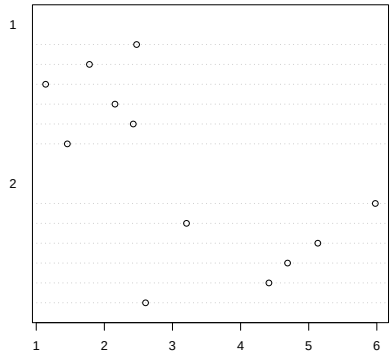
```
   (Intercept) f12
1            1   0
2            1   0
3            1   0
4            1   0
5            1   0
6            1   0
7            1   1
8            1   1
9            1   1
10           1   1
11           1   1
12           1   1
attr(,"assign")
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.treatment"
```

Le sous-espace associé au facteur `f1` est de dimension 2. La première colonne est le vecteur $\mathbf{1}_n$, le second est la deuxième indicatrice de classes. L'attribut `assign` conserve pour la suite que le premier vecteur est celui des constantes qui servira dans les modèles réponse = Cte + ... (ordonnée à l'origine ou intercept) et que le second vecteur est associé au facteur `f1` (le premier de la formule). On peut utiliser strictement les deux indicatrices de classe en enlevant la constante :

```
 model.matrix(~-1 + f1)
   f11 f12
1    1   0
2    1   0
3    1   0
4    1   0
5    1   0
6    1   0
7    0   1
8    0   1
9    0   1
10   0   1
11   0   1
12   0   1
attr(,"assign")
[1] 1 1
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.treatment"
```

L'attribut `assign` conserve pour la suite que les deux premiers vecteurs sont associés au facteur `f1` (le premier de la formule) et il n'y aura pas d'intercept.

```
 obs <- c(rnorm(6, 2), rnorm(6, 4))
 dotchart(obs, gr = f1)
 coefficients(lm(obs ~ f1))
(Intercept)         f12
      2.137       1.642
 coefficients(lm(obs ~ -1 + f1))
  f11   f12
2.137 3.779
```



Les vecteurs calculés dans `model.matrix` servent de prédicteurs dans une régression linéaire :

```
 a <- model.matrix(~-1 + f1)[, 1]
 b <- model.matrix(~-1 + f1)[, 2]
 coefficients(lm(obs ~ -1 + a + b))
    a     b
1.905 4.338
```

C'est seulement dans les cas les plus simples que les moyennes sont les prédictions du modèle :

```
 mean(obs[a > 0])
[1] 1.905
 mean(obs[b > 0])
[1] 4.338
```

Prenons deux facteurs :

```
 model.matrix(~-1 + f1 + f2)
   f11 f12 f22 f23
1    1   0   0   0
2    1   0   0   0
3    1   0   1   0
4    1   0   1   0
5    1   0   0   1
6    1   0   0   1
7    0   1   0   0
8    0   1   0   0
9    0   1   1   0
10   0   1   1   0
11   0   1   0   1
12   0   1   0   1
attr(,"assign")
[1] 1 1 2 2
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.treatment"
attr(,"contrasts")$f2
[1] "contr.treatment"
```

Il y a maintenant deux sous espaces. Le premier est de dimension 2 et une base est formée des deux indicatrices. Le second est de dimension 3 mais il contient le vecteur des constantes, lequel est dans le précédent. On perd systématiquement une dimension en enlevant la première indicatrice. On garde que les colonnes 3 et 4 sont associés au second facteur dans l'attribut `assign`.

```
 f3 <- gl(5, 2, 12)
 f3
 [1] 1 1 2 2 3 3 4 4 5 5 1 1
Levels: 1 2 3 4 5
 model.matrix(~f1 + f2 + f3)
   (Intercept) f12 f22 f23 f32 f33 f34 f35
1            1   0   0   0   0   0   0   0
2            1   0   0   0   0   0   0   0
3            1   0   1   0   1   0   0   0
4            1   0   1   0   1   0   0   0
5            1   0   0   1   0   1   0   0
6            1   0   0   1   0   1   0   0
7            1   1   0   0   0   0   1   0
8            1   1   0   0   0   0   1   0
9            1   1   1   0   0   0   0   1
10           1   1   1   0   0   0   0   1
11           1   1   0   1   0   0   0   0
12           1   1   0   1   0   0   0   0
attr(,"assign")
[1] 0 1 2 2 3 3 3
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.treatment"
attr(,"contrasts")$f2
[1] "contr.treatment"

attr(,"contrasts")$f3
[1] "contr.treatment"
```

Il y a maintenant 3 facteurs et sont systématiquement conservés toutes les indicatrices de classe sauf la première. Par défaut, une variable qualitative est l'ensemble de ces indicatrices diminué de la première.

```
 q1 <- as.numeric(f1)
 q2 <- as.numeric(f2)
 q3 <- as.numeric(f3)
 model.matrix(~q1 + q2 + q3 + f1 + f2 + f3)
   (Intercept) q1 q2 q3 f12 f22 f23 f32 f33 f34 f35
1            1  1  1  1   0   0   0   0   0   0   0
2            1  1  1  1   0   0   0   0   0   0   0
3            1  1  2  2   0   1   0   1   0   0   0
4            1  1  2  2   0   1   0   1   0   0   0
5            1  1  3  3   0   0   1   0   1   0   0
6            1  1  3  3   0   0   1   0   1   0   0
7            1  2  1  4   1   0   0   0   0   1   0
8            1  2  1  4   1   0   0   0   0   1   0
9            1  2  2  5   1   1   0   0   0   0   1
10           1  2  2  5   1   1   0   0   0   0   1
11           1  2  3  1   1   0   1   0   0   0   0
12           1  2  3  1   1   0   1   0   0   0   0
attr(,"assign")
 [1] 0 1 2 3 4 5 5 6 6 6 6
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.treatment"
attr(,"contrasts")$f2
[1] "contr.treatment"

attr(,"contrasts")$f3
[1] "contr.treatment"
```

3 Les contrastes

La question qui vient à l'esprit est : "pourquoi la première indicatrice est-elle enlevée ?" Pourrait-on en enlever une autre ? Cette question est ,en fait, plus générale.
Un modèle linéaire a deux fonctions : soit on cherche à l'utiliser (prédire), soit on cherche à le construire (analyser). C'est un va et vient. Si le facteur est contrôlé

dans une expérience, il est naturel qu'il y ait des témoins et de partir du principe que dans le groupe témoin, l'effet est nul. Le modèle est du type :

$$\hat{y}_i = Cte + \alpha_{Classe(i)} \text{ avec } \alpha_{temoin} = 0.$$

Dans $\mathbb{R}^n$, il s'écrit :

$$\hat{\mathbf{y}} = Cte\mathbf{1}_n + \sum_{j=1}^{m} \alpha_j \mathbf{I}_j.$$

Si la première classe est celle des témoins, le modèle est vue par la réponse du type $\alpha_{temoin} = 0$, $\alpha_2, \ldots, \alpha_m$ et vue par la géométrie du type combinaison des indicatrices des classes sans la première. Par défaut, c'est le point de vue utilisé.

```
 contrasts(f1)
  2
1 0
2 1
 contrasts(f2)
  2 3
1 0 0
2 1 0
3 0 1
 contrasts(f3)
  2 3 4 5
1 0 0 0 0
2 1 0 0 0
3 0 1 0 0
4 0 0 1 0
5 0 0 0 1
```

On peut s'intéresser aux coefficients ou aux combinaisons d'indicatrices utilisant ces coefficients. Si on ne dit rien, le choix est fait par les options :

```
 options("contrasts")
$contrasts
        unordered           ordered
"contr.treatment"      "contr.poly"
```

On voit une matrice dont les lignes sont les modalités du facteur et les colonnes sont les vecteurs engendrés dans le sous-espace des indicatrices. Le contraste désigne de manière équivalente :

- le vecteur engendré (`f22` ou `f23`)
- les coefficients de la combinaison linéaire des indicatrices (0/1/0 pour `f22`, 0/0/1 pour `f23`). Ainsi $\mathbf{f}_{22} = 0\mathbf{I}_1 + 1\mathbf{I}_2 + 0\mathbf{I}_3$. Il suffit de dire que `f22` est le contraste 0, 1, 0.

  Le nom **contr.treatment** vient du fait que dans un modèle basé sur ces contrastes le coefficient de la classe 1 est nul et que le test contre la non nullité du coefficient de `f22` dans le modèle est un test de l'effet de la classe 2, que le test contre la non nullité du coefficient de `f23` dans le modèle est un test de l'effet de la classe 3, ... Il convient parfaitement au cas où dans la classe 1, on a les témoins (on sait que l'effet est nul !) et dans la classe 2 le traitement 1, dans la classe 3 le traitement 2, ... On a les contrastes de l'effet des traitements quand on a mis les témoins dans la classe 1.
  On a le droit de changer :

```
 contrasts(f3) <- contr.treatment(5, base = 5)
 contrasts(f3)
```

```
  1 2 3 4
1 1 0 0 0
2 0 1 0 0
3 0 0 1 0
4 0 0 0 1
5 0 0 0 0
```

Ceci signifie que les contrastes du facteur `f3` doivent être ceux d'un facteur à 5 modalités dont les témoins sont dans la 5. Alors :

```
 model.matrix(~1 + f3)
   (Intercept) f31 f32 f33 f34
1            1   1   0   0   0
2            1   1   0   0   0
3            1   0   1   0   0
4            1   0   1   0   0
5            1   0   0   1   0
6            1   0   0   1   0
7            1   0   0   0   1
8            1   0   0   0   1
9            1   0   0   0   0
10           1   0   0   0   0
11           1   1   0   0   0
12           1   1   0   0   0
attr(,"assign")
[1] 0 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$f3
  1 2 3 4
1 1 0 0 0
2 0 1 0 0
3 0 0 1 0
4 0 0 0 1
5 0 0 0 0
```

Le test de non nullité du coefficient de `f31` sera celui de l'effet 1,..., le test de non nullité du coefficient de `f34` sera celui de l'effet 4, les témoins étant en 5. On peut modifier à sa guise ce choix par défaut et définir soit même les contrastes. Par exemple :

```
 contrasts(f1) <- matrix(c(-1, 1), 2, 1)
 f1
 [1] 1 1 1 1 1 1 2 2 2 2 2 2
attr(,"contrasts")
  [,1]
1   -1
2    1
Levels: 1 2
 attributes(f1)
$levels
[1] "1" "2"
$class
[1] "factor"

$contrasts
  [,1]
1   -1
2    1
 model.matrix(~f3 + f1)
   (Intercept) f31 f32 f33 f34 f11
1            1   1   0   0   0  -1
2            1   1   0   0   0  -1
3            1   0   1   0   0  -1
4            1   0   1   0   0  -1
5            1   0   0   1   0  -1
6            1   0   0   1   0  -1
7            1   0   0   0   1   1
8            1   0   0   0   1   1
9            1   0   0   0   0   1
10           1   0   0   0   0   1
11           1   1   0   0   0   1
12           1   1   0   0   0   1
attr(,"assign")
[1] 0 1 1 1 1 2
attr(,"contrasts")
attr(,"contrasts")$f3
  1 2 3 4
1 1 0 0 0
2 0 1 0 0
3 0 0 1 0
4 0 0 0 1
5 0 0 0 0
```

```
attr(,"contrasts")$f1
  [,1]
1   -1
2    1
```

Notez que le facteur ne change pas quand ses contrastes changent. On ne change pas le facteur mais l'usage qu'on en fait. Ici le test de non nullité du coefficient de `f11` sera exactement un test sur la différence des moyennes entre les deux groupes **sachant qu'on a enlevé l'effet de** `f3`. Supposons que `f3` qui a cinq modalités représente 5 sites, 3 au nord et 2 au sud. On testera nord contre sud avec :

```
contrasts(f3) <- matrix(c(1, 1, 1, -1, -1), 5, 1)
contrasts(f3)
  [,1]       [,2]     [,3]     [,4]
1    1 -4.307e-01 -0.49050 -0.49050
2    1 -3.854e-01  0.50899  0.50899
3    1  8.161e-01 -0.01849 -0.01849
4   -1  2.776e-17  0.50000 -0.50000
5   -1  2.776e-17 -0.50000  0.50000
```

La fonction a complété par une base orthonormée :

```
crossprod(contrasts(f3))
          [,1]       [,2]       [,3]       [,4]
[1,]  5.000e+00  1.665e-16 -2.220e-16 -2.220e-16
[2,]  1.665e-16  1.000e+00 -6.072e-17 -6.072e-17
[3,] -2.220e-16 -6.072e-17  1.000e+00 -5.551e-17
[4,] -2.220e-16 -6.072e-17 -5.551e-17  1.000e+00
```

Ce qui complète :

```
model.matrix(~f3)
   (Intercept) f31        f32      f33      f34
1            1   1 -4.307e-01 -0.49050 -0.49050
2            1   1 -4.307e-01 -0.49050 -0.49050
3            1   1 -3.854e-01  0.50899  0.50899
4            1   1 -3.854e-01  0.50899  0.50899
5            1   1  8.161e-01 -0.01849 -0.01849
6            1   1  8.161e-01 -0.01849 -0.01849
7            1  -1  2.776e-17  0.50000 -0.50000
8            1  -1  2.776e-17  0.50000 -0.50000
9            1  -1  2.776e-17 -0.50000  0.50000
10           1  -1  2.776e-17 -0.50000  0.50000
11           1   1 -4.307e-01 -0.49050 -0.49050
12           1   1 -4.307e-01 -0.49050 -0.49050
attr(,"assign")
[1] 0 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$f3
  [,1]       [,2]     [,3]     [,4]
1    1 -4.307e-01 -0.49050 -0.49050
2    1 -3.854e-01  0.50899  0.50899
3    1  8.161e-01 -0.01849 -0.01849
4   -1  2.776e-17  0.50000 -0.50000
5   -1  2.776e-17 -0.50000  0.50000
```

résumé, ce qui est très important est qu'une somme de carrés d'écarts SCE est toujours du type $\left\|\hat{Y}_{U+V} - \hat{Y}_U\right\|^2$ où U et V sont des sous-espaces. Les sous-espaces sont effectivement calculés (par une de leurs bases), numérotés et conservés dans une matrice. Ensuite, qu'il s'agisse de variables qualitatives ou quantitatives ou des deux, qu'il y en ait, une, deux , trois, ... le calcul est toujours celui d'une projection. Il n'y a jamais référence à une solution explicite seulement accessible dans les cas les plus simples. Il faut donc bien manipuler les vecteurs qui génèrent les sous-espaces et qui apparaissent dans les noms des coefficients. Si on veut des **bases orthogonales**, il suffira d'utiliser l'option `contr.helmert`

```
model.matrix(~f1 + f2, cont = list(f1 = "contr.helmert", f2 = "contr.helmert"))
   (Intercept) f11 f21 f22
1            1  -1  -1  -1
2            1  -1  -1  -1
3            1  -1   1  -1
4            1  -1   1  -1
5            1  -1   0   2
```

```
6            1  -1   0   2
7            1   1  -1  -1
8            1   1  -1  -1
9            1   1   1  -1
10           1   1   1  -1
11           1   1   0   2
12           1   1   0   2
attr(,"assign")
[1] 0 1 2 2
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.helmert"
attr(,"contrasts")$f2
[1] "contr.helmert"
 crossprod(model.matrix(~f1 + f2, cont = list(f1 = "contr.helmert",
     f2 = "contr.helmert")))
             (Intercept) f11 f21 f22
(Intercept)          12   0   0   0
f11                   0  12   0   0
f21                   0   0   8   0
f22                   0   0   0  24
```

4 Effets principaux et interaction

```
 f1 <- gl(2, 6)
 f2 <- gl(3, 2, 12)
 f1:f2
 [1] 1:1 1:1 1:2 1:2 1:3 1:3 2:1 2:1 2:2 2:2 2:3 2:3
Levels: 1:1 1:2 1:3 2:1 2:2 2:3
 model.matrix(~-1 + f1:f2)
   f11:f21 f12:f21 f11:f22 f12:f22 f11:f23 f12:f23
1        1       0       0       0       0       0
2        1       0       0       0       0       0
3        0       0       1       0       0       0
4        0       0       1       0       0       0
5        0       0       0       0       1       0
6        0       0       0       0       1       0
7        0       1       0       0       0       0
8        0       1       0       0       0       0
9        0       0       0       1       0       0
10       0       0       0       1       0       0
11       0       0       0       0       0       1
12       0       0       0       0       0       1
attr(,"assign")
[1] 1 1 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.treatment"
attr(,"contrasts")$f2
[1] "contr.treatment"
```

Le sous espace UxV est généré par les produits termes à termes des indicatrices. Il correspond ici au sous-espace engendré par les indicatrices des classes définies par le croisement des deux facteurs.

```
 model.matrix(~f1 * f2)
   (Intercept) f12 f22 f23 f12:f22 f12:f23
1            1   0   0   0       0       0
2            1   0   0   0       0       0
3            1   0   1   0       0       0
4            1   0   1   0       0       0
5            1   0   0   1       0       0
6            1   0   0   1       0       0
7            1   1   0   0       0       0
8            1   1   0   0       0       0
9            1   1   1   0       1       0
10           1   1   1   0       1       0
11           1   1   0   1       0       1
12           1   1   0   1       0       1
attr(,"assign")
[1] 0 1 2 2 3 3
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.treatment"
attr(,"contrasts")$f2
[1] "contr.treatment"
```

C'est le même sous-espace décomposé en 4 éléments :

- le vecteur des constantes pour l'intercept,
- l'effet 1,
- l'effet 2 (ou plus exactement l'effet 2 sachant 1),
- l'effet 1*2 (ou plus exactement l'effet 1 :2 sachant l'effet 1+2).

```
 model.matrix(~f1 + f1:f2)
   (Intercept) f12 f11:f22 f12:f22 f11:f23 f12:f23
1            1   0       0       0       0       0
2            1   0       0       0       0       0
3            1   0       1       0       0       0
4            1   0       1       0       0       0
5            1   0       0       0       1       0
6            1   0       0       0       1       0
7            1   1       0       0       0       0
8            1   1       0       0       0       0
9            1   1       0       1       0       0
10           1   1       0       1       0       0
11           1   1       0       0       0       1
12           1   1       0       0       0       1
attr(,"assign")
[1] 0 1 2 2 2 2
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.treatment"
attr(,"contrasts")$f2
[1] "contr.treatment"
```

C'est le même sous-espace décomposé en trois éléments.

```
 model.matrix(~-1 + f1 * f2)
   f11 f12 f22 f23 f12:f22 f12:f23
1    1   0   0   0       0       0
2    1   0   0   0       0       0
3    1   0   1   0       0       0
4    1   0   1   0       0       0
5    1   0   0   1       0       0
6    1   0   0   1       0       0
7    0   1   0   0       0       0
8    0   1   0   0       0       0
9    0   1   1   0       1       0
10   0   1   1   0       1       0
11   0   1   0   1       0       1
12   0   1   0   1       0       1
attr(,"assign")
[1] 1 1 2 2 3 3
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.treatment"
attr(,"contrasts")$f2
[1] "contr.treatment"
```

C'est le même sous-espace décomposé en trois éléments.

```
 model.matrix(~-1 + f1 * f2, , cont = list(f1 = "contr.helmert",
     f2 = "contr.helmert"))
   f11 f12 f21 f22 f11:f21 f11:f22
1    1   0  -1  -1       1       1
2    1   0  -1  -1       1       1
3    1   0   1  -1      -1       1
4    1   0   1  -1      -1       1
5    1   0   0   2       0      -2
6    1   0   0   2       0      -2
7    0   1  -1  -1      -1      -1
8    0   1  -1  -1      -1      -1
9    0   1   1  -1       1      -1
10   0   1   1  -1       1      -1
11   0   1   0   2       0       2
12   0   1   0   2       0       2
attr(,"assign")
[1] 1 1 2 2 3 3
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.helmert"
attr(,"contrasts")$f2
[1] "contr.helmert"
```

C'est le même sous-espace décomposé en trois éléments. Le test global est toujours le même, les tests emboîtés et les tests sur les coefficients dépendent étroitement du mode de présentation. Une grande liberté est accordée à l'utilisateur qui a une option par défaut.

5 Facteurs à modalités ordonnées

En revenant à l'exercice d'origine, on remarque que les modalités des variables sont ordonnées.

```
lm1 <- lm(resi ~ clou * vite * anne, data = exos2)
```

Passer les facteurs en modalités ordonnées :

```
w <- exos2
w$clou <- ordered(exos2$clou)
w$anne <- ordered(exos2$anne)
w$vite <- ordered(exos2$vite)
lm2 <- lm(resi ~ clou * vite * anne, data = w)
```

Passer les facteurs en variables quantitatives :

```
ww <- w
ww$clou <- as.numeric(w$clou)
ww$anne <- as.numeric(w$anne)
ww$vite <- as.numeric(w$vite)
lm3 <- lm(resi ~ clou * vite * anne, data = ww)
```

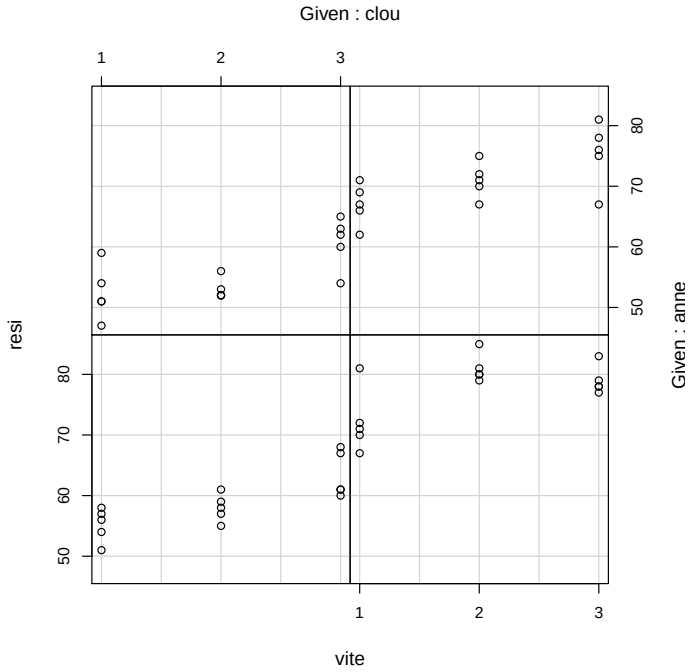
Caractériser ce qui distingue les trois tableaux.

```
apply(w[, 1:3], 2, function(x) contrasts(as.ordered(x)))
```

```
$clou
       .L
1 -0.7071
2  0.7071
$anne
       .L
1 -0.7071
2  0.7071

$vite
          .L      .Q
1 -7.071e-01  0.4082
2 -7.850e-17 -0.8165
3  7.071e-01  0.4082
```

Caractériser ce qui distingue les trois analyses de variances. Caractériser ce qui distingue les trois modèles. Représenter la résistance en fonction de la vitesse suivant les classes de taille des clous et des anneaux :



Peut-être aurions-nous du y penser plus tôt ? Identifier l'expression visible de l'inter-action. Conclure.