# Exploratory Factor Analysis with R

James H. Steiger

Exploratory Factor Analysis with R can be performed using the **factanal** function. In addition to this standard function, some **additional** facilities are provided by the **fa.promax** function written by Dirk Enzmann, the **psych** library from William Revelle, and the Steiger R Library functions. To investigate some of the capabilities of these functions and work through this handout, please set up your working directory (either through the menu system or using the **setwd** command).

```
setwd("d:/!!!Current Projects/!!!!!P312/2011/R Working/Factor Analysis/")
```

Then download the extra functions from the R Support Materials page at the course website and load them in with the commands:

```
source("fa.promax.R")
source("Steiger R Library Functions.R")
```

Make sure that the **Hmisc** library is installed on your computer and has been loaded, as follows.

```
install.packages(Hmisc)
library(Hmisc)
```

Once the library is loaded, you can load the AthleticsData file and attach it with the commands

```
AthleticsData <- spss.get("AthleticsData.sav")
attach(AthleticsData)
```

**spss.get** has changed the variable names slightly, as you can see by calling the **names** function.

```
names(AthleticsData)
```

```
[1] "PINBALL"   "BILLIARD"  "GOLF"      "X.1500M"   "X.2KROW"   "X.12MINTR"
[7] "BENCH"     "CURL"      "MAXPUSHU"
```

## Common Factor Extraction and Rotation with `factanal`

As mentioned in class, there are in wide use two primary approaches to "factor analytic" methods: (a) common factor analysis, and (b) component analysis. In this section, we discuss the common factor model.

The common factor model is a very restrictive model. It never fits perfectly in the sample (unless the sample is one we have constructed to fit perfectly!), and so we "fit the common factor model" in practice by making the discrepancy between the sample covariance matrix and the "reproduced" matrix as small as possible, according to a criterion known as a "discrepancy function." Specifically, the orthogonal common factor model implies that

$$\mathbf{\Sigma} = \mathbf{F}\mathbf{F}' + \mathbf{U}^2 \tag{1.1}$$

Of course, we don't know $\mathbf{\Sigma}$, and because of sampling error, even if the common factor model fit $\mathbf{\Sigma}$ perfectly, it would not fit the sample covariance matrix $\mathbf{S}$ perfectly. In practice then, we have

$$\mathbf{S} = \hat{\mathbf{F}}\hat{\mathbf{F}}' + \hat{\mathbf{U}}^2 + \mathbf{E} = \hat{\mathbf{\Sigma}} + \mathbf{E} \tag{1.2}$$

where $\mathbf{E}$ is made as small as possible according to some criterion. This criterion is a function of $\mathbf{S}$ and $\hat{\mathbf{\Sigma}}$, and reflects the size of the discrepancy between them.

There are a number of discrepancy functions in use. Perhaps the most popular is the "maximum likelihood (ML)" discrepancy function. When $\hat{\mathbf{F}}$ and $\hat{\mathbf{U}}$ are chosen to minimize the ML discrepancy function, they are referred to as "maximum likelihood estimates."

Maximum likelihood estimates are obtained by iteration, a process in which $\hat{\mathbf{F}}$ and $\hat{\mathbf{U}}$ are systematically altered to make the maximum likelihood discrepancy function get smaller and smaller.

As discussed in the handout on "The Algebra of Factor Analysis," for any $\hat{\mathbf{F}}$ in Equation (1.2), there are infinitely many alternative factor patterns that fit equally well. These are obtainable by "orthogonal" or "oblique" transformation. The process of transforming a factor pattern is generally referred to as "rotation." There are many methods of rotation. Two very popular methods are "varimax" rotation for orthogonal factors and "promax" rotation for oblique factors. Both methods are implemented in R.

The **`factanal`** function fits a common factor model by the method of maximum likelihood. You can find out a bit about the function through the R help system. Note: the function can analyze either raw data or a correlation or covariance matrix.
To begin with, let's analyze the AthleticsData with a 2 factor model.

```
fit.2 <- factanal(AthleticsData,factors=2,rotation="varimax")
print(fit.2)
```

```
Call:
factanal(x = AthleticsData, factors = 2, rotation = "varimax")

Uniquenesses:
  PINBALL  BILLIARD      GOLF    X.1500M    X.2KROW X.12MINTR     BENCH
CURL
    0.938     0.962     0.955     0.361     0.534     0.536     0.301
0.540
 MAXPUSHU
    0.560

Loadings:
          Factor1 Factor2
PINBALL    0.249
BILLIARD   0.190
GOLF       0.203
X.1500M   -0.137   0.787
X.2KROW    0.387   0.563
X.12MINTR          0.681
BENCH      0.821  -0.154
CURL       0.676
MAXPUSHU   0.526   0.404

               Factor1 Factor2
SS loadings      1.717   1.595
Proportion Var   0.191   0.177
Cumulative Var   0.191   0.368

Test of the hypothesis that 2 factors are sufficient.
The chi square statistic is 652.4 on 19 degrees of freedom.
The p-value is 4.3e-126
```

Near the bottom of the output, we can see that the significance level of the $\chi^2$ fit statistic is very small. This indicates that the hypothesis of perfect model fit is rejected. Since we are in a purely exploratory vein, let's fit a 3 factor model.

```
fit.3 <- factanal(AthleticsData,factors=3,rotation="varimax")
print(fit.3)
```

```
Call:
factanal(x = AthleticsData, factors = 3, rotation = "varimax")

Uniquenesses:
  PINBALL  BILLIARD      GOLF    X.1500M    X.2KROW X.12MINTR     BENCH
CURL
    0.635     0.414     0.455     0.361     0.520     0.538     0.302
0.536
 MAXPUSHU
    0.540

Loadings:
          Factor1 Factor2 Factor3
PINBALL            0.131   0.590
BILLIARD                   0.765
GOLF                       0.735
X.1500M    0.779  -0.179
X.2KROW    0.585   0.372
X.12MINTR  0.678
BENCH     -0.119   0.816   0.137
CURL               0.674
MAXPUSHU   0.433   0.522

               Factor1 Factor2 Factor3
SS loadings      1.613   1.584   1.502
Proportion Var   0.179   0.176   0.167
Cumulative Var   0.179   0.355   0.522

Test of the hypothesis that 3 factors are sufficient.
The chi square statistic is 12.94 on 12 degrees of freedom.
The p-value is 0.373
```

These results are much more promising. Although the sample size is reasonably large, $N = 1000$, the significance level of .373 indicates that the hypothesis of perfect fit cannot be rejected. Changing from two factors to three has produced a huge improvement.

We can "clean up" the factor pattern in several ways. One way is to hide small loadings, to reduce the visual clutter in the factor pattern. Another is to reduce the number of decimal places from 3 to 2. A third way is to sort the loadings to make the simple structure more obvious. The following command does all three.

```
print(fit.3, digits = 2, cutoff = .2, sort = TRUE)
```

```
Call:
factanal(x = AthleticsData, factors = 3, rotation = "varimax")

Uniquenesses:
  PINBALL  BILLIARD      GOLF   X.1500M    X.2KROW X.12MINTR     BENCH      CURL
     0.64      0.41      0.46      0.36      0.52      0.54      0.30      0.54
 MAXPUSHU
     0.54

Loadings:
          Factor1 Factor2 Factor3
X.1500M    0.78
X.2KROW    0.58    0.37
X.12MINTR  0.68
BENCH              0.82
CURL               0.67
MAXPUSHU   0.43    0.52
PINBALL                    0.59
BILLIARD                   0.76
GOLF                       0.73

               Factor1 Factor2 Factor3
SS loadings       1.61    1.58    1.50
Proportion Var    0.18    0.18    0.17
Cumulative Var    0.18    0.36    0.52

Test of the hypothesis that 3 factors are sufficient.
The chi square statistic is 12.94 on 12 degrees of freedom.
The p-value is 0.373
```

Now it is obvious that there are 3 factors. The traditional approach to naming factors is as follows:

- Examine the variables that load heavily on the factor
- Try do decide what construct is common to these variables
- Name the factor after that construct

It seems that there are three factors. The first factor is something that is common to strong performance in a 1500 meter run, a 2000 meter row, and a 12 minute run. It seems like a good name for this factor is "Endurance." The other two factors might be named “Strength,” and “Hand-Eye Coordination.” We can add these names to the loading matrix as follows:

```
colnames(fit.3$loadings)<-c("Endurance","Strength","Hand-Eye")
print(loadings(fit.3), digits = 2, cutoff = .2, sort = TRUE)
```

```
Loadings:
          Endurance Strength Hand-Eye
X.1500M    0.78
X.2KROW    0.58      0.37
X.12MINTR  0.68
BENCH                0.82
CURL                 0.67
MAXPUSHU   0.43      0.52
PINBALL                       0.59
BILLIARD                      0.76
GOLF                          0.73

               Endurance Strength Hand-Eye
SS loadings         1.61     1.58     1.50
Proportion Var      0.18     0.18     0.17
Cumulative Var      0.18     0.36     0.52
```

You can obtain an oblique promax solution by using the option rotation = promax.

```
fit.3.promax <- update(fit.3,rotation="promax")
colnames(fit.3.promax$loadings)<-c("Endurance","Strength","Hand-Eye")
print(loadings(fit.3.promax), digits = 2, cutoff = .2, sort = TRUE)
```

```
Loadings:
          Endurance Strength Hand-Eye
X.1500M    0.82     -0.29
X.2KROW    0.55      0.31
X.12MINTR  0.70
BENCH     -0.23      0.86
CURL                 0.70
PINBALL                       0.58
BILLIARD                      0.77
GOLF                          0.73
MAXPUSHU   0.37      0.49
```

For more information about the rotation methods, consult the R help with the command **?varimax**.

## Enzmann's Enhanced `fa.promax` Function

Dirk Enzmann has made an enhanced version of the **`factanal`** function available online. This function will compute and save a number of key quantities in its fit object. In particular, it automatically computes unrotated, varimax rotated, and promax rotated solutions, as well as the factor correlation matrix.

With Enzmann's function and some of the factor analysis utilities we have provided, many other interesting quantities can be computed.

Let's take a quick look at some input and output from **`fa.promax`**.

To enhance the output with factor names, use the following function.

```
AssignFactorNames <- function(fit.object,names)
{
colnames(fit.object$promax.loadings)<-names
colnames(fit.object$varimax.loadings)<-names
rownames(fit.object$corr.factors)<-names
colnames(fit.object$corr.factors)<-names
}
```

Here is a factor analysis of our *AthleticsData* file. The cutoff function does not work.

```
fit.3.Enzmann <- fa.promax(AthleticsData,factors=3, digits=2, sort=TRUE)
AssignFactorNames(fit.3.Enzmann,factor.names)
fit.3.Enzmann

$uniqueness
          residual variance
BENCH                  0.30
X.1500M                0.36
BILLIARD               0.41
GOLF                   0.46
X.2KROW                0.52
CURL                   0.54
X.12MINTR              0.54
MAXPUSHU               0.54
PINBALL                0.64

$unrotated.loadings
          Factor1 Factor2 Factor3
```

```
X.1500M       0.80   -0.01    0.01
X.12MINTR     0.67    0.10   -0.03
X.2KROW       0.50    0.41   -0.26
BENCH        -0.28    0.73   -0.30
CURL         -0.16    0.61   -0.27
MAXPUSHU      0.32    0.51   -0.31
BILLIARD      0.03    0.44    0.62
GOLF          0.05    0.45    0.58
PINBALL      -0.02    0.43    0.43

$varimax.SS
               Factor1 Factor2 Factor3
SS loadings       1.61    1.58    1.50
Proportion Var    0.18    0.18    0.17
Cumulative Var    0.18    0.36    0.52

$varimax.loadings
          Factor1 Factor2 Factor3
X.1500M       0.78   -0.18    0.02
X.12MINTR     0.68   -0.04    0.04
X.2KROW       0.58    0.37    0.01
BENCH        -0.12    0.82    0.14
CURL         -0.02    0.67    0.10
MAXPUSHU      0.43    0.52    0.02
BILLIARD      0.02    0.03    0.76
GOLF          0.05    0.05    0.73
PINBALL      -0.01    0.13    0.59

$promax.SS
               Factor1 Factor2 Factor3
SS loadings       1.63    1.61    1.47
Proportion Var    0.18    0.18    0.16
Cumulative Var    0.18    0.36    0.52

$promax.loadings
          Factor1 Factor2 Factor3
X.1500M       0.80   -0.25    0.03
X.12MINTR     0.69   -0.10    0.04
X.2KROW       0.56    0.33   -0.04
BENCH        -0.19    0.84    0.03
CURL         -0.08    0.69    0.01
MAXPUSHU      0.40    0.50   -0.05
BILLIARD      0.02   -0.02    0.77
GOLF          0.04    0.00    0.74
PINBALL      -0.02    0.10    0.58

$promax.structure
          Factor1 Factor2 Factor3
X.1500M       0.76   -0.11    0.01
X.12MINTR     0.67    0.02    0.04
X.2KROW       0.61    0.42    0.04
BENCH        -0.05    0.81    0.19
CURL          0.03    0.68    0.14
MAXPUSHU      0.47    0.55    0.06
BILLIARD      0.04    0.13    0.77
GOLF          0.06    0.15    0.74
PINBALL       0.01    0.20    0.60
```

```
$corr.factors
        Factor1 Factor2 Factor3
Factor1    1.00    0.16    0.03
Factor2    0.16    1.00    0.19
Factor3    0.03    0.19    1.00

$n
[1] 1000

$chi
objective
    12.94

$df
[1] 12

$p
objective
0.3734064
```

# Principal Components in R

The **princomp** function performs component analysis in R, but unfortunately it fails to provide some of the facilities we need for cleaning up the pattern. The **psych** library from William Revelle provides more functionality. Type **?psych** to find out more about it from the help facility.

```
fit <- principal(AthleticsData, nfactors=3, rotate="varimax")
fit # print results
```

```
           V   PC2   PC1   PC3
PINBALL    1               0.75
BILLIARD   2               0.84
GOLF       3               0.83
X.1500M    4  0.84
X.2KROW    5  0.68  0.43
X.12MINTR  6  0.81
BENCH      7        0.85
CURL       8        0.82
MAXPUSHU   9  0.49  0.63

                PC2  PC1  PC3
SS loadings    2.09 2.04 1.98
Proportion Var 0.23 0.23 0.22
Cumulative Var 0.23 0.46 0.68
```