Chronique R

LAtent VAriables ANalysis - Le package lavaan pour les modèles d'équations structurelles

Les modèles d'équations structurelles sont des modèles statistiques initialement créés afin de combiner l'information provenant des données à des hypothèses qualitatives concernant les relations causales entre les variables afin d'estimer des relations causales. En fait, l'adjectif *structurel* signifie ici que les équations mathématiques décrivent une causalité, que le terme à gauche du symbole "=" est supposé être causé par les termes à droite du "=". De nos jours, les modèles d'équations structurelles sont couramment utilisés dans plusieurs domaines, sans toujours leur donner une interprétation causale. Ces modèles sont, entre autres, couramment utilisés en sciences sociales, en psychologie, en théorie de la mesure et en économie.

Dans plusieurs applications, les modèles d'équations structurelles sont utiles parce qu'ils permettent d'utiliser des variables latentes. Les variables latentes sont des variables qu'on ne peut pas mesurer directement avec exactitude, mais dont on peut mesurer les effets. On peut ainsi attribuer une valeur à la variable latente en mesurant ses effets. Par exemple, on ne peut pas directement mesurer jusqu'à quel point une personne est dépressive, mais la dépression a des symptômes qu'on peut mesurer. On peut donc évaluer le niveau de dépression d'une personne en mesurant ses symptômes dépressifs.

Dans cet article, nous ferons une brève introduction au « package » *lavaan* en R qui permet d'estimer des modèles d'équations structurelles. Le « package » *lavaan* est disponible sur CRAN. On peut installer le package de différentes façons, entre autres en exécutant la ligne `install.packages("lavaan");`. Nous allons considérer un exemple avec le jeu de données `PoliticalDemocracy` fourni avec *lavaan*. On peut charger *lavaan* et regarder la description du jeu de données :

```
require(lavaan);
?PoliticalDemocracy;
```

Construire le modèle

La première étape sera de construire le modèle théorique. Cette étape correspond à décrire les hypothèses qualitatives concernant les relations causales entre les variables du modèle d'équations structurelles. On devra créer un objet qui donnera les équations du modèle, voici un exemple avec notre jeu de données :

```
model = '
  # Définition des variables latentes
     ind60 =~ x1 + x2 + x3
     dem60 =~ y1 + a*y2 + b*y3 + c*y4
     dem65 =~ y5 + a*y6 + b*y7 + c*y8

  # régressions
    dem60 ~ ind60
    dem65 ~ ind60 + dem60

  # corrélations résiduelles
    y1 ~~ y5
    y2 ~~ y4 + y6
    y3 ~~ y7
    y4 ~~ y8
    y6 ~~ y8
';
```

Dans cet exemple, on remarque qu'on crée un objet appelé `model` qui contient du texte entre deux apostrophes ('). Ce texte constitue les équations du modèle. La syntaxe des équations est relativement simple à comprendre.

- On peut utiliser le # pour inscrire des commentaires, par exemple "`# Définition des variables latentes`" est un commentaire qui ne sera pas considéré comme une équation.

- Le symbole =~ est utilisé pour définir une variable latente. Ainsi "`ind60 =~ x1 + x2 + x3`" définit la variable ind60, qui ne fait pas partie du jeu de données, comme étant une fonction des variables x1, x2 et x3, qui, elles. sont des variables observées qui font partie du jeu de données.

- Le symbole ~ est utilisé pour représenter une régression. Ainsi, `dem65 ~ ind60 + dem60` indique que le modèle comporte une régression linéaire où dem65 est la variables dépendante et ind60 et dem60 sont les variables explicatives.

- Le symbole ~~ est utilisé pour représenter des termes de covariances (soit entre des variables du modèle, soit entre des erreurs résiduelles). Par exemple `y2 ~~ y4 + y6` signifie que le modèle doit permettre à y2 d'être corrélée avec y4 et avec y6. En fait, puisque y2 et y6 sont des variables dépendantes (elles sont respectivement les effets des variables latentes dem60 et dem65), il s'agit en fait de permettre des corrélations entre leurs erreurs résiduelles.

- On peut facilement imposer des contraintes aux paramètres du modèle. Par exemple `dem60 =~ y1 + a*y2 + b*y3 + c*y4` et `dem65 =~ y5 + a*y6 + b*y7 + c*y8` forcent les paramètres associés à y2, y3 et y4 pour créer la variable latente dem60 à être identiques aux paramètres associés à y6, y7 et y8 respectivement pour créer la variable latente dem65.

Ajuster le modèle

On peut ensuite ajuster le modèle avec la fonction `sem`. À cette étape, on ajoute à nos hypothèses qualitatives les informations provenant des données. Dans sa forme la plus simple, la fonction `sem` ne requiert que le modèle et le jeu de données :

```
fit = sem(model, data=PoliticalDemocracy);
```

Voici quelques autres arguments utiles qui peuvent être utilisés dans la fonction `sem` :

- `mimic` indique à la fonction d'imiter l'algorithme d'estimation utilisé par un autre logiciel. On peut imiter Mplus et EQS (ou lavaan).
- `missing` indique à la fonction comment traiter les données manquantes. On a le choix entre l'élimination *listwise* ("listwise") ou la méthode FIML ("fiml", "ml", "direct").
- `estimator` permet de choisir le type d'estimateur à utiliser. Il en existe une panoplie, dont l'estimateur du maximum de vraisemblance ("ML"), des estimateurs du maximum de vraisemblance robustes à la non-normalité ("MLM", "MLR", "MLMV", "MLMVS", "MLF") et plusieurs types d'estimateurs des moindres carrés.

- `test` permet de choisir le type de statistique à utiliser pour tester le modèle, soit le Khi-deux ordinaire ("standard"), des versions robustes à la non-normalité ("Satora.Bentler", "Yuan.Bentler") ou le bootstrap de Bollen-Stine ("boot", "bootstrap", "Bollen.Stine").
- `bootstrap` permet de déterminer le nombre de réplications bootstrap à obtenir si on choisit d'effectuer du bootstrap.

On peut finalement consulter les résultats du modèle avec la fonction `summary` :

```
summary(fit);
```

Les différentes options de la fonction `summary` appliquée à un objet de classe *lavaan* sont :

- `standardized` indique si la fonction doit effectuer une standardisation des paramètres estimés (TRUE ou FALSE).

- `fit.measures` indique si la fonction doit donner les indices d'ajustement du modèle (TRUE ou FALSE).

- `rsquare` indique si la fonction doit donner les coefficients de détermination (TRUE ou FALSE).

- `modindices` indique si la fonction doit donner les *modification indices* (TRUE ou FALSE). Ces indices peuvent être utiles pour déterminer comment un modèle qui s'ajuste mal aux données pourrait être amélioré.

Conclusion

Les modèles d'équations structurelles peuvent être extrêmement utiles dans plusieurs circonstances. Le « package » *lavaan* offre plusieurs possibilités dans l'ajustement de ces modèles. Il s'agit d'une excellente option gratuite qui se compare bien aux meilleurs logiciels payants (dont Mplus) pour plusieurs applications pratiques. Par contre, certaines

fonctionnalités avancées ne sont pas disponibles à ce jour avec *lavaan*, par exemple pour les modèls multiniveaux ou des fonctionnalités graphiques

Denis Talbot, Rédacteur en chef

Références

http://lavaan.ugent.be/

http://www.jstatsoft.org/v48/i02/