STAT 3690 Lecture 27

zhiyanggeezhou.github.io

Zhiyang Zhou (zhiyang.zhou@umanitoba.ca)

Apr 04, 2022

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 days below 0C
  - Total area

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

dataset = state.x77
R = cor(dataset)

pairs(dataset)
```

```r
GGally::ggpairs(as.data.frame(dataset))

## PC factor analysis
pc_decomp = princomp(dataset, cor = T)
(q = which(cumsum((pc_decomp$sdev)^2)/sum((pc_decomp$sdev)^2)>.9)[1])
L_pc = pc_decomp$loadings[,1:q] %*% diag(pc_decomp$sdev[1:q])
Psi_pc = diag(diag(R - L_pc %*% t(L_pc)))
R_pc = L_pc %*% t(L_pc) + Psi_pc
sum(colSums(L_pc^2))/sum(diag(R)) # proportion of variance explained
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 sco
GGally::ggpairs(scores_pc)

## What states have the outlying values?
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
),]

## ML factor analysis
fa_decomp = psych::fa(r=dataset, covar=F, nfactors=q, rotate="varimax", fm ="ml")
L_ml = fa_decomp$loadings
Psi_ml = diag(fa_decomp$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
scores_ml = as.data.frame(scale(dataset, center = T, scale = T) %*% solve(R) %*% L_ml) # regression sco

## Compare both loadings and scores
L_pc_rot = varimax(L_pc)$loadings
scores_pc_rot = as.data.frame(scale(dataset, center = T, scale = T) %*% solve(R) %*% L_pc_rot) # regres
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 weather and education, 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")

## Try different values of q
(q = which(
  cumsum(sort(colSums((pc_decomp$loadings %*% diag(pc_decomp$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(pc_decomp$loadings[,1:q] %*% diag(pc_decomp$sdev[1:q]))$loadings
Psi_pc = diag(diag(R - L_pc %*% t(L_pc)))
R_pc = L_pc %*% t(L_pc) + Psi_pc
sum(colSums(L_pc^2))/sum(diag(R)) # proportion of variance explained
lattice::levelplot(unclass((R - R_pc)/R), xlab = "", ylab = "") # visualize the fitting error
scores_pc = as.data.frame(scale(dataset, center = T, scale = T) %*% solve(R) %*% L_pc) # regression sco
GGally::ggpairs(scores_pc)
scores_pc[!(
  abs(scores_pc$V1)<3 &
  abs(scores_pc$V2)<3 &
  abs(scores_pc$V3)<3
),]

fa_decomp = psych::fa(r=dataset, covar=F, nfactors=q, rotate="varimax", fm ="ml")
L_ml = fa_decomp$loadings
Psi_ml = diag(fa_decomp$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
```