STAT 3690 Lecture Note

Part VIII: Factor analysis

Zhiyang Zhou (zhiyang.zhou@umanitoba.ca, zhiyanggeezhou.github.io)

2023/Mar/27 15:13:22

Factor analysis

Latent variable model

- latent/unobservable variables give rise to observed data through a specific model, i.e., a regression model with unobservable covariates
- Factor analysis model is a special kind of latent variable model

Population version

- Model
$$\boldsymbol{Y} - \boldsymbol{\mu} = \mathbf{L}\boldsymbol{F} + \boldsymbol{E}$$
  - $\boldsymbol{Y} = [Y_1, \ldots, Y_p]^\top \sim (\boldsymbol{\mu}, \boldsymbol{\Sigma})$: random & observable p-vector
  - $\mathbf{L} = [\ell_{ij}]_{p\times q}$: fixed & unknown, a matrix of factor loadings
    - ℓ_{ij}: the contribution of jth factor to Y_i
  - $\boldsymbol{F} \sim (\mathbf{0}, \mathbf{I}_q)$: random & unobservable, q-vector of latent/common factors
  - $\boldsymbol{E} \sim (\mathbf{0}, \boldsymbol{\Psi})$: random & unobservable, p-vector of error/specific factors, with $\boldsymbol{\Psi} = \mathrm{diag}(\psi_1, \ldots, \psi_p)$ and $\mathrm{cov}(\boldsymbol{F}, \boldsymbol{E}) = \mathbf{0}$
- Covariance structure
  - $\mathrm{var}(\boldsymbol{Y}) = \boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^\top + \boldsymbol{\Psi}$
    - I.e., $\mathrm{var}(Y_i) = \sum_{j=1}^q \ell_{ij}^2 + \psi_i$
  - $\mathrm{cov}(\boldsymbol{Y}, \boldsymbol{F}) = \mathbf{L}$
  - $\sum_{i=1}^p \ell_{ij}^2$: the variability contributed by the jth latent factor

Sample version

- Model
$$\boldsymbol{Y}_i - \boldsymbol{\mu} = \mathbf{L}\boldsymbol{F}_i + \boldsymbol{E}_i, \quad i = 1, \ldots, n$$
  - $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n \overset{\text{iid}}{\sim} \boldsymbol{Y}$
  - $\boldsymbol{F}_1, \ldots, \boldsymbol{F}_n \overset{\text{iid}}{\sim} \boldsymbol{F}$
  - $\boldsymbol{E}_1, \ldots, \boldsymbol{E}_n \overset{\text{iid}}{\sim} \boldsymbol{E}$

Estimating L and Ψ

- Selection of q, i.e., the number of latent factors, with one of the following rules
  - PCA stopping rule

  - Taking q such that $\sum_{j=1}^{q}\sum_{i=1}^{p}\ell_{ij}^2/\text{tr}(\mathbf{S})$ is over a preset percentage, where $\mathbf{S}=(n-1)^{-1}\sum_{i=1}^{n}(\boldsymbol{Y}_i-\bar{\boldsymbol{Y}})(\boldsymbol{Y}_i-\bar{\boldsymbol{Y}})^\top$
    * $\sum_{i=1}^{q}\ell_{ij}^2/\text{tr}(\mathbf{S})$: the proportion of variation explained by the jth latent factor
  - Taking q as the number of positive eigenvalues of $\mathbf{S}$
  - Taking q as the number of eigenvalues of $\mathbf{S}$ that are above average
  - Taking q as the number of eigenvalues of correlation matrix greater than one
  - According to domain-knowledge expertise
- PC method
  1. Determine q
  2. Pick up the q largest eigenvalues $\lambda_1,\ldots,\lambda_q$ of $\mathbf{S}$ and corresponding eigenvectors $w_1,\ldots,w_q$
  3. $\widehat{\mathbf{L}}=[\sqrt{\lambda_1}w_1,\ldots,\sqrt{\lambda_q}w_q]_{p\times q}$ and $\widehat{\boldsymbol{\Psi}}=\text{diag}(\mathbf{S}-\widehat{\mathbf{L}}\widehat{\mathbf{L}}^\top)$

- Exercise 8.1: `psych::bfi` involves 2800 subjects, covering their 25 personality assessments, gender, education and age.

```
install.packages(c('psych'))
library(psych)
library(tidyverse)
options(digits = 4)
head(psych::bfi)
data = bfi %>%
  select(-gender, -education, -age) %>% # Remove gender, education and age
  filter(complete.cases(.))  # keep complete data only
S = cov(data)
fa_pc = prcomp(data) # decompose the covariance matrix

# PCA stopping rule
(q = which(cumsum(fa_pc$sdev^2)/sum(fa_pc$sdev^2)>.9)[1])
# the overall proportion of variation explained by latent factors
(q = which(
  cumsum(sort(colSums((fa_pc$rotation %*% diag(fa_pc$sdev))^2), decreasing = T))/sum(diag(S))>.9
)[1])
# the number of eigenvalues above the average
(q = sum(eigen(S, only.values = T)$values > mean(eigen(S, only.values = T)$values)))
# the number of eigenvalues greater than one for the correlation matrix
(q = sum(eigen(cor(data))$values > 1))

L_pc = fa_pc$rotation[,1:q] %*% diag(fa_pc$sdev[1:q])
Psi_pc = diag(diag(S - tcrossprod(L_pc)))

S_pc = tcrossprod(L_pc) + Psi_pc
lattice::levelplot(S - S_pc, scales=list(x=list(rot=90))) # fitting error
lattice::levelplot((S - S_pc)/S, scales=list(x=list(rot=90))) # difference in percentage
```

- ML method
  - Assuming
    * $\boldsymbol{F}\sim\text{MVN}_q(\mathbf{0},\mathbf{I})$
    * $\boldsymbol{E}\sim\text{MVN}_p(\mathbf{0},\boldsymbol{\Psi})$
    * Diagonal $\mathbf{L}^\top\boldsymbol{\Psi}^{-1}\mathbf{L}$
  - Resorting to `R` functions `factanal` or `psych::fa`

```r
install.packages(c('psych'))
library(psych)
library(tidyverse)
options(digits = 4)
head(psych::bfi)
data = bfi %>%
  select(-gender, -education, -age) %>% # Remove gender, education and age
  filter(complete.cases(.))  # keep complete data only
S = cov(data)

# the number of eigenvalues greater than one for the correlation matrix
(q = sum(eigen(cor(data))$values > 1))

# apply functions factanal OR psych::fa
## Note that the following four are all working on the correlation matrix
fa_ml_11 <- factanal(x = data, factors = q, rotation = "none")
fa_ml_12 <- factanal(covmat = cor(data), factors = q, rotation = "none")
fa_ml_13 <- factanal(covmat = S, factors = q, rotation = "none")
fa_ml_14 <- psych::fa(r = data, covar = F, nfactors = q, rotate = "none", fm = "ml")
head(fa_ml_11$loadings-fa_ml_12$loadings)
head(fa_ml_12$loadings-fa_ml_13$loadings)
head(fa_ml_13$loadings-fa_ml_14$loadings)
## Note that the next one is working on the covariance matrix
fa_ml_2 <- psych::fa(r = data, covar = T, nfactors = q, rotate = "none", fm = "ml")

L_ml <- fa_ml_2$loadings
Psi_ml <- diag(fa_ml_2$uniquenesses)
S_ml = tcrossprod(L_ml) + Psi_ml
lattice::levelplot(S - S_ml,
                   scales=list(x=list(rot=90)), xlab = "", ylab = "")
lattice::levelplot((S - S_ml)/S,
                   scales=list(x=list(rot=90)), xlab = "", ylab = "")
```

- Comments on the estimation of $\mathbf{L}$ and $\mathbf{\Psi}$
  - More methods other than ML and PC
  - Different statistical softwares/packages may apply different methods
    - Have to look into help manuals to figure out what is going on
  - Compare the outputs of multiple estimation methods
    - For a good fit, similar answers would be reached regardless of method

---

Factor rotation

- $\mathbf{L}$ is not uniquely defined: if $\boldsymbol{Y} - \boldsymbol{\mu} = \mathbf{L}\boldsymbol{F} + \boldsymbol{E}$, then $\boldsymbol{Y} - \boldsymbol{\mu} = \widetilde{\mathbf{L}}\widetilde{\boldsymbol{F}} + \boldsymbol{E}$, where
  - $\widetilde{\mathbf{L}} = \mathbf{L}\mathbf{P}$ and $\widetilde{\boldsymbol{F}} = \mathbf{P}^\top \boldsymbol{F}$ with $\mathbf{P}$ a $q \times q$ orthogonal matrix ($\mathbf{P}^{-1} = \mathbf{P}^\top$)
- A blessing to improve interpretation: pick up a $\mathbf{P}$ such that $\widetilde{\boldsymbol{F}}$ is more interpretable; to ease the interpretation, we want:
  - Each entry of $\boldsymbol{Y}$ to have large loadings for merely one latent factor and negligible loadings for remaining ones

- varimax: find $\mathbf{P}$ to maximize the sum of variance of squared (scaled) loadings over all the latent factors

$$\sum_{j=1}^{q}\left\{\frac{1}{p}\sum_{i=1}^{p}\tilde{\ell}_{ij}^{*4}-\left(\frac{1}{p}\sum_{i=1}^{p}\tilde{\ell}_{ij}^{*2}\right)^2\right\}$$

  - $\tilde{\ell}_{ij}^{*}=\tilde{\ell}_{ij}/\sqrt{\sum_{j=1}^{q}\tilde{\ell}_{ij}^{2}}$ with $\tilde{\ell}_{ij}$ the (i,j)-th entry of $\widetilde{\mathbf{L}}=\mathbf{LP}$

```
install.packages(c('psych'))
library(psych)
library(tidyverse)
options(digits = 4)
head(psych::bfi)
data = bfi %>%
  select(-gender, -education, -age) %>% # Remove gender, education and age
  filter(complete.cases(.))  # keep complete data only
S = cov(data)
q = sum(eigen(cor(data))$values > 1)

# Rotating loadings which are obtained via PC method
fa_pc = prcomp(data) # decompose the covariance matrix
L_pc = fa_pc$rotation[,1:q] %*% diag(fa_pc$sdev[1:q])
L_pc_rot = varimax(L_pc)$loadings
varimax(L_pc)$rotmat
## Plot loading matrices
lattice::levelplot(unclass(t(L_pc)),
                   xlab = "", ylab = "", scales=list(x=list(rot=90)))
lattice::levelplot(unclass(t(L_pc_rot)),
                   xlab = "", ylab = "", scales=list(x=list(rot=90)))

# Rotating loadings which are obtained via ML method
fa_ml <- psych::fa(r = data, covar = T, nfactors = q, rotate = "none", fm = "ml")
L_ml <- fa_ml$loadings
fa_ml_rot <- psych::fa(r = data, covar = T, nfactors = q, rotate = "varimax", fm = "ml")
L_ml_rot <- fa_ml_rot$loadings
head(L_ml-L_ml_rot)
fa_ml_rot$rot.mat # rotation matrix
## Plot loading matrices
lattice::levelplot(unclass(t(L_ml)),
                   xlab = "", ylab = "", scales=list(x=list(rot=90)))
lattice::levelplot(unclass(t(L_ml_rot)),
                   xlab = "", ylab = "", scales=list(x=list(rot=90)))
```

- Comments on factor rotation
  - Especially useful with loadings obtained through ML
  - Sometimes used even for loadings in PCA

Factor scores

- Weighted least square (WLS) method
  - Given $\bar{\boldsymbol{Y}}$, $\widehat{\mathbf{L}}$, and $\widehat{\boldsymbol{\Psi}}$, then, for the ith observation $\boldsymbol{Y}_i$,

$$\widehat{\boldsymbol{F}}_i=(\widehat{\mathbf{L}}^{\top}\widehat{\boldsymbol{\Psi}}^{-1}\widehat{\mathbf{L}})^{-1}\widehat{\mathbf{L}}^{\top}\widehat{\boldsymbol{\Psi}}^{-1}(\boldsymbol{Y}_i-\bar{\boldsymbol{Y}})$$

∗ I.e., the minimizer of $(\boldsymbol{Y}_i - \bar{\boldsymbol{Y}} - \widehat{\mathbf{L}}\boldsymbol{F})^\top \widehat{\boldsymbol{\Psi}}^{-1}(\boldsymbol{Y}_i - \bar{\boldsymbol{Y}} - \widehat{\mathbf{L}}\boldsymbol{F})$ with respect to $\boldsymbol{F}$

```
install.packages(c('psych'))
library(psych)
library(tidyverse)
options(digits = 4)
head(psych::bfi)
data = bfi %>%
  select(-gender, -education, -age) %>%
  filter(complete.cases(.))

# WLS factor scores following formula
fa_ml = psych::fa(r = data, covar = T, nfactors=q, rotate="varimax", fm="ml")
L_ml = fa_ml$loadings
Psi_ml = diag(fa_ml$uniquenesses)
Psi_ml_inv = diag(fa_ml$uniquenesses^-1)
Weight_mat = solve(t(L_ml) %*% Psi_ml_inv %*% L_ml) %*% t(L_ml) %*% Psi_ml_inv
scores_wls = scale(data, center = T, scale = F) %*% t(Weight_mat)
head(scores_wls)
```

- Regression method
  - Assuming $\boldsymbol{F} \sim \text{MVN}_q(\mathbf{0}, \mathbf{I})$ and $\boldsymbol{E} \sim \text{MVN}_p(\mathbf{0}, \boldsymbol{\Psi})$,

$$\begin{bmatrix} \boldsymbol{Y} - \boldsymbol{\mu} \\ \boldsymbol{F} \end{bmatrix} \sim \text{MVN}_{p+q}\left(\mathbf{0}, \begin{bmatrix} \mathbf{L}\mathbf{L}^\top + \Psi & \mathbf{L} \\ \mathbf{L}^\top & \mathbf{I} \end{bmatrix}\right)$$

    and hence

$$\boldsymbol{F} \mid \boldsymbol{Y} \sim \text{MVN}_p(\mathbf{L}^\top(\mathbf{L}\mathbf{L}^\top + \boldsymbol{\Psi})^{-1}(\boldsymbol{Y} - \boldsymbol{\mu}), \mathbf{I} - \mathbf{L}^\top(\mathbf{L}\mathbf{L}^\top + \boldsymbol{\Psi})^{-1}\mathbf{L})$$

  - Given $\bar{\boldsymbol{Y}}$, $\widehat{\mathbf{L}}$, and $\widehat{\boldsymbol{\Psi}}$, esitmate $\boldsymbol{F}_i$ by

$$\widehat{\boldsymbol{F}}_i = \widehat{\mathbf{L}}^\top(\widehat{\mathbf{L}}\widehat{\mathbf{L}}^\top + \widehat{\boldsymbol{\Psi}})^{-1}(\boldsymbol{Y}_i - \bar{\boldsymbol{Y}})$$

    OR

$$\widehat{\boldsymbol{F}}_i = \widehat{\mathbf{L}}^\top \mathbf{S}^{-1}(\boldsymbol{Y}_i - \bar{\boldsymbol{Y}})$$

```
# Regression based factor scores following formula
fa_ml = psych::fa(r=data, covar=T, nfactors=q, rotate="varimax", fm ="ml")
L_ml = fa_ml$loadings
Psi_ml = diag(fa_ml$uniquenesses)
Weight_mat = t(L_ml) %*% solve(cov(data))
scores_reg = scale(data, center = T, scale = F) %*% t(Weight_mat)
head(scores_reg)
```

- Comments on factor scores
  - More methods available
  - No uniformly superior way

Summary on factor analysis

- What we discussed is "exploratory" factor analysis

  - "Confirmatory" factor analysis would make stronger assumptions about the nature of the latent factors and perform statistical inference.
  - There are choices to make at every stage of factor analysis: estimation method, number of factors, factor rotation, and score estimation.
    * Too flexiable to be tracked
    * Close to an "art"
- General strategy for factor analysis
  1. Perform a PC factor analysis
     - It may help you identify potential outliers
  2. Perform an ML factor analysis.
     - Try a varimax rotation to see if it makes sense
  3. Compare the solutions of both methods to see if they generally agree.
  4. Repeat for different number of common factors q and check if adding more factors may improve the interpretation
  5. For large datasets, you can split your data, run the same model on both subsets, and compare the loadings to see if they generally agree

An example of factor analysis

- `state.x77` contains general information about all 50 US states
  - Population
  - Income per capita
  - Illiteracy rate
  - Life expectancy
  - Murder rate
  - High-school graduation rate
  - Average number of freezing degree days (with the temperature lower than 0 °C)
  - Total area

```
install.packages(c('psych', 'GGally', 'maps'))
library(psych)
options(digits = 2)

dataset = state.x77
R = cor(dataset)

## Pairs plots
pairs(dataset)
GGally::ggpairs(as.data.frame(dataset))

## Factor analysis via PC method
fa_pc = princomp(dataset, cor = T)
(q = which(cumsum((fa_pc$sdev)^2)/sum((fa_pc$sdev)^2)>.9)[1])
L_pc = fa_pc$loadings[,1:q] %*% diag(fa_pc$sdev[1:q])
Psi_pc = diag(diag(R - L_pc %*% t(L_pc)))
R_pc = L_pc %*% t(L_pc) + Psi_pc
lattice::levelplot(unclass((R - R_pc)/R),
                   xlab = "", ylab = "",scales=list(x=list(rot=90))) # visualize the fitting error
scores_pc = as.data.frame(
  scale(dataset, center = T, scale = T) %*% solve(R) %*% L_pc) # regression-based score
GGally::ggpairs(scores_pc)

## What states have the outlying values?
```

```r
scores_pc[!(
  abs(scores_pc$V1)<3 &
  abs(scores_pc$V2)<3 &
  abs(scores_pc$V3)<3 &
  abs(scores_pc$V4)<3 &
  abs(scores_pc$V5)<3
),]

## Factor analysis via ML method
fa_ml = psych::fa(r=dataset, covar=F, nfactors=q, rotate="varimax", fm ="ml")
L_ml = fa_ml$loadings
Psi_ml = diag(fa_ml$uniquenesses)
R_ml = L_ml %*% t(L_ml) + Psi_ml
lattice::levelplot(unclass((R - R_ml)/R), xlab = "", ylab = "") # visualize the fitting error
scores_ml = as.data.frame(
  scale(dataset, center = T, scale = T) %*% solve(R) %*% L_ml) # regression-based score

## Compare both loadings and scores (after rotation)
L_pc_rot = varimax(L_pc)$loadings
scores_pc_rot = as.data.frame(
  scale(dataset, center = T, scale = T) %*% solve(R) %*% L_pc_rot) # regression-based score
cor(scores_pc_rot, scores_ml) # look at the agreement of both scores

## Interpret the factors
L_pc # not interpretable
L_pc_rot
# factor1 on life span, factor2 on area, factor3 on population
# factor4 on weather and education, factor5 on education and income
L_ml
# ML1 on life span, ML4 on education and income
# ML5 on warmth level, ML2 on population, ML3 on area

# Association between each factor and original variables
cor(scores_pc, dataset)
cor(scores_ml, dataset)

## Plot the factor scores on the map
library(maps)
library(ggplot2)
states <- map_data("state")
row.names(scores_ml) = tolower(row.names(scores_ml))
scores_ml = tibble::rownames_to_column(scores_ml, var='region')
data_plot = dplyr::inner_join(
  x = scores_ml,
  y = states,
  by = "region"
)
ggplot(data = data_plot) +
  geom_polygon(aes(x = long, y = lat, fill = ML1, group = group)) +
  ggtitle("1st Factor Scores")
ggplot(data = data_plot) +
  geom_polygon(aes(x = long, y = lat, fill = ML2, group = group)) +
  ggtitle("2nd Factor Scores")
```

```r
## Try different values of q
(q = which(
  cumsum(sort(colSums((fa_pc$loadings %*% diag(fa_pc$sdev))^2), decreasing = T))/
    sum(diag(R))>.9
)[1])
(q = sum(eigen(R, only.values = T)$values > mean(eigen(R, only.values = T)$values)))
(q = sum(eigen(R, only.values = T)$values > 1))

L_pc = varimax(fa_pc$loadings[,1:q] %*% diag(fa_pc$sdev[1:q]))$loadings
Psi_pc = diag(diag(R - L_pc %*% t(L_pc)))
R_pc = L_pc %*% t(L_pc) + Psi_pc
lattice::levelplot(unclass((R - R_pc)/R), xlab = "", ylab = "") # visualize the fitting error
scores_pc = as.data.frame(
  scale(dataset, center = T, scale = T) %*% solve(R) %*% L_pc) # regression-based score
GGally::ggpairs(scores_pc)
scores_pc[!(
  abs(scores_pc$V1)<3 &
  abs(scores_pc$V2)<3 &
  abs(scores_pc$V3)<3
),]

fa_ml = psych::fa(r=dataset, covar=F, nfactors=q, rotate="varimax", fm ="ml")
(L_ml = fa_ml$loadings)
Psi_ml = diag(fa_ml$uniquenesses)
R_ml = L_ml %*% t(L_ml) + Psi_ml
sum(colSums(L_ml^2))/sum(diag(R)) # proportion of variance explained
lattice::levelplot(unclass((R - R_ml)/R), xlab = "", ylab = "") # visualize the fitting error
```