Die Methode der latenten Klassenanalyse

Pakete in R

Marcel Gumulak - SoSe 2024

Universität Bielefeld

Inhaltsverzeichnis

Einleitung: R-Paket Problematik

Die Welt wenn man LCA schnell und leicht in R machen könnte:

Einleitung: R-Paket Problematik (2)

poLCA is still undergoing active development. → **Große Erweiterungen angekündigt, aber kein Update bis heute**

The best way to do latent class analysis is by using Mplus, or if you are interested in some very specific LCA models you may need Latent Gold. Another decent option is to use PROC LCA in SAS. All the other ways and programs might be frustrating, but are helpful if your purposes happen to coincide with the specific R package.

First, there are all kinds of mixture models whose main purpose is to look for the classes in which the regression parameters differ. The meaning of the latent classes here is different as they are based not on the responses of respondents, but on the effects of one variables on other. These packages include flexmix, fpc, mmlcr, lcmm, and others. I do not discuss them.

→ **Modelle mit gleichem Namen, aber unterschiedlichem Modellansatz**

Der Bestseller:

Unfortunately, these approaches are currently only available in commercial software (Latent Gold, Mplus).

Einleitung: R-Paket Problematik (3)

	Classification by	
Design	Means (continuous data) / item probabilities (categorical data)	Regression parameters
Cross-sectional	Latent profile analysis Latent class analysis	Regression mixture models
Longitudinal	Repeated measures latent class analysis Latent transition analysis	Growth mixture models

Oft selbe Bezeichnung, aber unterschiedliche Modelle!

Übersicht – Paket: poLCA

Veröffentlicht:

Drew A. Linzer, Jeffrey B. Lewis am 14. Juni, 2011

Ziel:

Schätzung von latent-class Modellen & latent-class-Regressions-Modellen mit Kovariaten für dichotome und polytome Klassifikations-variablen (Indikatoren)

Verfügbar:

```
install.packages("poLCA")
library(poLCA)
```

poLCA: Vor- und Nachteile

Vorteile:

- Verknüpft simple Bedienung mit brauchbaren Methoden
  → Gut zum Einsteigen und für schnelle/simple Analysen
  → Bietet quasi alle grundlegend benötigten Funktionen
- Bietet neben einfachen latenten Klassenanalysemodellen die Möglichkeit Kovariaten durch Regressionen einzubauen
- Überschaubar strukturierte Ergebnisausgabe & Build-In Grafik
- Beinhaltet mehrere Modellprüfgrößen (AIC, BIC, χ^2, G^2, allg. Entropie)
- Nach Modellschätzung erweiterbar durch zusätzliche Funktionen

poLCA: Vor- und Nachteile (2)

Nachteile:

- Fehlen von konventionellen Funktionen (z.B. Ausgabe von class-means)
  → Nahezu keine (heutzutage zum Standard zählenden) Tests
- Ausschließlich veraltete Schätzmethode: One-Step Verfahren
  → Bei Verwendung von Kovariaten hängt die Klassifikation nicht nur von den Indikatoren, sondern auch den Kovariaten ab (im Vgl. zu Three-Step)
- Stark eingeschränkte Nutzungsmöglichkeiten
  → Schätzfunktion bietet gar keine Restriktionen und nur geringe Modifikation an
  → Kovariateneinfluss kann nur auf Indikatoren modelliert werden
  → Ausschließlich dichotome und polytome Indikatoren verwendbar

poLCA: Anwendungsbeispiel

Enthalten im Paket: `data(cheating, package = "poLCA")`

Quelle: Dayton CM (1998). Latent Class Scaling Analysis. Sage Publications, Thousand Oaks, CA.

Eigenschaften:

- dichotome Antworten von 319 Studenten auf Fragen zum Betrugsverhalten und deren Notendurchschnitt
- 4 Manifeste Variablen: LIEEXAM, LIEPAPER, FRAUD, COPYEXAM
- 1 Kovariate: GPA
- **Besonderheit:** Datensatz erfüllt Variablenvoraussetzungen bereits

poLCA: Variablenvoraussetzung

Damit Funktionen des Pakets fehlerfrei genutzt werden können, müssen bestimmte Voraussetzungen bzw. Eigenschaften bezüglich der genutzten Variablen im übergebenen Datensatz erfüllt sein

1. Indikatoren müssen integer-Werte (aber nicht zwingend vom Typ integer im Datensatz) besitzen:
   - Ganzzahlig und ab dem Wert 1 beginnend
   - Nicht negativ und aufsteigend
2. Kovariaten haben keine (weiteren) Vorgaben (Handhabung wie bei üblichen Regressionen)

	LIEEXAM	LIEPAPER	FRAUD	COPYEXAM	GPA
1	1	1	1	1	NA
2	1	1	1	1	NA
3	1	1	1	1	NA
4	1	1	1	1	NA
5	1	1	1	1	1
6	1	1	1	1	1
7	1	1	1	1	1
8	1	1	1	1	1
9	1	1	1	1	1
10	1	1	1	1	1

Showing 1 to 11 of 319 entries, 5 total columns

```
> str(cheating) # Kombiniert head() & typeof()
'data.frame':   319 obs. of  5 variables:
 $ LIEEXAM : num  1 1 1 1 1 1 1 1 1 1 ...
 $ LIEPAPER: num  1 1 1 1 1 1 1 1 1 1 ...
 $ FRAUD   : num  1 1 1 1 1 1 1 1 1 1 ...
 $ COPYEXAM: num  1 1 1 1 1 1 1 1 1 1 ...
 $ GPA     : int  NA NA NA NA 1 1 1 1 1 1 ...
```

```
> summary(cheating) # Detailierte Zusammenfassung
    LIEEXAM         LIEPAPER         FRAUD          COPYEXAM         GPA       
 Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000  
 1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000  
 Median :1.000   Median :1.000   Median :1.000   Median :1.000   Median :2.000  
 Mean   :1.107   Mean   :1.119   Mean   :1.066   Mean   :1.213   Mean   :2.327  
 3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:3.000  
 Max.   :2.000   Max.   :2.000   Max.   :2.000   Max.   :2.000   Max.   :5.000  
                                                                 NA's   :4      
```

poLCA: Modellschätzung

Ein Vorteil des Pakets ist seine leichte Bedienung in 2 Schritten:

1. **Formulierung einer Formel unter Verwendung von cbind()**
   - Latentes Klassenmodell: `formula = cbind(Manifeste Variablen) ~ 1`
   - Mit Kovariaten: `formula = cbind(Manifeste Variablen) ~ Kovariaten`

→ cbind(): Kombinieren von R-Objekten nach Spalten

→ Beispiel: `formula = cbind(LIEEXAM, LIEPAPER, FRAUD, COPYEXAM) ~ 1`

→ Ähnlichkeiten zur Formel einer linearen Regression

poLCA: Modellschätzung (2)

2. Schätzung des latenten Klassenmodells mithilfe von poLCA()

Allgemein:

```
poLCA(formula, data, nclass = 2, maxiter = 1000, graphs = FALSE,
      tol = 1e-10, na.rm = TRUE, probs.start = NULL, nrep = 1,
      verbose = TRUE, calc.se = TRUE)
```

→ Muss mehrmals ausgeführt werden für unterschiedliche Modelle

Beispiel:

```
poLCA(formula = formula, data = cheating, nclass = 2,
      maxiter = 3000, graphs = T, nrep = 10, na.rm = F)
```

poLCA: Modellschätzung (3)

Übersicht der (relevanten) verfügbaren Argumente: `?poLCA::poLCA()`

- **nclass** Anzahl latenter Klassen im Modell (Standard: 2)
  → nclass = 1 für log-linear Independence Model, in dem lediglich die wahren Klassenanteilswerte berechnet werden
- **maxiter** Maximale Anzahl an erlaubten Schätziterationen (Standard: 1000)
  → Je mehr Parameter bzw. Klassen, desto mehr Iterationen notwendig, um ein lokales oder globales Maximum zu finden
- **graphs** Ob eine 3D-Grafik der geschätzten Parameter erstellt werden soll (Std.: FALSE)

poLCA: Modellschätzung (4)

Übersicht der (relevanten) verfügbaren Argumente: `?poLCA::poLCA()`

- **na.rm** Mitberücksichtigung von fehlenden **Indikatorwerten** durch Full Information Maximum Likelihood (= FALSE); Ansonsten listwise deletion (Standard: TRUE)
  → **NAs in Kovariaten werden immer entfernt** (Keine Einstellung vorhanden)
- **nrep** Anzahl von Modellschätzungen unter unterschiedlichen zufälligen Startwerten für automatisierte Suche des globalen Maximums der Log-Likelihood
  → Je mehr Parameter bzw. Klassen, desto mehr Durchläufe erforderlich, um das beste Modell mit maximaler Log-Likelihood zu finden
- **verbose** Ob die Schätzungsergebnisse nach Durchlauf gezeigt werden sollen (Std.: TRUE)

poLCA: Modellergebnisse

Beispiel: `formula = cbind(LIEEXAM, LIEPAPER, FRAUD, COPYEXAM) ~ 1`

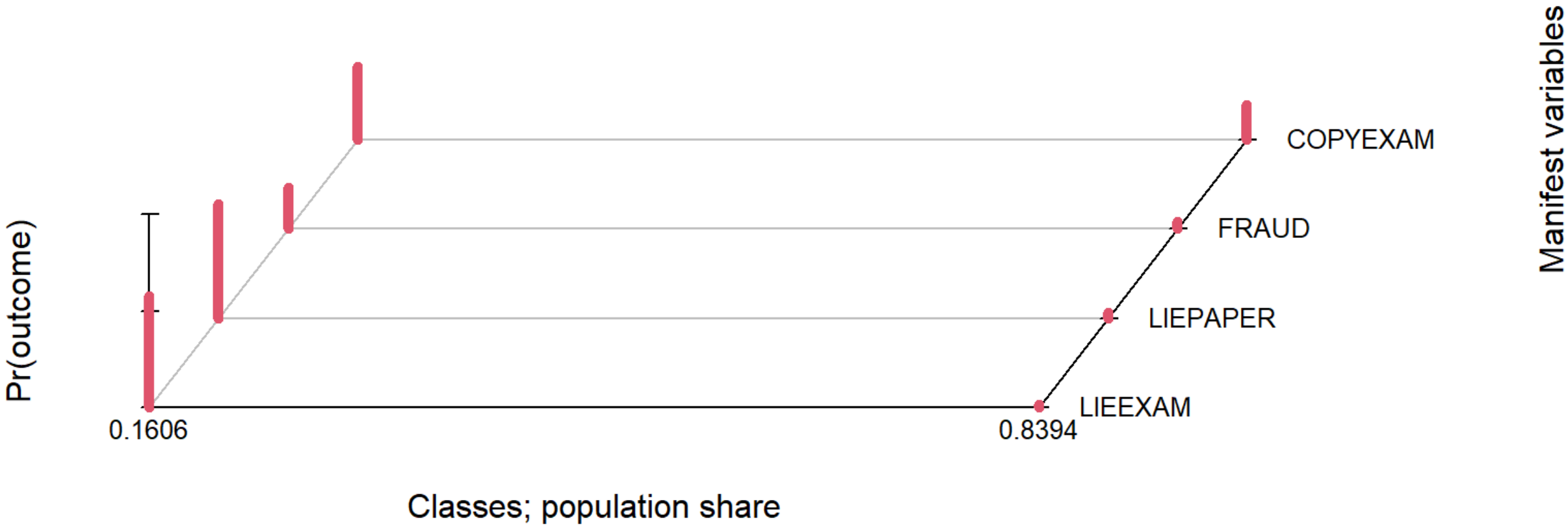
```
poLCA(formula = formula, data = cheating, nclass = 2,
      matiter = 3000, graphs = T, nrep = 10, na.rm = F)
```

```
> mod2 <- poLCA(formula = f, data = cheating, nclass = 2, maxiter = 3000,
+               graphs = T, nrep = 10, na.rm = F)
Model 1: llik = -440.0271 ... best llik = -440.0271
Model 2: llik = -440.0271 ... best llik = -440.0271
Model 3: llik = -440.0271 ... best llik = -440.0271
Model 4: llik = -440.0271 ... best llik = -440.0271
Model 5: llik = -440.0271 ... best llik = -440.0271
Model 6: llik = -440.0271 ... best llik = -440.0271
Model 7: llik = -440.0271 ... best llik = -440.0271
Model 8: llik = -440.0271 ... best llik = -440.0271
Model 9: llik = -440.0271 ... best llik = -440.0271
Model 10: llik = -440.0271 ... best llik = -440.0271
```

Das Modell ist hier mit 2 Klassen und keinen Kovariaten nicht sehr komplex
→ Die Likelihood-Funktion ist konkav und es existiert nur ein Maximum

```
Conditional item response (column) probabilities,
 by outcome variable, for each class (row)

$LIEEXAM
           Pr(1)  Pr(2)
class 1:  0.4231 0.5769
class 2:  0.9834 0.0166

$LIEPAPER
           Pr(1)  Pr(2)
class 1:  0.4109 0.5891
class 2:  0.9708 0.0292

$FRAUD
           Pr(1)  Pr(2)
class 1:  0.7840 0.2160
class 2:  0.9629 0.0371

$COPYEXAM
           Pr(1)  Pr(2)
class 1:  0.6236 0.3764
class 2:  0.8181 0.1819

Estimated class population shares
 0.1606 0.8394

Predicted class memberships (by modal posterior prob.)
 0.1693 0.8307
```

```
Estimated class population shares
 0.1606 0.8394

Predicted class memberships (by modal posterior prob.)
 0.1693 0.8307
```



poLCA: Modellprüfgrößen

AIC: $\mathrm{AIC} = -2\Lambda + 2\Phi$ **BIC**: $\mathrm{BIC} = -2\Lambda + \Phi \ln N$

χ^2-Teststatistik: $\chi^2 = \sum_{c=1}^{C} (q_c - \tilde{Q}_c)^2 / \tilde{Q}_c$

G^2-Teststatistik: $G^2 = 2 \sum_{c=1}^{C} q_c \log(q_c / \tilde{Q}_c)$

erwartete Anzahl an Observationen in Zelle c: $\tilde{Q}_c = N\tilde{P}(y_c)$

Beobachtete Anzahl an Observationen in Zelle c: q_c

poLCA: Modellprüfgrößen (2)

Modellprüfgrößen für das 2-Klassenmodell:

```
==========================================================
Fit for 2 latent classes:
==========================================================
number of observations: 319
number of estimated parameters: 9
residual degrees of freedom: 6
maximum log-likelihood: -440.0271

AIC(2): 898.0542
BIC(2): 931.9409
G^2(2): 7.764242 (Likelihood ratio/deviance statistic)
X^2(2): 8.323401 (Chi-square goodness of fit)
```

Differenz aus Anzahl an Zellen und geschätzten Parameter:

$$DoF_{Resid} = N_{Zellen} - K_{Parameter} - 1$$

poLCA: Modellprüfgrößen (3)

Modellprüfgrößen für das 2-Klassenmodell:

```
============================================================
Fit for 2 latent classes:
============================================================
number of observations: 319
number of estimated parameters: 9
residual degrees of freedom: 6
maximum log-likelihood: -440.0271

AIC(2): 898.0542
BIC(2): 931.9409
G^2(2): 7.764242 (Likelihood ratio/deviance statistic)
X^2(2): 8.323401 (Chi-square goodness of fit)
```

```
# kritischer Chi^2-Wert
qchisq(0.95, df = 6)
12.59159
```

⇒ Falls darüber = Signifikant = Schlechter Modellfit

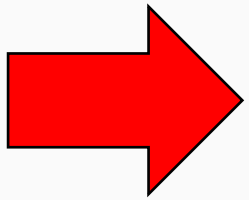
Aber: Annahmen erfüllt?

poLCA: Modellprüfgrößen (4)

Sind die Annahmen für den Chi-Quadrat- und LR-Test erfüllt?

Faustregel: Nicht mehr als 10-20% der Zellen beinhalten weniger als 5 Beobachtungen

```
> mod2$predcell
   LIEEXAM LIEPAPER FRAUD COPYEXAM observed expected
1        1        1     1        1      207  205.717
2        1        1     1        2       46   47.414
3        1        1     2        1        7    8.958
4        1        1     2        2        5    2.450
5        1        2     1        1       13   12.304
6        1        2     1        2        4    5.115
7        1        2     2        1        1    1.954
8        1        2     2        2        2    1.090
9        2        1     1        1       10    9.339
10       2        1     1        2        3    4.339
11       2        1     2        1        1    1.767
12       2        1     2        2        2    1.017
13       2        2     1        1       11    8.613
14       2        2     1        2        4    5.159
15       2        2     2        1        1    2.349
16       2        2     2        2        2    1.416
```



9 von 16 Zellen (56.25%) beinhalten weniger als 5 Beobachtungen

→ Test hier nicht aussagekräftig und daher nicht verwendbar

poLCA: Verfügbare Outputs

Verfügbare Outputs zu Klassenhäufigkeiten auf Basis von poLCA():

```
> mod2 <- poLCA(formula = f, data = cheating, nclass = 2, maxiter = 3000,
+               graphs = T, nrep = 10, na.rm = F)
```

- **mod2$P** (tatsächliche) latente Modellklassenanteile
- **mod2$predcell** (tatsächliche) latente Modellklassenhäufigkeiten vs. Erwartete Häufigkeiten
  → Wichtig: Nur Häufigkeiten größer 1 werden gelistet
- **mod2$predclass** Deterministische Klassenzugehörigkeiten
- **poLCA.predcell(lc = mod2, y = c(1, 2, 1, 2))** erwartete Anteil an Observationen für die Ausprägung **c(1, 2, 1, 2)**

poLCA: Verfügbare Outputs (2)

Verfügbare Outputs zu Wahrscheinlichkeiten auf Basis von poLCA():

```
> mod2 <- poLCA(formula = f, data = cheating, nclass = 2, maxiter = 3000,
+               graphs = T, nrep = 10, na.rm = F)
```

- **mod2$probs** — Liste von geschätzten klassenbedingten Ausprägungs-wahrscheinlichkeiten
- **mod2$probs.se** — Standardfehler von geschätzten klassenbedingten Ausprägungswahrscheinlichkeiten
- **mod2$posterior** — Matrix der posteriori Klassenwahrscheinlichkeiten
- **poLCA.posterior(lc = mod2, y = c(1, 2, 1, 2))** — Posteriori Klassenwahrscheinlichkeiten für die Ausprägung **c(1, 2, 1, 2)**

poLCA: Erweiterungen

Problem: Fehlen von konventionellen Funktionen

Lösung: Eigene Erweiterungen auf Basis des Pakets entwickeln

Beispiele (im R-File):

- **pruefgroeßen(mod)**

```
> rbind(pruefgroeßen(mod1), pruefgroeßen(mod2), pruefgroeßen(mod3)) # Wir wählen
          AIC     BIC    AIC3    Gsq   Chisq DoF crit.val95 p.Gsq.val95 p.Chisq.val95
mod1 942.876 957.937 946.876 62.586 136.342  11     19.675       0.000         0.000
mod2 898.054 931.941 907.054  7.764   8.323   6     12.592       0.256         0.215
mod3 900.471 953.184 914.471  0.181   0.182   1      3.841       0.670         0.669
```

poLCA: Erweiterungen (2)

Weitere Beispiele (im R-File):

- **calc.class.means(mod2)**

```
> calc.class.means(mod2)
   LIEEXAM LIEPAPER FRAUD COPYEXAM
C1    1.00    1.000 1.045    1.192
C2    1.63    1.704 1.167    1.315
```

- **overlap.rough(mod2 & mod3)**

C1	C2	n
1	1	13
0	1	41
1	0	265

C1	C2	C3	n
1	1	0	2
0	1	0	11
1	0	0	16
1	0	1	30
0	0	1	260

- **overlap.dunn(mod2)***

$$\text{DUNN}_K = \left(\frac{1}{n}\sum_g\sum_k p^2(k|g) - \frac{1}{K}\right) \Big/ \left(1 - \frac{1}{K}\right)$$

```
> overlap.dunn(mod2)
[1] 0.8019822
```

- **overlap.backer(mod2)***

$$\text{BACKER}_K = 1 - \frac{1}{n} \cdot \frac{2}{K-1} \cdot \sum_g \sum_k \sum_{k^* \neq k} \min\left(p(k|g), p(k^*|g)\right)$$

```
> overlap.backer(mod2)
[1] 0.9873538
```

Übersicht – Paket: tidyLPA

Veröffentlicht:

Rosenberg, Joshua & Beymer, Patrick & Anderson, Daniel & Schmidt, Jennifer am 10. Oktober, 2018

Ziel:

Schätzung von latent-profile Modellen für stetige (metrische) Klassifikationsvariablen (Indikatoren) unter Spezifikationen von ausgewählten Restriktionen und auf Basis von **mclust** bzw. **MPlus**

Verfügbar:

```
install.packages("tidyLPA")
library(tidyLPA)
```

tidyLPA: Vor- und Nachteile

Vorteile:

- Extrem simple Oberfläche für Schätzung von LPA-Modellen (Wrapper)
- Erlaubt die Spezifikation von ausgewählten Restriktionen
- Umfassende und moderne Fit-Statistiken verfügbar
- Mehrere Modelle können zeitgleich geschätzt und verglichen werden

Nachteile:

- Nur metrisch skalierte Variablen (→ Normalverteilung) verwendbar
- Keine großen zusätzlichen Funktionen (z.B. keine Kovariaten & FIML)

tidyLPA: Anwendungsbeispiel

Enthalten im Paket: `data("pisaUSA15", package = tidyLPA)`

Quelle: OECD PISA Studie 2015, Link

Eigenschaften:

- Schülerfragebogendaten aus der PISA-Studie 2015 in den USA
- Manifeste Variablen (Bsp. auf Zeile 1:100, ohne NAs & instrumental_mot):

1.	broad_interest	Maß für allgemeines persönliche Interesse
2.	enjoyment	Maß für persönlichen Spaß
3.	instrumental_mot	Maß der pers. instrumentellen Motivation
4.	self-efficacy	Maß für Selbstwirksamkeit

tidyLPA: Modellschätzung

Die Nutzung des Pakets erfolgt durch einen Befehl:

→ Die Modellschätzung durch estimate_profiles() erlaubt die Schätzung von mehreren unterschiedlichen Modellen gleichzeitig

→ Weitere Argumente anhand von ?mclust::mclust() spezifizierbar

Allgemein:

```
estimate_profiles(df, n_profiles, models = NULL, variances = "equal",
                  covariances = "zero", select_vars = NULL, ...)
```

Beispiel:

```
estimate_profiles(n_profiles = 1:3, variances = c("equal", "varying")
                  df = data, covariances = c("zero", "varying"))
```

tidyLPA: Modellschätzung (2)

Das Paket erlaubt es eines der folgenden 4 Modelle zu schätzen:

- Identisch fixierte Varianzen & auf 0 fixierte Kovarianzen (Model 1)
- Variierende Varianzen & auf 0 fixierte Kovarianzen (Model 2)
- Identisch fixierte Varianzen & Kovarianzen (Model 3)
- Variierende Varianzen & Kovarianzen (Model 6)

→ Die Modellparameterisierungen 4 & 5 sind ausschließlich unter Verwendung von MPlus verfügbar

tidyLPA: Verfügbare Outputs

- **get_fit(mod)**

Umfangreiche Liste an verschiedenen Fit Indizes

→ Für Details: Link (Unten)

- **compare_solutions(mod)**

Automatisierter Modellvergleich anhand Fit Indizes

```
> get_fit(mod) # Umfangreiche Liste an Fit Indizes
# A tibble: 6 × 18
  Model Classes LogLik   AIC   AWE   BIC  CAIC   CLC   KIC SABIC   ICL Entropy
  <dbl>   <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
1     1       1  -314.  639.  698.  655.  661.  629.  648.  636. -655.   1
2     1       2  -292.  605.  704.  630.  640.  586.  618.  598. -653.   0.649
3     1       3  -284.  596.  736.  632.  646.  570.  613.  587. -650.   0.806
4     6       1  -285.  588.  677.  611.  620.  572.  600.  582. -611.   1
5     6       2  -279.  595.  786.  644.  663.  558.  617.  584. -693.   0.426
6     6       3  -263.  583.  874.  657.  686.  527.  615.  565. -666.   0.860
# ℹ 6 more variables: prob_min <dbl>, prob_max <dbl>, n_min <dbl>, n_max <dbl>,
#   BLRT_val <dbl>, BLRT_p <dbl>
```

```
> compare_solutions(mod, statistics = c("CAIC", "BIC"))
Compare tidyLPA solutions:

 Model Classes CAIC    BIC
 1     1       660.721 654.721
 1     2       640.059 630.059
 1     3       645.589 631.589
 6     1       619.704 610.704
 6     2       662.507 643.507
 6     3       685.889 656.889

Best model according to CAIC is Model 6 with 1 classes.
Best model according to BIC is Model 6 with 1 classes.

An analytic hierarchy process, based on the fit indices
AIC, AWE, BIC, CLC, and KIC (Akogul & Erisoglu, 2017),
suggests the best solution is Model 6 with 1 classes.
```

tidyLPA: Verfügbare Outputs (2)

• **get_data(mod)**	Datensatz samt Klassenwahrscheinlich-keiten und det. Klassenzugehörigkeiten
• **get_estimates(mod)**	Übersicht der Schätzwerte
• **plot(mod)**	Grafik einer Vergleichsgröße nach Wahl für alle modellierten Modelle
• **plot_profiles(mod)**	Boxplot-Grafik der klassenspezifischen Variablenausprägungen
• **plot_density(mod)**	Grafiken zur klassenbedingten Dichte
• **plot_bivariate(mod)**	2-dim. Grafiken zu Korrelation & Dichte

Beispiel: plot_profiles(mod)

Übersicht – Paket: depmixS4

Veröffentlicht:

Visser, Ingmar and Speekenbrink, Maarten am 5. August, 2010

Ziel:

Schätzung von latent-transition Modellen (Hidden Markov Modellen) unter Spezifikationen von Restriktionen und jegliche Art von Indikatoren
→ Latent-class und latent-profile Modelle als Unterart schätzbar

Verfügbar:

```
install.packages("depmixS4")
library(depmixS4)
```

Übersicht – Paket: depmixS4 (2)

	Classification by	
Design	Means (continuous data) / item probabilities (categorical data)	Regression parameters
Cross-sectional	Latent profile analysis	Regression mixture models
	Latent class analysis	
Longitudinal	Repeated measures latent class analysis	Growth mixture models
	Latent transition analysis	

Wir sind eigentlich hier!

depmixS4: Vor- und Nachteile

Vorteile:

- Ermöglicht die Schätzung von Modellen mit fast jeder Variablenart und Kovariaten unter Spezifikation der zugrundeliegenden Verteilung
- Erlaubt die Spezifikation von Restriktionen auf Parameterebene

Nachteile:

- Keine Fit-Statistiken, Tests und ordinalen Variablen möglich
- Andere Terminologie als Grundlage (z.B. Classes = States)
- (Fast) Gar keine zusätzlichen Funktionen (Grafiken, Entropie, etc.)

depmixS4: Modellschätzung

Zur Nutzung werden 2 Befehle benötigt:

1. **Konstruktion eines Modells mithilfe der Funktion mix()**

Beispiel:

```
definition <- depmixs4::mix(list(LIEEXAM ~ 1, LIEPAPER ~ 1,
                                 FRAUD ~ 1, COPYEXAM ~ 1),
                            family = list(multinomial("identity"),
                                          multinomial("identity"),
                                          multinomial("identity"),
                                          multinomial("identity")),
                            data = cheating,
                            nstates = 2,
                            prior =~ GPA,
                            initdata = cheating)
```

→ **Es lassen sich eigene Verteilungs- und Link-Funktionen spezifizieren**

depmixS4: Modellschätzung (2)

2. Starten der Modellschätzung mithilfe der Funktion fit()

Allgemein:

```
fit(object, fixed=NULL, equal=NULL,
        conrows=NULL, conrows.upper=NULL, conrows.lower=NULL,
        method=NULL, verbose=FALSE,
        emcontrol=em.control(),
        solnpcntrl=list(rho = 1, outer.iter = 400, inner.iter = 800,
        delta = 1e-7, tol = 1e-8),
        donlpcntrl=donlp2Control(),
        ...)
```

Beispiel:

```
mod <- fit(dep.definition, verbose = T,
           fixed = NULL, equal = NULL)
```

depmixS4: Verfügbare Outputs

Der allgemeine Output erfolgt durch den summary()-Befehl:

```
> summary(mod) # Übersichtslegende zum Interpretieren:
Mixture probabilities model
      pr1       pr2
0.8384694 0.1615306

Response parameters
Resp 1 : multinomial
Resp 2 : multinomial
Resp 3 : multinomial
Resp 4 : multinomial
        Re1.1      Re1.2     Re2.1      Re2.2     Re3.1      Re3.2     Re4.1     Re4.2
St1 0.9836949 0.01630513 0.9710677 0.02893232 0.9629843 0.03701566 0.8181307 0.1818693
St2 0.4248042 0.57519577 0.4127219 0.58727813 0.7845967 0.21540330 0.6243790 0.3756210
```

depmixS4: Verfügbare Outputs (2)

Der allgemeine Output erfolgt durch den summary()-Befehl:

```
> summary(mod) # Übersichtslegende zum Interpretieren:
Mixture probabilities model
      pr1       pr2
0.8384694 0.1615306

Response parameters
Resp 1 : multinomial
Resp 2 : multinomial
Resp 3 : multinomial
Resp 4 : multinomial
        Re1.1      Re1.2     Re2.1      Re2.2     Re3.1      Re3.2     Re4.1     Re4.2
St1 0.9836949 0.01630513 0.9710677 0.02893232 0.9629843 0.03701566 0.8181307 0.1818693
St2 0.4248042 0.57519577 0.4127219 0.58727813 0.7845967 0.21540330 0.6243790 0.3756210
```

Sonstiges (bei anderen Verteilungen):
Re1.Intercept = Mittelwert
Re1.sd = Standardabweichung

→ Re2.2
Ausprägung 2 der zweiten Variable

Klassen

Erste definierte Variable
→ Re1.1 und Re1.2

Weitere Pakete

Die meisten Pakete sind im Zusammenhang mit spezifischen Studien entstanden. Allgemeinere Alternativen in Paketform sind hingegen:

- **tidySEM** Kann als Verallgemeinerung von tidyLPA angesehen werden und ermöglicht LCA & LPA auf ausschließlich einem von mehreren Variablentypen (z.B. nur ordinal)

→ Wrapper-Funktionen für leichtere Verwendung auf Basis von OpenMX

→ Restriktionen wie in tidyLPA spezifizierbar & allgemein mehr Umfang im Paket

- **OpenMX** Paket bietet (lediglich) einen Rahmen für die Erstellung von LCA & LPA Modellen jeglicher Form & Variablentypen

→ **Aber:** Algorithmus oder Programm praktisch selbst schreiben (High Know-How)

Weitere Pakete (2)

Die meisten Pakete sind im Zusammenhang mit spezifischen Studien entstanden. Allgemeinere Alternativen in Paketform sind hingegen:

- **randomLCA** Ermöglicht es LCA mit Random Effects und bis zu 2 Ebenen zu modellieren, aber sehr eingeschränkt

➢ Bei Verletzungen der lokalen Unabhängigkeit wird ein normalverteilter Random Effect anstelle von zusätzlichen Klassen modelliert, aber benötigt auch mehr Parameter und damit geringere Freiheitsgrade

➢ Beinhaltet Bestrafungsterm bei der Likelihood der Wahrscheinlichkeiten für bessere Standardfehlerberechnung und Bootstrap Standardfehler-Funktion

➢ Random Effect- und Standardmodell nicht genestet (Unvergleichbar)

Quellenangabe

- Bauer, J. (2022). A Primer to Latent Profile and Latent Class Analysis. In: Goller, M., Kyndt, E., Paloniemi, S., Damşa, C. (eds) Methods for Researching Professional Learning and Development. Professional and Practice-based Learning, vol 33. Springer, Cham. Link
- Rudnev, M. (2016, December 28). *Ways to do Latent Class Analysis in R*. Elements of Cross-cultural Research. Link
- Mair, P., Rosseel, Y., Gruber, K. (2023, December 15). *Latent Class and Profile Analysis*. CRAN TASK VIEW: Psychometric Models and Methods. Link
- *Bacher, J.; Pöge, A.; Wenzig, K. (2010): Clusteranalyse. Anwendungsorientierte Einführung in Klassifikationsverfahren. 3. Aufl. München.*
- Linzer, D. A., & Lewis, J. B. (2011). poLCA: An R Package for Polytomous Variable Latent Class Analysis. *Journal of Statistical Software*, *42*(10), 1–29. Link, R-Manual

Quellenangabe (2)

- Rosenberg et al., (2018). tidyLPA: An R Package to Easily Carry Out Latent Profile Analysis (LPA) Using Open-Source or Commercial Software. Journal of Open-Source Software, 3(30), 978, Link, R-Manual, Supplemental Material
- Visser, I., & Speekenbrink, M. (2010). depmixS4: An R Package for Hidden Markov Models. *Journal of Statistical Software*, *36*(7), 1–21. Link, R-Manual
- Van Lissa, C. J., Garnier-Villarreal, M., & Anadria, D. (2024). Recommended Practices in Latent Class Analysis Using the Open-Source R-Package tidySEM. *Structural Equation Modeling: A Multidisciplinary Journal*, *31*(3), 526–534. Link, Supplemental Material
- Neale, M.C., Hunter, M.D., Pritikin, J.N. *et al.* OpenMx 2.0: Extended Structural Equation and Statistical Modeling. *Psychometrika* 81, 535–549 (2016). Link, R-Manual, Website
- Beath, K. J. (2017). randomLCA: An R Package for Latent Class with Random Effects Analysis. *Journal of Statistical Software*, *81*(13), 1–25. Link, R-Manual