Displays for Statistics 5401/8401

Lecture 30
(corrected)

November 16, 2005

Christopher Bingham, Instructor

612-625-1024, kb@umn.edu
372 Ford Hall

Class Web Page

http://www.stat.umn.edu/~kb/classes/5401

On Monday I outlined the *iterated principal factor* (IPF) *method*, an iterative method to estimate $\mathbf{L}$ and $\mathbf{\Psi}$.

It can be shown that each step decreases $\|\mathbf{S} - \hat{\mathbf{\Sigma}}\|^2 = \text{tr}(\mathbf{S} - \hat{\mathbf{\Sigma}})^2 = \sum_j \sum_k (s_{jk} - \hat{\sigma}_{jk})^2$, $\hat{\mathbf{\Sigma}} = \hat{\mathbf{L}}\hat{\mathbf{L}}' + \hat{\mathbf{\Psi}}$. Because the diagonal is fit exactly, $\|\mathbf{S} - \hat{\mathbf{\Sigma}}\|^2 = \sum\sum_{j \neq k} (s_{jk} - \hat{\sigma}_{jk})^2$.

Example: Artificial data, n = 100, p = 5.

```
Cmd> r <- cor(y); r
           Y1         Y2         Y3         Y4         Y5
Y1          1     0.3219   -0.45859   -0.40979    0.42777
Y2     0.3219          1   -0.37866   -0.37583    0.26888
Y3   -0.45859   -0.37866          1    0.72635   -0.72049
Y4   -0.40979   -0.37583    0.72635          1   -0.69711
Y5    0.42777    0.26888   -0.72049   -0.69711          1

Cmd> psi0 <- 1/diag(solve(r)); psi0
(1)    0.74263    0.80596    0.36209    0.39667    0.40739
```

psi0 contains starting values, $\hat{\psi}_{k0} = 1/r^{kk}$, where $\mathbf{R}^{-1} = [r^{k\ell}]$. $\hat{\psi}_{k0}$ is $1 - R_k^2$, where R_k^2 is multiple R^2 in the LS regression of x_k on the other x's, $x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_p$.

```
Cmd> regs(y[,-1],y[,1]) # usage: regs(x,y), regress y on x
Model used is @Y=@X1+@X2+@X3+@X4
                Coef      StdErr           t
CONSTANT        60.2      44.347      1.3575
X1           0.25308     0.14708      1.7207
X2          -0.14171      0.0932     -1.5205
X3          -0.12404     0.29143    -0.42562
X4           0.34799     0.26208      1.3278

N: 100, MSE: 246.87, DF: 95, R^2: 0.25737
Regression F(4,95): 8.2309, Durbin-Watson: 1.7376
To see the ANOVA table type 'anova()'

Cmd> vector(psi0[1], 1 - 0.25737)
(1)     0.74263     0.74263     psi0[1] and 1 - R^2
```

Start the iteration. Compute $V^* = R - \hat{\Psi}_0$

```
Cmd> m <- 2 # assuming two factors

Cmd> vstar <- r - dmat(psi0)

Cmd> eigs <- eigen(vstar); eigs$values
(1)      2.5061     0.10703   -0.048556      -0.113    -0.16632
```

Compute $\hat{L} = [\hat{\ell}_1, \hat{\ell}_2] = [\sqrt{\delta_1}\mathbf{e}_1, \sqrt{\delta_2}\mathbf{e}_2]$, where δ_i and $\mathbf{e}_i$ are eigenvalues and eigenvectors of $V^* = R - \Psi_0$.

```
Cmd> Lhat <- eigs$vectors[,run(m)] * sqrt(eigs$values[run(m)])'

Cmd> Lhat # unrotated loading matrix; Lhat' %*% Lhat is diagonal
            (1)       (2)
 Y1     0.53669     0.15806
 Y2     0.43935     0.23599
 Y3      -0.847    0.032682
 Y4    -0.81832    0.048827
 Y5     0.79876    -0.15133
```

$\hat{L}$ satisfies $\hat{L}'\hat{L}$ is diagonal

```
Cmd> Lhat ' %*% Lhat # diag elements are eigenvalues
              (1)           (2)
(1)        2.5061 -9.5373e-17     Diagonal matches eigs$values
(2) -9.5373e-17        0.10703
```


Update $\hat{\Psi}$ so diagonal of $\hat{\rho} = \hat{L}\hat{L}' + \hat{\Psi}$ matches the diagonal of R (all 1's for correlation)

```
Cmd> Vhat <- Lhat %*% Lhat' # estimated rank 2 part

Cmd> psihat <- diag(r - Vhat); psihat
(1)     0.68698     0.75128     0.28153     0.32797     0.33909

Cmd> rhohat <- Vhat + dmat(psihat); rhohat
             Y1          Y2          Y3          Y4          Y5
Y1            1     0.27309    -0.44941    -0.43146     0.40476
Y2      0.27309           1    -0.36441      -0.348     0.31522
Y3     -0.44941    -0.36441           1     0.69471    -0.68149
Y4     -0.43146      -0.348     0.69471           1    -0.66102
Y5      0.40476     0.31522    -0.68149    -0.66102           1

Cmd> sum(vector(r - rhohat)^2) # SS residuals
(1)    0.020829
```

This is $\mathrm{tr}(R - \hat{\rho})^2$, $\hat{\rho} = \hat{L}\hat{L}' + \hat{\Psi}$.

Start next iteration:

```
Cmd> eigs <- eigen(r - dmat(psihat)); eigs$values
(1)      2.5756     0.16543    0.013377   -0.036741    -0.10448

Cmd> Lhat <- eigs$vectors[,run(m)] * sqrt(eigs$values[run(m)])'

Cmd> Lhat
            (1)         (2)
 Y1     0.54104     0.19693
 Y2     0.44269     0.29218
 Y3    -0.86234    0.038787
 Y4    -0.82939    0.057781
 Y5     0.80953    -0.19088
```

Updated unrotated loading matrix

```
Cmd> Vhat <- Lhat %*% Lhat'

Cmd> psihat <- diag(r - Vhat); psihat # updated uniquenesses
(1)     0.66849     0.71865     0.25487     0.30877     0.30823
```

```
Cmd> rhohat <- Vhat + dmat(psihat); rhohat
            Y1          Y2          Y3          Y4          Y5
Y1           1     0.29706    -0.45892    -0.43736      0.4004
Y2     0.29706           1    -0.37042    -0.35028      0.3026
Y3    -0.45892    -0.37042           1     0.71746    -0.70549
Y4    -0.43736    -0.35028     0.71746           1    -0.68245
Y5      0.4004      0.3026    -0.70549    -0.68245           1

Cmd> sum(vector(r - rhohat)^2) # SS residuals
(1)    0.0090067

Cmd> for(i,1,11){ # compute SS residuals for 11 more steps
 eigs <- eigen(r - dmat(psihat))
 Lhat <- eigs$vectors[,run(m)] * sqrt(eigs$values[run(m)])'
 Vhat <- Lhat %*% Lhat'
 psihat <- diag(r - Vhat)
 rhohat <- Vhat + dmat(psihat)
 print(paste("SS residuals =",sum(vector(r - rhohat)^2)))}
SS residuals = 0.0065539    Change = -0.0024528 on step 3
SS residuals = 0.0055527    Change = -0.0010012 on step 4
SS residuals = 0.0048955    Change = -0.0006572 on step 5
SS residuals = 0.0044088    Change = -0.0004867 on step 6
SS residuals = 0.0040434    Change = -0.0003654 on step 7
SS residuals = 0.0037671    Change = -0.0002763 on step 8
SS residuals = 0.0035549    Change = -0.0002122 on step 9
SS residuals = 0.0033884    Change = -0.0001665 on step 10
SS residuals = 0.0032545    Change = -0.0001339 on step 11
SS residuals = 0.0031442    Change = -0.0001103 on step 12
SS residuals = 0.003051     Change = -0.0000932 on step 13
```

Residual SS is always decreasing but rather slowly.

```
Cmd> print(psihat, Lhat)
psihat:Estimated uniquenesses after 13 steps
(1)     0.69117     0.59215     0.24966     0.31583     0.24864
Lhat:   Estimated unrotated loadings after 13 steps
            (1)         (2)
Y1     0.53605     0.14656
Y2     0.46525     0.43748
Y3    -0.86508    0.044508
Y4    -0.82601    0.043232
Y5     0.82982    -0.25052
```

All $\psi_j \geq 0$, $j = 1, \ldots, p$ when I redid it.

Macro **stepuls()** on the new version of **mulvar.mac.txt** does IPF automatically once you have a starting **psi0**. Let's run it for 100 steps,

```
Cmd> addmacrofile("") # find new mulvar.mac.txt

Cmd> stepuls(r,psi0,m,nsteps:100)
component: psi
          Y1          Y2          Y3          Y4          Y5
     0.71102     0.38008     0.24482     0.31463     0.27222
component: loadings
             F1          F2
Y1     0.53159     0.07992
Y2     0.50944     0.60033
Y3     -0.8637    0.095944
Y4    -0.82381     0.08194
Y5     0.81807    -0.24197
component: crit
(1)   0.0017449
```

The residual SS (**crit**) is down by almost 50% from 13 steps.

One criterion for convergence is that the relative change in **crit** is small.

This iterates until the relative change is smaller than .01%

```
Cmd> psihat <- structure(psi:psi0,crit:1) # starting value

Cmd> for(i,1,1000){
        lastpsihat <- psihat
        psihat <- stepuls(r,psihat,m)
        if (1 - psihat$crit/lastpsihat$crit < 1e-4) { break}
     }
ERROR: min(psi) <= 0 on step 1 in macro stepuls()

Cmd> i # one psihat went negative on step 821
(1)         821
```

It didn't converge before an improper $\hat{\Psi}$ was reached.

```
Cmd> psihat
component: psi
          Y1           Y2          Y3          Y4          Y5
     0.72313  0.00019846     0.24031     0.31388     0.28806
component: loadings
            F1          F2
Y1     0.52617   0.0031454
Y2      0.6048     0.79625
Y3     -0.8547     0.17082
Y4    -0.81458     0.15022
Y5     0.79986    -0.26862
component: crit
(1)   0.0012582
```

It looks like $\hat{\psi}_2 \approx 0$, the Heywood case. **crit** has been substantially reduced.

stepuls() creates "side-effect" variables:

```
Cmd> PSI
          Y1           Y2          Y3          Y4          Y5
     0.72313  0.00019846     0.24031     0.31388     0.28806

Cmd> LOADINGS
            F1          F2
Y1     0.52617   0.0031454
Y2      0.6048     0.79625
Y3     -0.8547     0.17082
Y4    -0.81458     0.15022
Y5     0.79986    -0.26862

Cmd> CRITERION
(1)   0.0012582
```


Factors have been extracted; it's time to rotate $\hat{L}$ using **rotation()**.

```
Cmd> rotation(Lhat,method:"varimax",kaiser:T)
            (1)         (2)
Y1     0.47578     0.22473  Estimated rotated loadings
Y2     0.21264     0.97703  Not very simple
Y3    -0.84703    -0.20551
Y4    -0.80197    -0.20728
Y5     0.83854    0.093713
```

I used Kaiser normalization (**kaiser:T**) as part of the rotation process. This divides row k of $\hat{L}$ by $c_k = \sqrt{\{\sum_{1 \le j \le m} \hat{\ell}_{kj}^2\}}$ before rotation and then multiplies each row of the rotated matrix by c_k. Here I do the Kaiser normalization "by hand", using **rotation()** without **kaiser:T**:

```
Cmd> c <- vector(sqrt(sum(Lhat'^2)))  #sqrt(row SS of Lhat)
Cmd> c * rotation(Lhat/c,method:"varimax")
            (1)         (2)
Y1     0.47578     0.22473  Same as above
Y2     0.21264     0.97703
Y3    -0.84703    -0.20551
Y4    -0.80197    -0.20728
Y5     0.83854    0.093713
```

MacAnova: **c** is column vector so that row k of **Lhat/c** is row k of **Lhat** divided by **c[k]** and row k of the result is row k of the rotated **Lhat/c** multiplied by **c[k]**.

```
Cmd> rotation(Lhat, method:"quartimax",kaiser:T)
          (1)         (2)
Y1     0.50942     0.13174  Quartimax rotated loadings
Y2     0.39168     0.91999  These appear somewhat simpler
Y3    -0.87052   -0.043412  Y3, Y4, Y5 load heavily on
Y4    -0.82659   -0.053572  factor 1, but not on factor 2
Y5     0.84127   -0.064828  Y1 loads mainly on factor 1
```

Without knowing something about what the actual variables are, it is impossible to interpret the factors.

The ***Unweighted Least Squares*** (ULS) estimates are values of $\hat{\mathbf{V}}$ and $\hat{\mathbf{\Psi}}$ which minimize $\|\mathbf{S} - \hat{\mathbf{\Sigma}}\|^2$ or $\|\mathbf{R} - \hat{\rho}\|^2$

That is, the ULS estimates $\hat{\mathbf{L}}_{ULS}$, $\hat{\mathbf{\Psi}}_{ULS}$ make $\|\mathbf{S} - (\mathbf{LL}' + \mathbf{\Psi})\|^2$ or $\|\mathbf{R} - (\mathbf{LL}' + \mathbf{\Psi})\|^2$ as small as possible.

Iterated principal factor estimation is a *particular algorithm* that tries to do this. However, for serious estimation, one should use a better algorithm such as is provided by **facanal()** that converges more rapidly and reliably.

```
Cmd> result <- facanal(r, m, method:"uls",rotate:"quartimax")
WARNING: no convergence in 30 iterations
estimated uniquenesses:
          Y1          Y2          Y3          Y4          Y5
     0.72309  8.3218e-05     0.24038     0.31384     0.28805
quartimax rotated estimated loadings:
      Factor 1    Factor 2
Y1     0.50943     0.13172
Y2     0.39167     0.92022
Y3     -0.8705    -0.04342
Y4     -0.8266   -0.053566
Y5     0.84127   -0.064805
minimized uls criterion:
(1)    0.000629
```

I specified ULS extraction followed by quartimax rotation.

The **WARNING** message tells you this did not converge in the default number of steps (30). You can increase the allowed number of steps by keyword **maxit**.

```
Cmd> result <- facanal(r, m,method:"uls",rotate:"quartimax",\
    maxit:100) # allow up to 100 iterations
Convergence in 33 iterations by criterion 2
estimated uniquenesses:
          Y1          Y2          Y3          Y4          Y5
     0.72313  7.0008e-06      0.2403     0.31389     0.28808
varimax rotated estimated loadings:
      Factor 1    Factor 2
Y1     0.50942     0.13171
Y2     0.39167     0.92026
Y3    -0.87053   -0.043408
Y4    -0.82659   -0.053571
Y5     0.84126   -0.064792
minimized uls criterion:
(1)  0.00062897
```

The solution is essentially the same as we got using 830 steps of **stepuls()**.

The criterion is actually $\|\mathbf{R} - \hat{\boldsymbol{\rho}}\|^2/2$, a sum of squares of above the diagonal of $\mathbf{R} - \hat{\boldsymbol{\rho}}$. You multiply by 2 to compare it with the IPF solution:

```
Cmd> 2*CRITERION
(1)   0.0012579   Slightly less than 0.0012582 from stepuls
```


facanal() returns a structure with lots of information:

```
Cmd> result # created by facanal()
component: psihat
          Y1          Y2          Y3          Y4          Y5
     0.72313  7.0008e-06      0.2403     0.31389     0.28808
component: loadings      This was quartimax rotated
      Factor 1    Factor 2
Y1     0.50942     0.13171
Y2     0.39167     0.92026
Y3    -0.87053   -0.043408
Y4    -0.82659   -0.053571
Y5     0.84126   -0.064792
component: criterion
(1)  0.00062897          .5*minimized residual SS
component: eigenvals
(1)      2.6765     0.75834    0.019211   0.0089472    -0.02844
component: gradient      Should and is be close to 0
(1) -5.5026e-06  2.0504e-09 -8.4716e-07  3.4599e-07 -4.8732e-07
component: method
(1) "uls"            Estimation method
component: rotation
(1) "quartimax"      Rotation method
component: iter
(1)          33      Number of iterations
component: status
(1)           2      Converged when criterion change < 1e-8
```

result$eigenvals contains eigenvalues δ_1, ..., δ_p of $\hat{\mathbf{V}} = \mathbf{S} - \hat{\boldsymbol{\Psi}}$. δ_i should be close to 0 for $i > m$.

result$criterion is $\sum_{m+1 \le i \le p} [\delta(\hat{\boldsymbol{\Psi}})]^2/2 = \sum\sum (r_{ij} - \hat{\rho}_{ij})^2/2$, where $\hat{\boldsymbol{\rho}} = \hat{\mathbf{L}}\hat{\mathbf{L}}' + \hat{\boldsymbol{\Psi}}$.

```
Cmd> sum(result$eigenvals[-run(m)]^2)/2
(1)   0.00062897
```

`facanal()` creates "side effect" variables `LOADINGS`, `PSI` and `CRITERION` which are the same as `result$loadings`, `result$psihat` and `result$criterion`.

```
Cmd> PSI
         Y1           Y2          Y3          Y4          Y5
    0.72313  7.0008e-06      0.2403     0.31389     0.28808

Cmd> rhohat <- LOADINGS %*% LOADINGS' + dmat(PSI)

Cmd> rhohat # fitted correlation matrix
            Y1          Y2          Y3          Y4          Y5
Y1     0.99999     0.32074    -0.44918    -0.42814     0.42002
Y2     0.32074      1.0003     -0.3809    -0.37305     0.26987
Y3    -0.44918     -0.3809           1     0.72189    -0.72953
Y4    -0.42814    -0.37305     0.72189           1     -0.6919
Y5     0.42002     0.26987    -0.72953     -0.6919           1

Cmd> sum(vector(r - rhohat)^2)/2
(1)    0.00062897        0.5 Residual SS

Cmd> CRITERION
(1)    0.00062897
```

The estimated model is clearly a Heywood case:

y_2 is essentially completely dependent on f_1 and f_2

GLS method (Generalized least squares):

Find $\hat{\mathbf{L}}$ and $\hat{\boldsymbol{\Psi}}$ to minimize

$$G(\mathbf{S},\hat{\boldsymbol{\Sigma}}) \equiv \mathrm{tr}(\mathbf{S}^{-1}(\mathbf{S} - \hat{\boldsymbol{\Sigma}}))^2/2 = \mathrm{tr}(\mathbf{I}_p - \mathbf{S}^{-1}\hat{\boldsymbol{\Sigma}})^2/2$$

$$\hat{\boldsymbol{\Sigma}} = \hat{\mathbf{L}}\hat{\mathbf{L}}' + \hat{\boldsymbol{\Psi}}$$

or

$$G(\mathbf{R},\hat{\boldsymbol{\Sigma}}) \equiv \mathrm{tr}(\mathbf{R}^{-1}(\mathbf{R} - \hat{\boldsymbol{\rho}}))^2/2 = \mathrm{tr}(\mathbf{I}_p - \mathbf{R}^{-1}\hat{\boldsymbol{\rho}})^2/2$$

$$\hat{\boldsymbol{\rho}} = \hat{\mathbf{L}}\hat{\mathbf{L}}' + \hat{\boldsymbol{\Psi}}$$

Note that $G(\mathbf{S},\hat{\boldsymbol{\Sigma}}) = 0$ when $\hat{\boldsymbol{\Sigma}} = \mathbf{S}$

ML method (Maximum likelihood):

Find $\hat{\mathbf{L}}$ and $\hat{\boldsymbol{\Psi}}$ to minimize

$$M(\mathbf{S},\hat{\boldsymbol{\Sigma}}) \equiv \mathrm{tr}(\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{S} - \mathbf{I}_p) - \log(\det[\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{S}]) \geq 0$$

or

$$M(\mathbf{R},\hat{\boldsymbol{\rho}}) \equiv \mathrm{tr}(\hat{\boldsymbol{\rho}}^{-1}\mathbf{R} - \mathbf{I}_p) - \log(\det[\hat{\boldsymbol{\rho}}^{-1}\mathbf{R}]) \geq 0$$

The ML method maximizes the likelihood based on an assumed Wishart distribution for $\mathbf{S}$.

Note that $M(\mathbf{S},\hat{\boldsymbol{\Sigma}}) = 0$ only $\hat{\boldsymbol{\Sigma}} = \mathbf{S}$ and $M(\mathbf{R},\hat{\boldsymbol{\rho}}) = 0$ only when $\hat{\boldsymbol{\rho}} = \mathbf{R}$ so minimizing it may make sense without normality.

Notation

- $\gamma_i = \gamma_i(\Psi)$: eigenvalues of S relative to Ψ, $i = 1,\ldots,p$
- $u_i = u_i(\Psi)$: corresponding eigenvectors

Fact: You can minimize either criterion by minimizing a function of Ψ alone:

GLS: $G(\Psi) \equiv \sum_{m+1 \le j \le p}((\gamma_i - 1)/\gamma_i)^2/2$

ML: $M(\Psi) \equiv \sum_{m+1 \le j \le p}(\gamma_i - 1 - \log(\gamma_i))$

$G(\Psi) = 0$ or $M(\Psi) = 0$ if and only if the "trailing" eigenvalues $\gamma_{m+1},\ldots,\gamma_p$ are all 1.

GLS or ML estimation tries to bring these as close to 1 as possible.

Similarly for ULS estimation, the residual SS depends only on Ψ since

$$\mathrm{tr}(S - \hat{\Sigma})^2/2 = \sum_{m+1 \le j \le p}\delta_j^2/2$$

where $\delta_j = \delta_j(\hat{\Psi})$ are the eigenvalues of $\hat{V} = S - \hat{\Psi}$ which depends only on $\hat{\Psi}$.

`facanal()` uses numerical minimization of the criteria as functions of Ψ.

The estimate of the rank m component V is

$$\hat{V} = \hat{L}\hat{L}',$$

where

$$\hat{L} = \hat{\Psi} \times [\sqrt{\{\gamma_1 - 1\}} \times u_1,\ldots,\sqrt{\{\gamma_m - 1\}} \times u_m]$$

is an unrotated loading matrix. This is the only place where the relative eigenvectors u_j of S relative to $\hat{\Psi}$ come into play.

$\hat{L}$ satisfies the mathematical restriction:

$\hat{L}'\hat{\Psi}^{-1}\hat{L} = \mathrm{diag}[\gamma_1 - 1,\ldots,\gamma_m - 1]$ is diagonal

For both GLS and ML estimation, there are relative slow iterative methods similar to IPF. These are implemented in `stepgls()` and `stepml()` which are used essentially identically to `stepuls()`.

Greatly to be preferred is `facanal()` with `method:"gls"` or `method:"ml"`.

GLS Estimates

```
Cmd> result_gls <- facanal(r,m,method:"gls",\
    rotation:"quartimax")
Convergence in 26 iterations by criterion 2
estimated uniquenesses:
          Y1          Y2          Y3          Y4          Y5
     0.71672  1.6257e-06     0.24535     0.30095     0.29349
quartimax rotated estimated loadings:
      Factor 1    Factor 2
Y1     0.51074     0.13258
Y2      0.3914     0.92022
Y3    -0.86744   -0.042541
Y4    -0.83343    -0.05393
Y5     0.83788   -0.064179
minimized gls criterion:
(1)   0.0036705
```

This is a similar fit to GLS, again apparently a Heywood case with $\hat{\psi}_2 \approx 0$.

MLE Estimates

```
Cmd> result_mle <- facanal(r,2,method:"mle",\
    rotation:"quartimax")
Convergence in 26 iterations by criterion 2
estimated uniquenesses:
          Y1          Y2          Y3          Y4          Y5
     0.72178  1.5031e-06      0.2457     0.30368     0.29364
quartimax rotated estimated loadings:
      Factor 1    Factor 2
Y1     0.51052      0.1326
Y2     0.39154     0.92016
Y3    -0.86747   -0.042402
Y4     -0.8327   -0.054116
Y5     0.83799   -0.064357
minimized mle criterion:
(1)   0.0035949
```

This is almost the same as the GLS solution.

Silently redo MLE without rotation

```
Cmd> unrotated <- facanal(r,2,method:"mle",\
       rotation:"none",quiet:T)#find PSI, LOADINGS

Cmd> LOADINGS' %*% dmat(1/PSI) %*% LOADINGS
            Factor 1    Factor 2
Factor 1  6.6528e+05 -1.8829e-13   Diagonal matrix
Factor 2 -1.8863e-13      6.7155   from unrotated Lhat
```

This confirms that $\hat{L}'\hat{\Psi}^{-1}\hat{L}$ is diagonal.

This does not hold for rotated $\hat{L}$:

```
Cmd> result_mle$loadings' %*% dmat(1/result_mle$psihat) %*% \
       result_mle$loadings
            Factor 1    Factor 2
Factor 1    1.02e+05  2.3969e+05   Non-diagonal matrix
Factor 2  2.3969e+05  5.6329e+05   from rotated Lhat
```

Goodness of Fit Tests

GLS and ML have **goodness-of-fit** test statistics of the same form:

$$\{f_e - (2p+5)/6 - 2m/3\}C(\hat{\Psi}),$$

where $C(\Psi) = G(\Psi)$ or $C(\Psi) = M(\Psi)$.

When Σ has factor analytic form, in large samples when $\mathbf{x}$ is $N_p(\mu,\Sigma)$, both are close to χ^2_f, where $f = \{(p-m)^2 - p - m\}/2$
When $f \leq 0$, no test is possible and GLS or ML estimation may not work.

```
Cmd> p <- ncols(y)
Cmd> N <- nrows(y);fe <- N - 1
Cmd> constant <- fe - (2*p+5)/6 - 2*m/3; constant
(1)      95.167
Cmd> test_gls <- constant*result_gls$criterion; test_gls
(1)     0.34931
Cmd> test_mle <- constant*result_mle$criterion; test_mle
(1)     0.34211
Cmd> f <- {(p-m)^2 - p - m}/2; f # degrees of freedom
(1)           1
Cmd> cumchi(vector(test_gls,test_mle),f,upper:T)
(1)     0.55451     0.55861     P-values
```

Both P-values > .50, not even close to significant.

Using either statistic, there is no evidence against the null hypothesis

H_0: Σ (or ρ) has factor analytic form with m = 2 factors.

Caution

Like almost all tests about variance matrices or correlation matrices, this one is non-robust to non-normality. You should view conclusions skeptically.

What next?

After rotation comes "identification" or "naming" of factors. You use knowledge of the subject matter field, say, psychology, and the pattern of near 0's in $\hat{L}$ to identify each f_j with a concept. This is usually done by finding aspects in common among variables that load heavily on each factor.

Estimating Factor Scores

Factor scores f_i are not directly *observable*, but *can* be estimated.

You can use factor scores as variables that *summarize the information* in $\mathbf{x}$. This is a form of *dimension reduction*.

- The results of testing instruments often have several "*scales*", designed to measure conceptually different things. Such scales are often estimates of factor scores, usually normalized so as to have specific population means and standard deviations.
- You can sometimes use factor scores as data in further analysis, with the estimated score vector $\hat{\mathbf{f}}$ replacing $\mathbf{x}$, either as a dependent variable or as an explanatory variable.
- Scores can also be used to identify exceptional cases -- *score outliers*.