CFA with Ordinal Data: Estimation Methods and 'Robust' Standard Errors

PSYC520 Final Project Code

Table of contents

Analyses

```r
set.seed(7)

library(tidyverse)
library(lavaan)
library(semPlot, include.only = "semPaths")
library(modelsummary, include.only = "msummary")
library(semTools, include.only = "compRelSEM")
library(semptools)
library(flextable)
library(psych)
library(dplyr)
```

We will conduct a CFA on the five-item Subjective Emptiness Scale (Price et al., 2022), items of which are rated on a 1-4 Likert Scale (1=not at all true to 4=very true). We will be using data from the second study (n = 1,067) in a series of three studies the authors conducted to develop and validate a self-report measure on subjective emptiness. The participants for the second study were recruited online via ads on social media platforms and study recruitment listing websites, and 94% reported having received a psychiatric diagnosis.

```r
# read and format data
tmp_path <- tempfile(fileext = "csv")  # temporary file
download.file("https://osf.io/download/c3akx",
              destfile = tmp_path)
emp <- read.csv(tmp_path, col.names = cn)
emp2 <- emp %>% filter(Sample == 2)
emp2 <- emp2[, c(1:28, 78:114)] # select columns
names(emp2)[24:29] <- c("i1", "i2", "i3", "i4", "i5")
#c("empty", "absent", "unfulfilled", "exist", "alone")
ses <- emp2 %>% select(c("i1", "i2", "i3", "i4", "i5", "Age", "Gender"))
ses[ses == 999] <- NA ## recode 999 as NA

#max(ses, na.rm = TRUE)

# As the goal is to illustrate differences in SE, and as
# SE goes down as N goes up, we initially conducted analyses using
# randomly sample of 300 out of all the observations, and reran the
# analyses with the full sample after receiving feedback during the
# presentation.
#set.seed(7)
#ses$id <- 1:length(ses$i1)
#ids <- sample(ses$id, size = 300)
```

```r
#ses <- ses[ses$id %in% ids,]
#ses <- ses[, -6]

dim(ses)
```

```
[1] 1067    7
```

```r
mean(ses$Age, na.rm = TRUE)
```

```
[1] 29.80983
```

```r
sd(ses$Age, na.rm = TRUE)
```

```
[1] 11.50751
```

```r
round(table(ses$Gender)/sum(table(ses$Gender)), 3)
```

```

    1     2 
0.323 0.677 
```

```r
round(colMeans(ses, na.rm = TRUE),2)
```

```
    i1     i2     i3     i4     i5    Age Gender 
  2.60   2.48   2.72   2.41   2.50  29.81   1.68 
```

```r
round(sqrt(diag(var(ses, na.rm = TRUE))), 2)
```

```
    i1     i2     i3     i4     i5    Age Gender 
  1.08   1.14   1.10   1.24   1.16  11.53   0.47 
```

```r
#compRelSEM(cfa_uls)
```

[1]

```r
ses <- na.omit(ses) # drop 14 observations listwise
ses <- ses[,1:5]
```

[1] All analyses were first conducted with the full dataset, and as expected with large N, the SE were quite small. We then took a random subset of 300 observations from this dataset to work with larger standard errors for illustrative purposes, and based on feedback received during the presentation, went back to the initial full sample of 1,053.

Thresholds

Below, we build a contingency table for the first two items X_1 and X_2 (empty, absent) in the Subjective Emptiness Scale:

```
emp_abs <- as.matrix(table(ses[1:2])/sum(table(ses[1:2])))
# marginal proportions
emp_abs <- rbind(emp_abs, as.numeric(colSums(emp_abs)))
emp_abs <- cbind(emp_abs, as.numeric(rowSums(emp_abs)))
rownames(emp_abs) <- c("1", "2", "3", "4", "P_absent(x)")
colnames(emp_abs)  <- c("1", "2", "3", "4", "P_empty(x)")
round(emp_abs, 3)
```

```
                1     2     3     4 P_empty(x)
1           0.125 0.041 0.015 0.007      0.189
2           0.095 0.125 0.056 0.029      0.305
3           0.030 0.055 0.079 0.066      0.229
4           0.018 0.032 0.068 0.160      0.278
P_absent(x) 0.268 0.253 0.217 0.261      1.000
```

Using the cumulative marginal proportions from the contingency table above, we can estimate the thresholds for variable empty as:

```
# c("empty", "absent", "unfulfilled", "exist", "alone")
(empty_thr <- qnorm(cumsum(emp_abs[1:4, 5])))
```

```
          1           2           3           4
-0.88270693 -0.01680769  0.58945580         Inf
```

```
#and for the remaining items as:
(absent_thr <- qnorm(cumsum(prop.table(table(ses$i2)))))
```

```
          1           2           3           4
-0.61826834  0.05284626  0.63874550  8.20953615
```

```
(unf_thr <- qnorm(cumsum(prop.table(table(ses$i3)))))
```

```
         1          2          3          4
-0.9187286 -0.1957222  0.4652360        Inf
```

```
(exist_thr <- qnorm(cumsum(prop.table(table(ses$i4)))))
```

```
         1          2          3          4
-0.4202135  0.1494152  0.5112223        Inf
```

```r
(alone_thr <- qnorm(cumsum(prop.table(table(ses$i5)))))
```

```
         1           2           3           4 
-0.64169260  0.07209195  0.56112453         Inf 
```

As we will later see, these values match the thresholds reported by `lavaan::lavCor()`.

Polychoric correlation matrix

We now build a polychoric correlation matrix for the Subjective Emptiness Scale items.

```r
# Function that takes in a correlation, a count table for two
# items, an two vectors of thresholds and returns the sum of
# the product of the category frequencies and the logarithm
# of the cell probabilities.
ll <- function(rho, ct, x1_th, x2_th) {
  # non redundant pairs of lower and upper thresholds
  llim <- as.matrix(expand.grid(c(-Inf, x1_th), c(-Inf, x2_th)))
  ulim <- as.matrix(expand.grid(c(x1_th, Inf), c(x2_th, Inf)))

  cellprobs <-
    vapply(seq_len(nrow(llim)),
           function(i, cor = rho) {
             mvtnorm::pmvnorm(llim[i, ], ulim[i, ],
                              corr = matrix(c(1, cor, cor, 1),
                                            nrow = 2))
             },
           FUN.VALUE = numeric(1))
  return(sum(ct * log(cellprobs)))
}
```

```r
# Maximize log likelihood f() for non-redundant item pairs
ests <- c()
thr <- list(empty_thr, absent_thr, unf_thr, exist_thr,
            alone_thr)
ijs <- combn(1:5, 2)
for (col in seq_len(ncol(ijs))) {
  i <- ijs[1, col]
  j <- ijs[2, col]
  ests <- c(ests,
           optim(par = 0, # initial value
                 fn = ll, # function to maximize
```

```r
                    ct = table(ses[, c(i, j)]),
                    # 1st item thresholds (excluding Inf)
                    x1_th = thr[[i]][1:3],
                    # 2nd item thresholds (excluding Inf)
                    x2_th = thr[[j]][1:3],
                    lower = -.99, upper = .99,
                    # allows box constraints
                    method = "L-BFGS-B",
                    # maximize the function
                    control = list(fnscale = -1))$par)
}

mt <- diag(5)
mt[lower.tri(mt, diag = FALSE)] <- round(ests, 3)
S <- as.data.frame(rstatix::pull_lower_triangle(mt, diag = 1))
rownames(S) <- colnames(S) <- c("i1", "i2", "i3", "i4", "i5")
S # input correlation matrix (= cov since variances are 1)
```

```
      i1     i2     i3     i4 i5
i1     1
i2 0.668      1
i3 0.653 0.687      1
i4 0.682 0.667 0.686      1
i5 0.689 0.671 0.677 0.673  1
```

We can confirm that the polychoric correlation matrix computed as above matches the polychoric correlation matrix computed by R:

Table 1: Polychoric correlation estimates

	est	se	ci.lower	ci.upper
s12	0.668	0.020	0.629	0.708
s13	0.653	0.021	0.612	0.695
s14	0.682	0.021	0.641	0.723
s15	0.689	0.019	0.651	0.726
s23	0.687	0.020	0.649	0.726
s24	0.667	0.021	0.625	0.708
s25	0.671	0.021	0.629	0.712
s34	0.686	0.020	0.646	0.725
s35	0.677	0.021	0.636	0.718
s45	0.673	0.022	0.630	0.716

Table 2: Threshold estimates

	est	se	ci.lower	ci.upper
t11	-0.883	0.045	-0.971	-0.795
t12	-0.017	0.039	-0.093	0.059
t13	0.589	0.041	0.508	0.671
t21	-0.618	0.042	-0.700	-0.537
t22	0.053	0.039	-0.023	0.129
t23	0.639	0.042	0.557	0.721
t31	-0.919	0.045	-1.008	-0.830
t32	-0.196	0.039	-0.272	-0.119
t33	0.465	0.040	0.386	0.544
t41	-0.420	0.040	-0.499	-0.342
t42	0.149	0.039	0.073	0.226
t43	0.511	0.041	0.431	0.591
t51	-0.642	0.042	-0.724	-0.560
t52	0.072	0.039	-0.004	0.148
t53	0.561	0.041	0.481	0.642

Estimation of model parameters

Having obtained the threshold and polychoric correlation estimates, we can proceed to fit the model.

```
s_lower <- coef(pcorr)[1:10] # polychoric corr matrix lower triangle

# function to compute the implied correlation matrix lower triangle
# (thelatent variables were standardized)
sigma_lower <- function(lambdas) {
  pc <- lambdas %*% t(lambdas)
  return(pc[lower.tri(pc)])
}
# asymptotic covariance matrix of the matrix of sample polychoric correlations
a_cov_mat <- vcov(pcorr)[1:10, 1:10]
w_mat <- diag(a_cov_mat)
# asymptotic standard errors
asymptotic_se <- sqrt(diag(a_cov_mat))
```

```
# Fit functions

# Takes in loadings, the lower triangle of the sample
# polychoric correlations matrix (s), and the asymptotic
```

```
# covariance matrix (weight matrix)
wls_fit <-
  function(lambdas, s = s_lower, w = a_cov_mat) {
    sigma <- sigma_lower(lambdas)
    (t(s - sigma) %*% matlib::inv(w)) %*% (s - sigma)
}

# Takes in loadings, the lower triangle of the sample
# polychoric correlations matrix (s), and the diagonals of
# the asymptotic covariance matrix (weight vector, diag(w))
dwls_fit <-
  function(lambdas, s = s_lower, w = diag(a_cov_mat)) {
    sigma <- sigma_lower(lambdas)
    (t(s - sigma) * (1 /w )) %*% (s - sigma)
}

# Takes in loadings and the lower triangle of the
# sample polychoric correlations matrix (s)
uls_fit <- function(lambdas, s = s_lower) {
  sigma <- sigma_lower(lambdas)
  t(s - sigma) %*% (s - sigma)
}
tictoc::tic()
optim_dwls <- optim(rep(.5, 5), dwls_fit)
tictoc::toc()
```

```
0.008 sec elapsed
```

```
tictoc::tic()
optim_uls <- optim(rep(.5, 5), uls_fit)
tictoc::toc()
```

```
0.009 sec elapsed
```

```
tictoc::tic()
optim_wls <- optim(rep(.5, 5), wls_fit)
tictoc::toc()
```

```
0.73 sec elapsed
```

We see that the WLS estimator is a lot slower due to the matrix inversion.

We fit the model in `lavaan` by inputting the model syntax, the raw data, and specifying the following: `std.lavaan = TRUE` (to identify the model by standardizing the latent variable), `ordered = TRUE` (as the data are ordinal), `estimator = "DWLS"`, `estimator = "WLS"` or `estimator = "ULS"`, `missing = "listwise"` (the default option; FIML is not available with DWLS, WLS, or ULS). For DWLS, robust standard errors are specified using `se = "robust.sem"` and robust (scaled) test statistic is requested with `test = "scaled.shifted"`.

Note that specifying `estimator = "DWLS", se = "robust.sem", test = "scaled.shifted"` is equivalent to specifying `estimator = "WLSMV"` or `"WLSM"`.

```
cfa_dwls_robust <-
  cfa('sbj_e =~  i1 + i2 + i3 + i4 + i5',
      data = ses,
      std.lv = TRUE,
      ordered = names(ses),
      estimator = "DWLS",
      se = "robust.sem",
      test = "scaled.shifted",
      missing = "listwise"
)

# cfa_dwls_simple <-
#   cfa('sbj_e =~  i1 + i2 + i3 + i4 + i5',
#       data = ses,
#       std.lv = TRUE,
#       ordered = names(ses),
#       estimator = "DWLS",
#       missing = "listwise"
# )

cfa_wls <-
  cfa('sbj_e =~  i1 + i2 + i3 + i4 + i5',
      data = ses,
      std.lv = TRUE,
      ordered = names(ses),
      estimator = "WLS",
      se = "robust.sem"#,
     # missing = "listwise"
)

cfa_uls <-
  cfa('sbj_e =~  i1 + i2 + i3 + i4 + i5',
      data = ses,
      std.lv = TRUE,
      ordered = names(ses),
      estimator = "ULSM",
```

```r
      missing = "listwise"
)
```

Compare loading estimates produced by `lavaan` with the ones we computed:

Table 3: Estimated loadings

	WLS		DWLS		ULS	
	*	lavaan	*	lavaan	*	lavaan
lambda 1	0.8213	0.8215	0.8189	0.8189	0.8182	0.8182
lambda 2	0.8195	0.8195	0.8186	0.8186	0.8186	0.8185
lambda 3	0.8264	0.8263	0.8234	0.8234	0.8228	0.8228
lambda 4	0.8258	0.8257	0.8243	0.8242	0.8242	0.8243
lambda 5	0.8290	0.8290	0.8254	0.8256	0.8250	0.8250

Note.
* denotes estimates obtained via direct computation. Columns labeled 'lavaan'indicate that estimates were obtained from the cfa() function output.

Estimation of standard errors

```r
# First derivatives of the model implied polychoric
# correlations with respect to the estimated loadings
Delta_dwls <-
  numDeriv::jacobian(sigma_lower, optim_dwls$par)
Delta_wls <-
  numDeriv::jacobian(sigma_lower, optim_wls$par)
Delta_uls <-
  numDeriv::jacobian(sigma_lower, optim_uls$par)

w_dwls <- diag(diag(a_cov_mat))
w_wls <- a_cov_mat
```

```r
# Functions to compute the asymptotic covariance
# matrices with DWLS, WLS, ULS estimators
asymptotic_cov_dwls_robust <-
  function(Delta = Delta_dwls, W = a_cov_mat,
           V = w_dwls) {
  solve(t(Delta) %*% solve(V) %*% Delta) %*% t(Delta) %*%
    solve(V) %*% W %*% solve(V) %*% Delta %*%
    solve(t(Delta) %*% solve(V) %*% Delta)
  }
```

```r
asymptotic_cov_dwls_simple <-
  function(Delta = Delta_dwls, V = w_dwls) {
  solve(t(Delta) %*% solve(V) %*% Delta)
  }

asymptotic_cov_wls <-
  function(Delta = Delta_wls, W = a_cov_mat) {
  # V cancels out in equation
  solve(t(Delta) %*% solve(W) %*% Delta)
}

asymptotic_cov_uls <-
  function(Delta = Delta_uls, W = a_cov_mat) {
  solve(t(Delta) %*% Delta) %*% t(Delta) %*% W %*%
    Delta %*% solve(t(Delta) %*% Delta)
  }
```

```r
acov_wls <-
  asymptotic_cov_wls(Delta_wls, W = a_cov_mat)
acov_dwls_robust <-
  asymptotic_cov_dwls_robust(Delta_dwls, W = a_cov_mat,
                             V = w_dwls)
acov_uls_robust <-
  asymptotic_cov_uls(Delta_uls, W = a_cov_mat)
# acov_dwls_simple <-
#   asymptotic_cov_dwls_simple(Delta_dwls, V = w_dwls)

a_se_wls <- sqrt(diag(acov_wls))
a_se_dwls_robust <- sqrt(diag(acov_dwls_robust))
a_se_uls_robust <- sqrt(diag(acov_uls_robust))
# a_se_dwls_simple <- sqrt(diag(acov_dwls_simple))

#estimates from lavaan
lav_wls_se <- sqrt(diag(vcov(cfa_wls)[1:5,1:5]))
lav_dwls_se_r <- sqrt(diag(vcov(cfa_dwls_robust)[1:5,1:5]))
# lav_dwls_se_s <- sqrt(diag(vcov(cfa_dwls_simple)[1:5,1:5]))
lav_uls_se_r <- sqrt(diag(vcov(cfa_uls)[1:5,1:5]))
```

```r
# Obtain the Hessian, matrix containing the second derivatives of
# the discrepancy function with respect to the (free) model parameters
H_uls <- inspect(cfa_uls, "hessian")
H_dwls <- inspect(cfa_dwls_robust, "hessian")
```

```
# Take the inverse of the Hessian
H_uls.inv <- try(chol2inv(chol(H_uls)), TRUE)
H_dwls.inv <- try(chol2inv(chol(H_dwls)), TRUE)

# Obtain the (inverse) of the asymptotic variance matrix of the sample
# statistics (given by wls.v)
# https://groups.google.com/g/lavaan/c/Rkwq10jV8JU.
W_uls <- inspect(cfa_uls, "wls.v")  # we know this is a 25x25 identity matrix
W_dwls <- inspect(cfa_dwls_robust, "wls.v")

# Obtain the asymptotic 4th moment, N times the asymptotic variance matrix
# of the sample statistics. Alias: "sampstat.nacov".
Gamma <- inspect(cfa_dwls_robust, "gamma") #same for uls and dwls

# Scaling factors
Delta_uls_new <- inspect(cfa_uls, "delta")
Delta_dwls_new <- inspect(cfa_dwls_robust, "delta")

# derivative of the discrepancy functions w.r.t. s and theta
K_uls <- t(Delta_uls_new) # lai and simoes eq (29)
K_dwls <- t(Delta_dwls_new) %*% diag(1/diag(Gamma)) # lai and simoes eq (37)

# N times asymptotic covariance matrix of the parameter estimates
Pi_uls <- - H_uls.inv %*% K_uls %*% Gamma %*% t(-H_uls.inv %*% K_uls)
Pi_dwls <- - H_dwls.inv %*% K_dwls %*% Gamma %*% t(-H_dwls.inv %*% K_dwls)

n <- inspect(cfa_uls, "nobs") #number of observations

# compute the standard errors of the parameter estimates
SE_new_uls <-  sqrt(diag(Pi_uls)/n)[1:5]
SE_new_dwls <-  sqrt(diag(Pi_dwls)/n)[1:5]

# for compaison with lavaan output
round(lav_dwls_se_r, 6)
```

```
sbj_e=~i1 sbj_e=~i2 sbj_e=~i3 sbj_e=~i4 sbj_e=~i5
 0.014207  0.015200  0.014675  0.015414  0.014650
```

```
round(SE_new_dwls, 6)
```

```
[1] 0.014205 0.015202 0.014667 0.015420 0.014651
```

```
round(lav_uls_se_r, 6)
```

```
sbj_e=~i1 sbj_e=~i2 sbj_e=~i3 sbj_e=~i4 sbj_e=~i5
```

```
0.014246  0.015232  0.014710  0.015450  0.014733
```

```r
round(SE_new_uls, 6)
```

```
[1] 0.014251 0.015239 0.014710 0.015459 0.014740
```

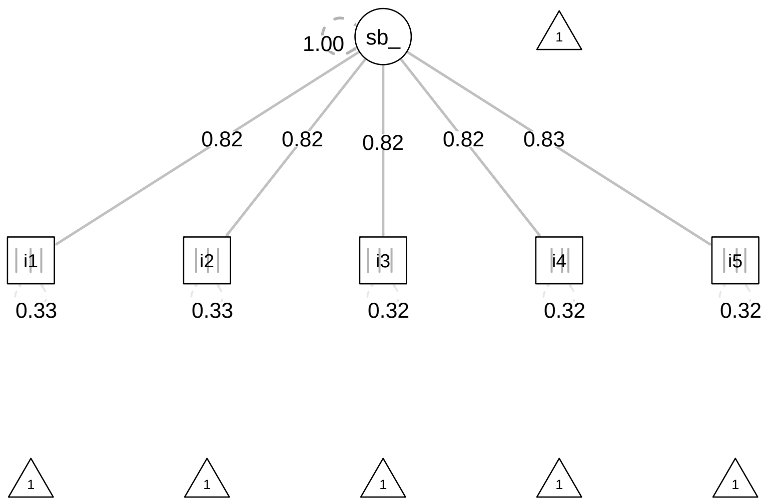
```r
# Note: Lai and Simoes provide a function for this method as part of
# their suplemental materials at # https://bit.ly/3sOuLfR
```

% latex table generated in R 4.2.2 by xtable 1.8-4 package % Wed May 3 17:14:18 2023

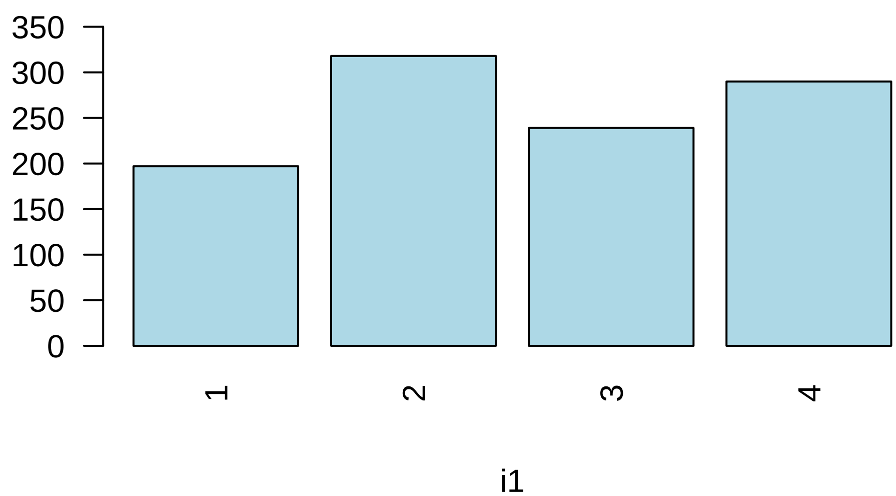
			new	robust	robust	new		
SE(lambda 1)	0.014122	0.014123	0.014205	0.014208	0.014207	0.014251	0.014246	0.014246
SE(lambda 2)	0.015139	0.015140	0.015202	0.015200	0.015200	0.015239	0.015232	0.015232
SE(lambda 3)	0.014556	0.014556	0.014667	0.014675	0.014675	0.014710	0.014710	0.014710
SE(lambda 4)	0.015326	0.015326	0.015420	0.015415	0.015414	0.015459	0.015450	0.015450
SE(lambda 5)	0.014497	0.014496	0.014651	0.014650	0.014650	0.014740	0.014732	0.014733

Table 4: Factor loading estimate and SEs with WLS, DWLS, ULS estimation

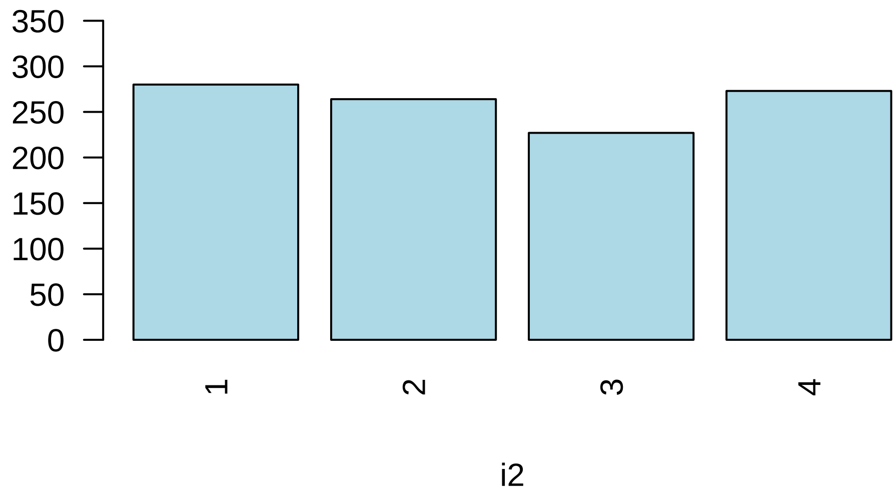
	WLS		DWLS		ULS	
	Est.	S.E.	Est.	S.E.	Est.	S
empty	0.8215 [0.7938, 0.8492]	0.0141	0.8189 [0.7911, 0.8468]	0.0142	0.8182 [0.7903, 0.8461]	0.
absent	0.8195 [0.7898, 0.8492]	0.0151	0.8186 [0.7888, 0.8484]	0.0152	0.8185 [0.7886, 0.8484]	0.
unfulfilled	0.8263 [0.7978, 0.8548]	0.0146	0.8234 [0.7946, 0.8521]	0.0147	0.8228 [0.7940, 0.8516]	0.
exist	0.8257 [0.7956, 0.8557]	0.0153	0.8242 [0.7940, 0.8544]	0.0154	0.8243 [0.7940, 0.8546]	0.
alone	0.8290 [0.8006, 0.8574]	0.0145	0.8256 [0.7969, 0.8543]	0.0146	0.8250 [0.7961, 0.8538]	0.
Num.Obs.	1044		1044		1044	



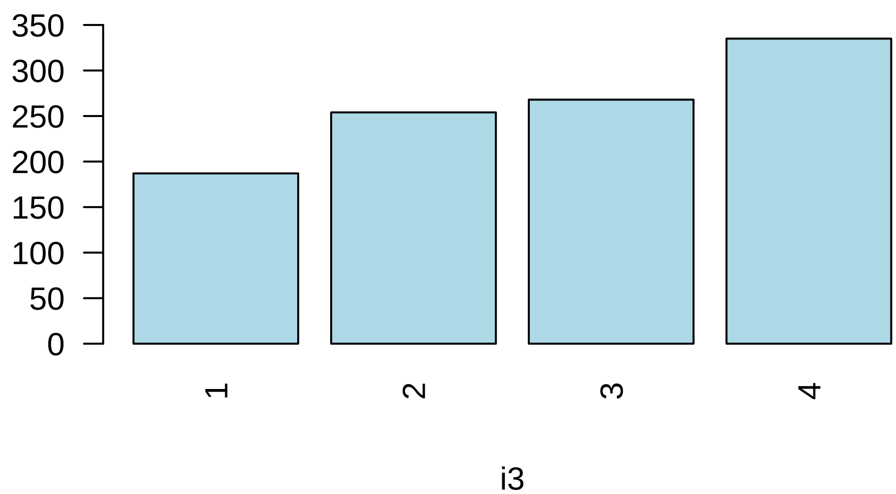

```
par( las = 2)

barplot(table(ses$i1), xlab = "i1", main = "", col = "lightblue", ylim = c(0,350))
```



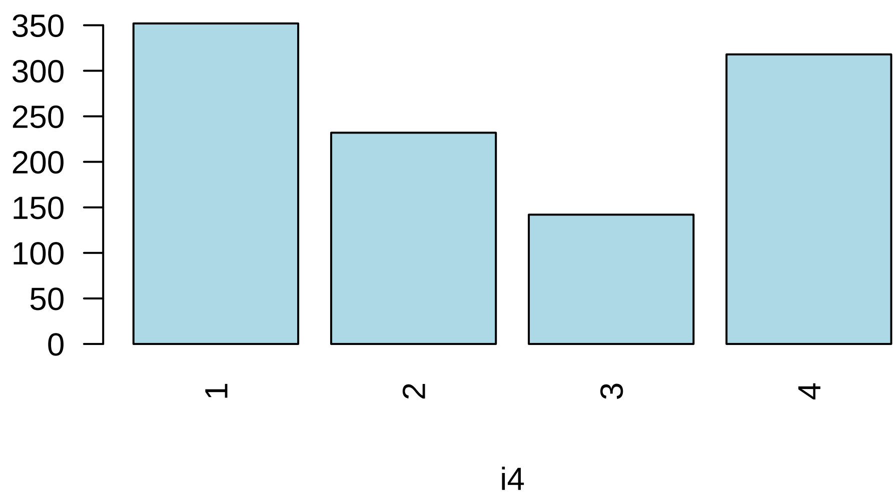

```
barplot(table(ses$i2), xlab = "i2", main = "", col = "lightblue", ylim = c(0,350))
```



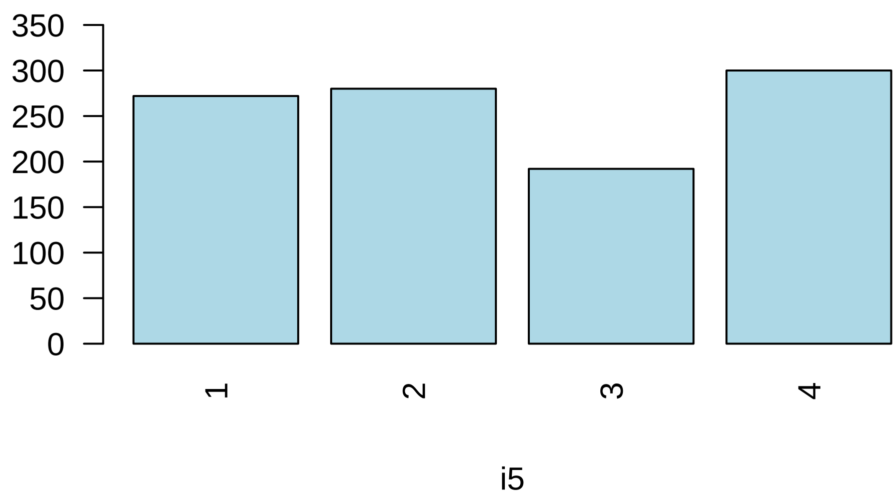

```
barplot(table(ses$i3), xlab = "i3", main = "", col = "lightblue", ylim = c(0,350))
```




```
barplot(table(ses$i4), xlab = "i4", main = "", col = "lightblue", ylim = c(0,350))
```




```r
barplot(table(ses$i5), xlab = "i5", main = "", col = "lightblue", ylim = c(0,350))
```



References

Lai, K., & Simoes, A. (2023). Reflecting on the “robust” standard errors for two-stage sem estimation with categorical data: Mistakes and correction. Structural Equation Modeling: A Multidisciplinary Journal, 1–17.

Olsson, U. (1979). Maximum likelihood estimation of the polychoric correlation coefficient. Psychometrika, 44(4), 443–460.

https://quantscience.rbind.io/2020/06/12/weighted-least-squares/#polychoric-correlations

Flora, D.B, & Curran, P.J. An empirical evaluation of alternative methods of estimation for confirmatory factor analysis with ordinal data. *Psychological methods* vol. 9,4 (2004): 466-91. doi:10.1037/1082-989X.9.4.466.

Liddell, T., & Kruschke, J. K. (2017). Analyzing ordinal data with metric models: What could possibly go wrong?. https://doi.org/10.31219/osf.io/9h3et.

Li C. H. (2016). Confirmatory factor analysis with ordinal data: Comparing robust maximum likelihood and diagonally weighted least squares. *Behavior research methods*, *48*(3), 936–949. https://doi.org/10.3758/s13428-015-0619-7.

R Core Team (2022). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Savalei, V., & Rosseel, Y. (2022). Computational options for standard errors and test statistics with incomplete normal and nonnormal data in sem. Structural Equation Modeling: A Multidisciplinary Journal, 29(2).

Stevens, S. S. (1946). On the Theory of Scales of Measurement. Science, 103, 677-80. doi:10.1126/science.103.2684.677.

Yang-Wallentin, F., Jöreskog, K.G. & Luo, H. (2010). Confirmatory Factor Analysis of Ordinal Variables With Misspecified Models', Structural Equation Modeling: A Multidisciplinary Journal, 17(3), 392-423. 10.1080/10705511.2010.489003.

Wu, H., Estabrook, R. (2016). Identification of Confirmatory Factor Analysis Models of Different Levels of Invariance for Ordered Categorical Outcomes. *Psychometrika* **81**, 1014–1045. https://doi.org/10.1007/s11336-016-9506-0.