# Statistics 5401
# 23. Factor Rotation

Gary W. Oehlert
School of Statistics
313B Ford Hall
612-625-1557
gary@stat.umn.edu

Let $x$ be a p-dimensional random variable with mean $\mu$ and variance $\Sigma$. Recall the orthogonal factor model:

$$\begin{aligned} x &= \mu + \mathbf{L}f + \epsilon \\ \Sigma &= \mathbf{V} + \Psi \\ &= \mathbf{L}\mathbf{L}' + \Psi \end{aligned}$$

where $\mathbf{V}$ has rank $m < p$, $\Psi$ is diagonal with nonnegative diagonal elements, and $\mathbf{L}$ is $p \times m$.
Factor extraction finds estimates of $\mathbf{V}$ and $\Psi$, along with a preliminary estimate of $\mathbf{L}$.
If $\mathbf{H}$ is an $m \times m$ orthogonal matrix, then

$$\mathbf{L}^\star = \mathbf{L}\mathbf{H}$$

is just as good as the original $\mathbf{L}$ for specifying the covariance between components of $x$.
Why might we prefer one $\mathbf{L}$ over another?
We are looking for simplicity of interpretation.
$\mathbf{L}$ tells us how the components of $x$ depend on the latent factors. We like an $\mathbf{L}$ that leads to easily interpretable factors.
What makes $\mathbf{L}$ easy to interpret? Basically, lots of zeroes.
If $\mathbf{L}$ has lots of zeroes, then each factor affects relatively few components of $x$, or, alternatively, each component of $x$ is affected by relatively few factors.
Well, actually we can almost never hope for real zeroes, the best we can usually get is small numbers.
For example, consider the O2 data. The ML loadings for two factors were extracted as.

```
Cmd> R <- cor(o2)

Cmd> out <- extractml(R,2)$L
(1,1)      0.87422            0
(2,1)      0.84993      -0.5069
(3,1)      0.49093      0.82364
(4,1)      0.53638      0.61417
```

This has one real 0, but not many small numbers.
Consider rotation by the matrix

$$\mathbf{H} = \begin{bmatrix} 0.87651 & 0.48139 \\ -0.48139 & 0.87651 \end{bmatrix}$$

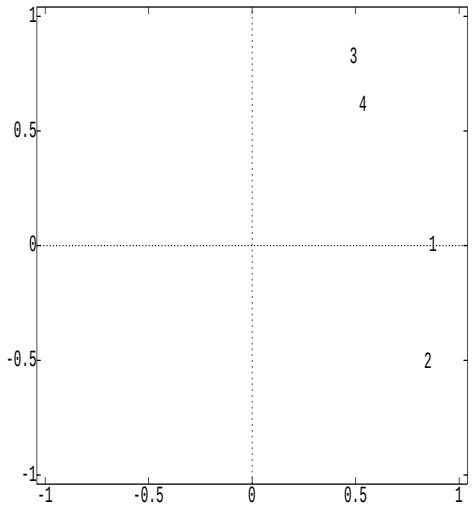
Then we get

```
Cmd> L %*% H
(1,1)      0.76626      0.42084
```

```
(2,1)      0.98898    -0.035163
(3,1)     0.033813      0.95825
(4,1)      0.17449      0.79653
```

No exact zeroes, but more smallish loadings
Here we might interpret factor 1 as “driving” much of components 1 and 2 of $x$, and factor 2 as “driving” much of components 3 and 4 of $x$.
We can look at the loadings row by row (component of $x$ by component) to see what happened after rotation.

```
Cmd> chplot(L[,1],L[,2],\
xmin:-1,ymin:-1,xmax:1,ymax:1)
```



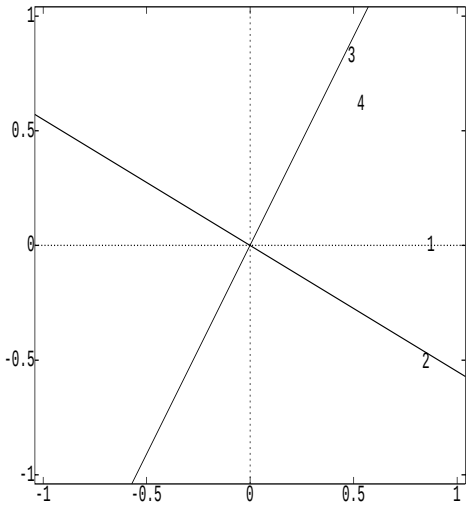

The rotated axes are

```
Cmd> new1 <- H[,1]'*vector(2,-2)

Cmd> addlines(new1[,1],new1[,2])

Cmd> new2 <- H[,2]'*vector(2,-2)

Cmd> addlines(new2[,1],new2[,2])
```




Components 2, 3, and 4 are almost on the new axes; component 1 is off somewhat.
*Factor rotation* is the exercise of finding an orthogonal matrix $\mathbf{H}$ so that $\mathbf{L}^\star = \mathbf{LH}$ is more interpretable.
Technically, $\det(\mathbf{H})$ must equal 1 for $\mathbf{H}$ to be a rotation (otherwise it includes some reflections as well). But as the directions of factors are arbitrary, reflections are acceptable.
Another technicality. We are doing orthogonal or "rigid" rotations. The axes are kept orthogonal to each other when we use orthogonal matrices $\mathbf{H}$. This keeps the identity covariance matrix for the factors $f$ (ie, the orthogonal factors model).
If we use matrices that are not orthogonal to rotate the axes, we get "oblique" rotations.
Rotations are done using various algorithmic methods that optimize some criterion.
Varimax rotation is fairly typical. Varimax argues that we want absolute values of loadings to be big or little, but not in the middle. Thus the varimax criterion is the variance of the squares of the loadings. If you make that big, you drive the loadings away from the middle towards 0 or 1.
Other rotation methods are equimax, quartimax, and orthomax with parameter $\gamma$. Consider orthomax. It tries to maximize:

$$\sum_{j=1}^{m}\left(p\sum_{i=1}^{p}\mathbf{L}_{ij}^4 - \gamma(\sum_{i=1}^{p}\mathbf{L}_{ij}^2)^2\right)$$

$\gamma$ s equal to 1, 0, and $m/2$ are equivalent to varimax, quartimax, and equimax, respectively.
One last variation is Kaiser normalization. In Kaiser normalization, the rows of $\mathbf{L}$ are scaled to have unit length:

$$\mathbf{L} = \mathbf{D}\tilde{\mathbf{L}}$$

where $\mathbf{D}$ is a diagonal matrix and the rows of $\tilde{\mathbf{L}}$ have unit length. Then $\tilde{\mathbf{L}}$ is rotated to get $\tilde{\mathbf{L}}^\star$. Note that the rows of $\tilde{\mathbf{L}}^\star$ still have unit length. Finally, $\tilde{\mathbf{L}}^\star$ is rescaled back the original row lengths in $\mathbf{L}$.

$$\mathbf{L}^\star = \mathbf{D}^{-1}\tilde{\mathbf{L}}^\star$$

The `rotation()` function in MacAnova will do these rigid rotations, with or without Kaiser normalization. Begin with the O2 data.

```
Cmd> L
(1,1)      0.87422           0
```

```
(2,1)      0.84993      -0.5069
(3,1)      0.49093      0.82364
(4,1)      0.53638      0.61417

Cmd> rotation(L,method:"varimax")
(1,1)      0.76626      0.42084
(2,1)      0.98898    -0.035163
(3,1)     0.033813      0.95825
(4,1)      0.17449      0.79653

Cmd> rotation(L,method:"varimax",kaiser:T)
(1,1)      0.77025       0.4135
(2,1)       0.9886    -0.044613
(3,1)     0.042969      0.95789
(4,1)      0.18209      0.79483

Cmd> rotation(L,method:"equimax")
(1,1)      0.76626      0.42084
(2,1)      0.98898    -0.035163
(3,1)     0.033813      0.95825
(4,1)      0.17449      0.79653

Cmd> rotation(L,method:"quartimax")
(1,1)      0.76269      0.42729
(2,1)      0.98925    -0.026822
(3,1)     0.025731      0.95851
(4,1)      0.16777      0.79798

Cmd> R <- cor(X)

Cmd> L <- extractml(R,3)$L;L

(1,1)      0.97373            0            0
(2,1)      0.98005     0.076302            0
(3,1)      0.89604      0.21178     -0.20893
(4,1)       0.8306      0.27585     -0.51123
(5,1)      0.77855      0.49133     -0.34236
(6,1)      0.68619      0.66331      -0.2826
(7,1)      0.69493      0.67458     -0.27561
(8,1)      0.58108      0.70262     -0.31463

Cmd> sum(L'^2)
(1,1)   0.94816   0.96633   0.89139    1.0273
(1,5)   0.96476   0.99069    1.0139   0.93032
```

```
Cmd> rotation(L,method:"varimax")
(1,1)      0.91094      0.32221     -0.12052
(2,1)      0.89605       0.3929    -0.095148
(3,1)      0.73357      0.54683     -0.23289
(4,1)      0.58696      0.66949     -0.48435
(5,1)      0.51746      0.79755      -0.2468
(6,1)      0.39759      0.90445     -0.12077
(7,1)      0.40426      0.91547     -0.11149
(8,1)      0.28134       0.9142     -0.12412

Cmd> rotation(L,method:"quartimax")
(1,1)      0.79639     -0.56023   -0.0076743
(2,1)      0.83881     -0.51148      0.03339
(3,1)      0.89987     -0.27725    -0.068945
(4,1)      0.96971    -0.061221     -0.28855
(5,1)      0.98089     0.041594    -0.029782
(6,1)      0.97109      0.18624      0.11396
(7,1)      0.98161      0.18584      0.12584
(8,1)      0.91408      0.28794      0.10898

Cmd> rotation(L,method:"equimax")
(1,1)      0.87854      0.24875      -0.3383
(2,1)      0.86477      0.32405     -0.33688
(3,1)      0.68177      0.42381     -0.49695
(4,1)      0.50129      0.45507     -0.75429
(5,1)      0.45804      0.65763     -0.56787
(6,1)      0.35159      0.80287     -0.47168
(7,1)      0.35902       0.8163     -0.46764
(8,1)      0.23558      0.81251      -0.4633

Cmd> cars <- matrix(vecread("cars"),22)’
Read from file "cars"

Cmd> R <- cor(cars)


Cmd> for(i,run(100))
out <- stepml(R,out,10);out$crit

(1)       1.5121
(1)       1.1746
(1)       1.0916
...
(1)       0.91422
(1)      0.91416
```

```
Cmd> ((22-10)^2-22-10)/2
(1)           56

Cmd> (90-(2*22+5)/6-2*10/3)
(1)       75.167

Cmd> .914*75.167
(1)       68.703

Cmd> 1-cumchi(68.7,56)
(1)     0.11867

Cmd> L <- out$loadings

Cmd> r <- rotation(out$loadings)

Cmd> print(r,format:"f5.2")
r:
 (1,1) -0.89  0.22  0.28 -0.21  0.16 -0.01 -0.04  0.13  0.02 -0.01
 (2,1) -0.95  0.00  0.22 -0.19  0.10  0.00 -0.02  0.06  0.02  0.01
 (3,1) -0.95 -0.17  0.17 -0.17  0.06  0.01 -0.02  0.01  0.02  0.03
 (4,1)  0.34 -0.01 -0.41  0.81 -0.15  0.13 -0.02 -0.06  0.13 -0.02
 (5,1)  0.33 -0.02 -0.34  0.82 -0.16  0.11  0.04  0.10  0.03  0.02
 (6,1) -0.54 -0.00  0.28 -0.00 -0.02 -0.01  0.11  0.14 -0.17  0.27
 (7,1)  0.29 -0.00 -0.20 -0.12  0.12  0.15 -0.03 -0.64  0.01 -0.00
 (8,1) -0.46  0.13  0.67 -0.25  0.14 -0.18 -0.06  0.03 -0.12 -0.16
 (9,1) -0.38  0.05  0.76 -0.26  0.23 -0.28 -0.02  0.14 -0.15 -0.15
(10,1) -0.61  0.04  0.68 -0.27 -0.01  0.21  0.13 -0.05 -0.02 -0.08
(11,1) -0.10 -0.00 -0.29  0.18 -0.20  0.86  0.03 -0.06  0.15 -0.01
(12,1)  0.19  0.01 -0.65  0.33 -0.15  0.24 -0.00 -0.14  0.28  0.07
(13,1)  0.21 -0.09 -0.52  0.09 -0.33  0.19  0.36  0.03  0.35  0.04
(14,1) -0.37 -0.09  0.59 -0.50  0.30 -0.12 -0.07 -0.03  0.13  0.02
(15,1)  0.01 -0.10  0.29 -0.38  0.66 -0.21 -0.33 -0.26 -0.09  0.01
(16,1) -0.31  0.03  0.64 -0.26  0.34 -0.12 -0.06  0.30 -0.19  0.20
(17,1) -0.33 -0.09  0.60 -0.30  0.49 -0.18 -0.15  0.07 -0.05  0.19
(18,1) -0.20 -0.06  0.79 -0.30  0.23 -0.24 -0.09  0.07 -0.14  0.11
(19,1) -0.18  0.01  0.66 -0.29  0.26 -0.22  0.06 -0.02 -0.30  0.19
(20,1) -0.16  0.05  0.23 -0.10  0.91 -0.10  0.07  0.03 -0.01 -0.02
(21,1) -0.41 -0.03  0.69 -0.47  0.28 -0.16 -0.07 -0.05  0.04  0.12
(22,1)  0.20 -0.01  0.32 -0.10  0.02 -0.29 -0.03  0.01 -0.68  0.02

Cmd> rc <- floor(4*abs(r))+1

Cmd> rc <- matrix(\
vector(" ",".","o","*")[vector(rc)],22)

Cmd> rc <- paste(rc,multiline:T)
```

```
Cmd> rc
 (1) "*   .              "
 (2) "*                  "
 (3) "*                  "
 (4) ".   . *            "
 (5) ".   . *            "
 (6) "o   .             ."
 (7) ".             o    "
 (8) ".   o              "
 (9) ".   * .   .        "
(10) "o   o .            "
(11) "    .     *        "
(12) "    o .         .  "
(13) "    o   .   .   .  "
(14) ".   o . .          "
(15) "    . . o   . .    "
(16) ".   o . .     .    "
(17) ".   o . .          "
(18) "    * .            "
(19) "    o . .       .  "
(20) "        *          "
(21) ".   o . .          "
(22) "    .     .     o  "
```