Exploratory Factor Analysis in R

In these notes we will explore a number of functions related to factor analysis that are available in the R base package as well as several very useful functions in the "psych" library by William Revelle. To access these functions, load the "psych" library.

```
> library(psych)
```

The "psych" library contains many data sets. In the following examples we will use the "bfi" data. According to the help page:

```
25 personality self report items taken from the International Personality Item Pool
(ipip.ori.org) were included as part of the Synthetic Aperture Personality
Assessment (SAPA) web based personality assessment project. The data from
1000 subjects are included here as a demonstration set for scale
construction and factor analysis.


Usage

data(bfi)

Format

A data frame with 1000 observations on the following 25 variables.

A1
    Am indifferent to the feelings of others.
A2
    Inquire about others' well-being.
A3
```

Know how to comfort others.

A4

Love children.

A5

Make people feel at ease.

C1

Am exacting in my work.

C2

Continue until everything is perfect.

C3

Do things according to a plan.

C4

Do things in a half-way manner.

C5

Waste my time.

E1

Don't talk a lot.

E2

Find it difficult to approach others.

E3

Know how to captivate people.

E4

Make friends easily.

E5

Take charge.

N1

Get angry easily.

N2

Get irritated easily.

N3

```
    Have frequent mood swings.
N4
    Often feel blue.
N5
    Panic easily.
O1
    Am full of ideas.
O2
    Avoid imposing my will on others.
O3
    Carry the conversation to a higher level.
O4
    Spend time reflecting on things.
O5
    Will not probe deeply into a subject.

Details

The 25 items are organized by five putative factors:
Agreeableness, Conscientiousness,
Extraversion, Neuroticism, and Opennness. The scoring
key is created using make.keys,
the scores are found using score.items
Source

The items are from the ipip (Goldberg, 1999). The data are from the SAPA project
(Revelle, Wilt and Rosenthal, 2009), collected Fall, 2006.
```

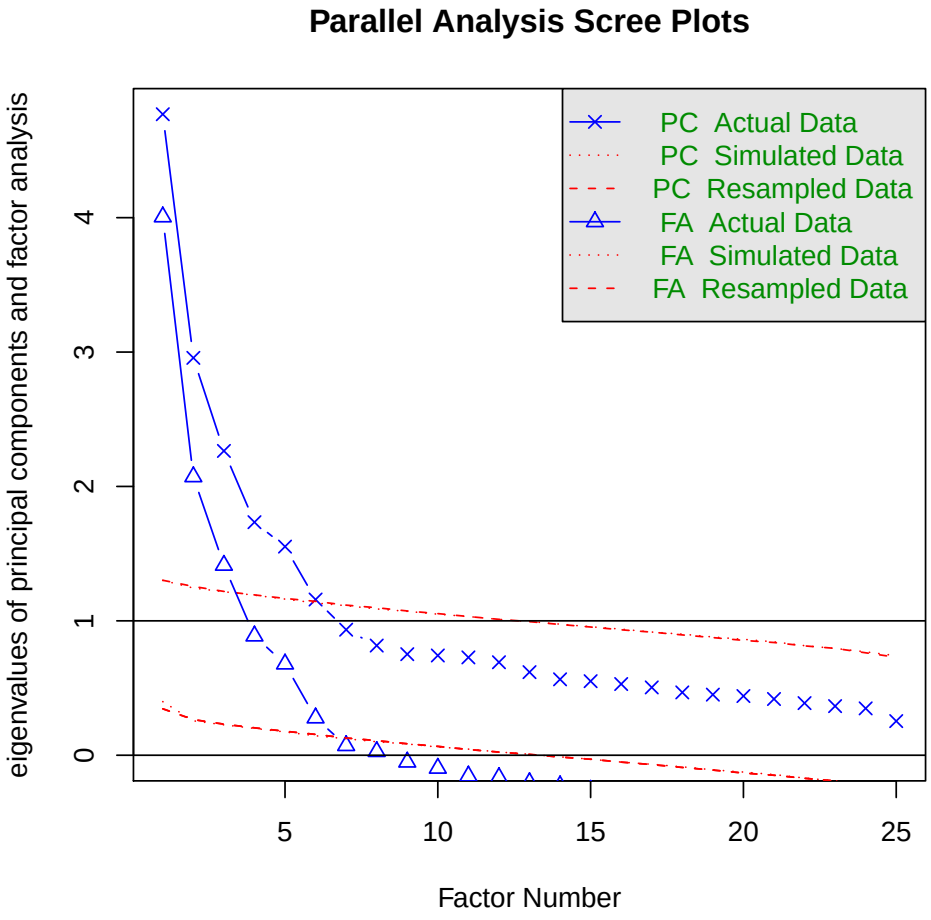
The bfi items were written to measure 5 higher-order personality dimensions. Let us perform a parallel analysis on these data to see if a 5 dimensional solution is reasonable.

```
> data(bfi)
> describe(bfi)
```

	var	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
A1	1	1000	2.29	1.28	2	2.13	1.48	1	6	5	0.87	-0.11	0.04
A2	2	994	4.82	1.12	5	4.98	1.48	1	6	5	-1.08	1.03	0.04
A3	3	989	4.60	1.21	5	4.76	1.48	1	6	5	-1.00	0.60	0.04
A4	4	993	4.76	1.40	5	5.00	1.48	1	6	5	-1.08	0.29	0.04
A5	5	988	4.58	1.15	5	4.70	1.48	1	6	5	-0.76	0.24	0.04
C1	6	997	4.39	1.22	5	4.50	1.48	1	6	5	-0.72	0.00	0.04
C2	7	997	4.22	1.29	4	4.32	1.48	1	6	5	-0.64	-0.18	0.04
C3	8	995	4.28	1.26	5	4.38	1.48	1	6	5	-0.68	-0.13	0.04
C4	9	986	2.63	1.36	2	2.51	1.48	1	6	5	0.51	-0.76	0.04
C5	10	997	3.47	1.52	4	3.48	1.48	1	6	5	-0.09	-1.06	0.05
E1	11	996	2.93	1.58	3	2.84	1.48	1	6	5	0.36	-1.07	0.05
E2	12	994	3.32	1.58	3	3.29	1.48	1	6	5	0.05	-1.16	0.05
E3	13	994	3.98	1.29	4	4.02	1.48	1	6	5	-0.35	-0.55	0.04
E4	14	997	4.42	1.43	5	4.59	1.48	1	6	5	-0.86	-0.19	0.05
E5	15	991	4.38	1.24	5	4.50	1.48	1	6	5	-0.72	-0.03	0.04
N1	16	990	2.83	1.51	3	2.72	1.48	1	6	5	0.46	-0.88	0.05
N2	17	990	3.51	1.53	4	3.52	1.48	1	6	5	0.01	-1.07	0.05
N3	18	997	3.20	1.52	3	3.15	1.48	1	6	5	0.22	-1.06	0.05
N4	19	996	3.06	1.54	3	2.97	1.48	1	6	5	0.35	-0.96	0.05
N5	20	992	2.94	1.55	3	2.84	1.48	1	6	5	0.42	-0.95	0.05
O1	21	994	4.74	1.18	5	4.88	1.48	1	6	5	-0.82	0.13	0.04
O2	22	994	3.93	1.33	4	3.95	1.48	1	6	5	-0.15	-0.75	0.04
O3	23	991	4.32	1.22	5	4.42	1.48	1	6	5	-0.66	-0.05	0.04
O4	24	992	4.83	1.18	5	5.00	1.48	1	6	5	-1.01	0.58	0.04
O5	25	996	2.51	1.22	2	2.39	1.48	1	6	5	0.65	-0.22	0.04

```
> fa.parallel(bfi)
```

```
Parallel analysis suggests that the number of factors =  6  and the number of components
```



Based on theory, we will extract and examine a 5 factor, orthogonal solution. Using the base package function for maximum likelihood factor analysis:

```
> ml5.out <- factanal(covmat = cor(bfi, use = "complete.obs"),
+     factors = 5, rotation = "none")
> ml5.out

Call:
factanal(factors = 5, covmat = cor(bfi, use = "complete.obs"),     rotation = "none")


Uniquenesses:
   A1    A2    A3    A4    A5    C1    C2    C3    C4    C5    E1    E2    E3
```

```
0.848 0.630 0.642 0.829 0.442 0.566 0.635 0.572 0.504 0.603 0.541 0.457 0.541
   E4    E5    N1    N2    N3    N4    N5    O1    O2    O3    O4    O5
0.420 0.549 0.272 0.321 0.526 0.514 0.675 0.625 0.804 0.544 0.630 0.814

Loadings:
   Factor1 Factor2 Factor3 Factor4 Factor5
A1  0.248                   -0.282
A2 -0.402   0.290           0.215   0.271
A3 -0.413   0.310           0.212   0.207
A4 -0.306                           0.273
A5 -0.556   0.294  -0.139   0.198   0.322
C1 -0.193   0.201   0.596
C2 -0.136   0.195   0.537           0.128
C3 -0.274   0.262   0.484  -0.167   0.150
C4  0.432          -0.526   0.180
C5  0.391          -0.401   0.281
E1  0.404  -0.401   0.307   0.126   0.158
E2  0.565  -0.291   0.238   0.266   0.112
E3 -0.438   0.487  -0.119   0.102
E4 -0.552   0.321  -0.345  -0.118   0.197
E5 -0.371   0.482          -0.170  -0.218
N1  0.604   0.587          -0.132
N2  0.573   0.555          -0.199
N3  0.518   0.443
N4  0.588   0.220           0.294
N5  0.418   0.236   0.107           0.276
O1 -0.199   0.363   0.144   0.292  -0.312
O2         -0.147   0.210   0.229   0.278
O3 -0.279   0.441           0.280  -0.310
O4          0.285   0.212   0.477  -0.118
```

```
O5  0.130  -0.124  -0.120  -0.165   0.335

                Factor1 Factor2 Factor3 Factor4 Factor5
SS loadings       4.00   2.635   1.784   1.116   0.961
Proportion Var    0.16   0.105   0.071   0.045   0.038
Cumulative Var    0.16   0.265   0.337   0.381   0.420

The degrees of freedom for the model is 185 and the fit was 0.8649
```

Notice that, by default, R fails to print loadings lower than an arbitrary threshold. To see all loadings, we use the print.loadings function and set the cutoff argument to a very small number.

```
> print(loadings(ml5.out), cutoff = 1e-05)

Loadings:
   Factor1 Factor2 Factor3 Factor4 Factor5
A1  0.248   0.005  -0.044  -0.282  -0.094
A2 -0.402   0.290  -0.069   0.215   0.271
A3 -0.413   0.310  -0.058   0.212   0.207
A4 -0.306   0.046   0.027  -0.003   0.273
A5 -0.556   0.294  -0.139   0.198   0.322
C1 -0.193   0.201   0.596  -0.019  -0.004
C2 -0.136   0.195   0.537  -0.061   0.128
C3 -0.274   0.262   0.484  -0.167   0.150
C4  0.432  -0.021  -0.526   0.180   0.004
C5  0.391   0.056  -0.401   0.281  -0.037
E1  0.404  -0.401   0.307   0.126   0.158
E2  0.565  -0.291   0.238   0.266   0.112
E3 -0.438   0.487  -0.119   0.102  -0.078
E4 -0.552   0.321  -0.345  -0.118   0.197
E5 -0.371   0.482   0.062  -0.170  -0.218
```

```
N1  0.604   0.587  -0.027  -0.132   0.017
N2  0.573   0.555   0.016  -0.199   0.062
N3  0.518   0.443  -0.008   0.048   0.088
N4  0.588   0.220   0.061   0.294   0.040
N5  0.418   0.236   0.107   0.084   0.276
O1 -0.199   0.363   0.144   0.292  -0.312
O2 -0.012  -0.147   0.210   0.229   0.278
O3 -0.279   0.441   0.094   0.280  -0.310
O4  0.048   0.285   0.212   0.477  -0.118
O5  0.130  -0.124  -0.120  -0.165   0.335

                Factor1 Factor2 Factor3 Factor4 Factor5
SS loadings       4.00   2.635   1.784   1.116   0.961
Proportion Var    0.16   0.105   0.071   0.045   0.038
Cumulative Var    0.16   0.265   0.337   0.381   0.420
```

Obviously, the unrotated solution is difficult to interpret. Later we will see how to use Jennrich's Gradient Projection Algorithm to perform various types of rotations on existing loading matrices. At this point, we will call the GPA rotation algorithms using Revelle's very flexible factor routines. For example, to perform a principal axes factor analysis of the bfi data with varimax rotation,

```
> pa.out <- factor.pa(r = bfi, nfactors = 5, residuals = FALSE,
+     rotate = "varimax", n.obs = NA, scores = FALSE, SMC = TRUE,
+     missing = FALSE, impute = "median", min.err = 0.001, digits = 2,
+     max.iter = 100, symmetric = TRUE, warnings = TRUE, fm = "pa")
> pa.out

Factor Analysis using method =  pa
Call: factor.pa(r = bfi, nfactors = 5, residuals = FALSE, rotate = "varimax",
    n.obs = NA, scores = FALSE, SMC = TRUE, missing = FALSE,
    impute = "median", min.err = 0.001, digits = 2, max.iter = 100,
```

```
    symmetric = TRUE, warnings = TRUE, fm = "pa")
   item   PA2   PA1   PA3   PA4   PA5   h2   u2
A1    1       -0.30                    0.21 0.79
A2    2        0.62                    0.42 0.58
A3    3        0.57                    0.37 0.63
A4    4        0.36                    0.18 0.82
A5    5        0.67                    0.49 0.51
C1    6              0.60              0.42 0.58
C2    7              0.58              0.38 0.62
C3    8              0.62              0.42 0.58
C4    9             -0.65              0.51 0.49
C5   10             -0.55              0.39 0.61
E1   11       -0.36        0.51        0.43 0.57
E2   12       -0.36        0.57        0.51 0.49
E3   13        0.44       -0.37 -0.35 0.45 0.55
E4   14        0.62       -0.41        0.57 0.43
E5   15                   -0.50 -0.30 0.45 0.55
N1   16  0.79                          0.68 0.32
N2   17  0.76                          0.63 0.37
N3   18  0.70                          0.51 0.49
N4   19  0.58              0.31        0.53 0.47
N5   20  0.56                          0.36 0.64
O1   21                         -0.54 0.33 0.67
O2   22                    0.39        0.18 0.82
O3   23                         -0.59 0.44 0.56
O4   24                         -0.49 0.37 0.63
O5   25                          0.41 0.18 0.82

                PA2  PA1  PA3  PA4  PA5
SS loadings    2.65 2.47 2.04 1.72 1.54
```

```
Proportion Var 0.11 0.10 0.08 0.07 0.06
Cumulative Var 0.11 0.20 0.29 0.36 0.42

Test of the hypothesis that 5 factors are sufficient.

The degrees of freedom for the null model are  300  and the objective function was  7.54
The degrees of freedom for the model are 185  and the objective function was  0.88
The number of observations was  1000  with Chi Square =  867.04  with prob <  1e-88

Tucker Lewis Index of factoring reliability =  0.84
Fit based upon off diagonal values = 0.97
Measures of factor score adequacy
                                                PA2  PA1  PA3  PA4  PA5
Correlation of scores with factors             0.92 0.87 0.87 0.85 0.82
Multiple R square of scores with factors       0.85 0.75 0.76 0.72 0.67
Minimum correlation of factor score estimates 0.69 0.51 0.52 0.44 0.33
```

And to see all loadings

```
> print(pa.out$loadings, cutoff = 1e-05, digits = 2)

Loadings:
   PA2   PA1   PA3   PA4   PA5
A1  0.15 -0.30 -0.02 -0.25  0.18
A2  0.00  0.62  0.10  0.01 -0.14
A3 -0.05  0.57  0.09 -0.04 -0.19
A4 -0.10  0.36  0.17  0.02  0.09
A5 -0.11  0.67  0.07 -0.09 -0.11
C1  0.02  0.01  0.60  0.07 -0.24
C2  0.11  0.08  0.58  0.09 -0.10
C3  0.06  0.17  0.62 -0.07 -0.04
C4  0.26 -0.09 -0.65  0.03  0.11
```

```
C5  0.28 -0.09 -0.55  0.08 -0.07
E1  0.04 -0.36  0.02  0.51  0.20
E2  0.21 -0.36 -0.09  0.57  0.07
E3  0.00  0.44  0.08 -0.37 -0.35
E4 -0.10  0.62  0.02 -0.41  0.05
E5  0.02  0.19  0.27 -0.50 -0.30
N1  0.79 -0.15 -0.05 -0.18 -0.03
N2  0.76 -0.14  0.02 -0.19  0.03
N3  0.70 -0.06 -0.08  0.02 -0.05
N4  0.58 -0.17 -0.19  0.31 -0.16
N5  0.56  0.03  0.04  0.21  0.09
O1  0.03  0.11  0.11 -0.12 -0.54
O2 -0.05  0.11  0.11  0.39  0.02
O3  0.04  0.21  0.10 -0.20 -0.59
O4  0.20  0.12  0.04  0.26 -0.49
O5  0.08  0.01 -0.06  0.05  0.41

                PA2  PA1  PA3  PA4  PA5
SS loadings    2.65 2.47 2.04 1.72 1.54
Proportion Var 0.11 0.10 0.08 0.07 0.06
Cumulative Var 0.11 0.20 0.29 0.36 0.42
```

The problem of local minima: Varimax

```
> library(GPArotation)
> pa5.loadings <- print(loadings(pa.out), cutoff = 1e-08)

Loadings:
   PA2    PA1    PA3    PA4    PA5
A1  0.154 -0.301 -0.021 -0.254  0.178
A2 -0.002  0.621  0.100  0.012 -0.141
A3 -0.052  0.569  0.092 -0.042 -0.194
```

```
A4 -0.105  0.364  0.167  0.023  0.090
A5 -0.112  0.674  0.073 -0.088 -0.108
C1  0.016  0.008  0.602  0.066 -0.239
C2  0.114  0.082  0.584  0.088 -0.096
C3  0.059  0.169  0.619 -0.074 -0.045
C4  0.263 -0.089 -0.649  0.033  0.110
C5  0.276 -0.089 -0.547  0.077 -0.066
E1  0.038 -0.358  0.016  0.512  0.196
E2  0.209 -0.360 -0.093  0.568  0.067
E3 -0.003  0.436  0.079 -0.367 -0.350
E4 -0.105  0.623  0.018 -0.408  0.048
E5  0.019  0.194  0.275 -0.500 -0.301
N1  0.789 -0.147 -0.050 -0.182 -0.031
N2  0.756 -0.143  0.018 -0.192  0.028
N3  0.702 -0.060 -0.081  0.025 -0.047
N4  0.581 -0.175 -0.186  0.313 -0.157
N5  0.555  0.025  0.038  0.215  0.085
O1  0.029  0.108  0.111 -0.117 -0.540
O2 -0.050  0.107  0.110  0.387  0.023
O3  0.038  0.210  0.103 -0.199 -0.586
O4  0.204  0.115  0.040  0.260 -0.494
O5  0.085  0.015 -0.060  0.053  0.413

                 PA2   PA1   PA3   PA4   PA5
SS loadings    2.655 2.468 2.036 1.723 1.539
Proportion Var 0.106 0.099 0.081 0.069 0.062
Cumulative Var 0.106 0.205 0.286 0.355 0.417
```

```
> Global.min <- function(A, method, B = 10) {
+     fv <- rep(0, B)
+     seeds <- sample(1e+07, B)
```

```
+     for (i in 1:B) {
+         cat(i, " ")
+         set.seed(seeds[i])
+         gpout <- GPForth(A = A, Random.Start(ncol(A)), method = method)
+         dtab <- dim(gpout$Table)
+         fv[i] <- gpout$Table[dtab[1], 2]
+         cat(fv[i], "\n")
+     }
+     cat("Min is ", min(fv), "\n")
+     set.seed(seeds[order(fv)[1]])
+     ans <- GPForth(A = A, Random.Start(ncol(A)), method = method,
+         normalize = TRUE)
+     ans
+ }
> Global.min(pa5.loadings, "varimax", 10)

1  -0.5974817
2  -0.5974817
3  -0.5974817
4  -0.5974817
5  -0.5974817
6  -0.5974817
7  -0.5974817
8  -0.5974817
9  -0.5974817
10  -0.5974817
Min is  -0.5974817
Orthogonal rotation method varimax converged.
Loadings:
        PA2     PA1     PA3       PA4      PA5
A1 -0.15478  0.0215  0.1794  0.248079  0.30470
```

```
A2  0.00225 -0.1003 -0.1405 -0.000941 -0.62085
A3  0.05229 -0.0922 -0.1932  0.052588 -0.56814
A4  0.10440 -0.1665  0.0904 -0.017147 -0.36485
A5  0.11148 -0.0725 -0.1071  0.099855 -0.67263
C1 -0.01544 -0.6024 -0.2391 -0.063999 -0.00957
C2 -0.11382 -0.5838 -0.0968 -0.086396 -0.08307
C3 -0.05955 -0.6190 -0.0440  0.077388 -0.16817
C4 -0.26289  0.6492  0.1092 -0.035957  0.08856
C5 -0.27530  0.5472 -0.0668 -0.078080  0.08797
E1 -0.03666 -0.0157  0.1932 -0.518326  0.34954
E2 -0.20824  0.0928  0.0640 -0.574842  0.35083
E3  0.00254 -0.0788 -0.3480  0.376327 -0.43025
E4  0.10402 -0.0178  0.0502  0.417613 -0.61658
E5 -0.01943 -0.2746 -0.2983  0.504608 -0.18566
N1 -0.78895  0.0501 -0.0300  0.178121  0.15040
N2 -0.75631 -0.0175  0.0286  0.188107  0.14608
N3 -0.70187  0.0815 -0.0478 -0.026366  0.05972
N4 -0.58075  0.1858 -0.1591 -0.315919  0.16975
N5 -0.55494 -0.0379  0.0836 -0.215772 -0.02859
O1 -0.02885 -0.1115 -0.5389  0.122080 -0.10584
O2  0.05017 -0.1104  0.0207 -0.385443 -0.11307
O3 -0.03863 -0.1026 -0.5850  0.205365 -0.20653
O4 -0.20328 -0.0398 -0.4959 -0.255111 -0.11935
O5 -0.08457  0.0601  0.4129 -0.055461 -0.01540

Rotating matrix:
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] -0.999999  6.57e-05 -0.000386 -0.001562  0.000309
[2,] -0.000334  1.95e-04  0.000245  0.015965 -0.999872
[3,] -0.000066 -1.00e+00  0.000329  0.000117 -0.000193
```

```
[4,]  0.001559 -1.16e-04 -0.005755 -0.999855 -0.015967
[5,] -0.000377  3.29e-04  0.999983 -0.005759  0.000153
```

The problem of local minima: Oblique Rotation:

```
> Global.min <- function(A, method, B = 10) {
+     fv <- rep(0, B)
+     seeds <- sample(1e+07, B)
+     for (i in 1:B) {
+         cat(i, " ")
+         set.seed(seeds[i])
+         gpout <- GPFoblq(A = A, Random.Start(ncol(A)), method = method)
+         dtab <- dim(gpout$Table)
+         fv[i] <- gpout$Table[dtab[1], 2]
+         cat(fv[i], "\n")
+     }
+     cat("Min is ", min(fv), "\n")
+     set.seed(seeds[order(fv)[1]])
+     ans <- GPFoblq(A = A, Random.Start(ncol(A)), method = method)
+     ans
+ }
> Global.min(pa5.loadings, "oblimin", 10)

1  0.2531282
2  0.2531282
3  0.2531282
4  0.2531282
5  0.2531282
6  0.2531282
7  0.2531282
8  0.2531282
9  0.2531282
```

```
10  0.2531282
Min is  0.2531282
Oblique rotation method Oblimin Quartimin converged.
Loadings:
        PA2      PA1     PA3     PA4      PA5
A1 -0.23118  0.00263 -0.3010 -0.1490  0.28505
A2  0.00190  0.05602  0.1320  0.0669 -0.63157
A3  0.05057  0.04370  0.0634  0.1341 -0.55458
A4  0.06683  0.15290  0.0681 -0.1529 -0.37906
A5  0.08526  0.01790  0.0217  0.0378 -0.67344
C1 -0.03216  0.61455  0.0847  0.1787  0.08828
C2 -0.14810  0.60356  0.1270  0.0166 -0.03596
C3 -0.14490  0.62916 -0.0339 -0.0438 -0.11523
C4 -0.19524 -0.65266  0.0503 -0.0426 -0.03137
C5 -0.18116 -0.55534  0.1065  0.1297  0.00693
E1  0.05232  0.07418  0.4503 -0.1906  0.30152
E2 -0.07256 -0.03851  0.5408 -0.0506  0.28734
E3 -0.04178  0.01667 -0.2752  0.3271 -0.36409
E4 -0.01162 -0.03905 -0.3211 -0.1029 -0.62857
E5 -0.11866  0.23422 -0.4536  0.2838 -0.08564
N1 -0.81432 -0.02586 -0.0935  0.0291  0.05646
N2 -0.80153  0.04625 -0.1113 -0.0396  0.05288
N3 -0.67679 -0.05949  0.1205  0.0363 -0.04137
N4 -0.45851 -0.15948  0.3824  0.1717  0.08696
N5 -0.52491  0.07051  0.3002 -0.1288 -0.14014
O1  0.01753  0.07143 -0.0639  0.5502  0.00328
O2  0.11779  0.12386  0.4050 -0.0620 -0.13267
O3 -0.00418  0.04964 -0.1242  0.5919 -0.09322
O4 -0.07144  0.02072  0.3444  0.4881 -0.07749
O5 -0.13332 -0.03258  0.0448 -0.4383 -0.11734
```

```
Rotating matrix:
        [,1]    [,2]      [,3]    [,4]    [,5]
[1,] -0.9986  0.0332  0.147361 -0.0416 -0.1492
[2,] -0.0232 -0.0645  0.180675 -0.1128 -1.0756
[3,] -0.1166  1.0404 -0.000107 -0.1170  0.0987
[4,]  0.2256  0.0418  1.018511 -0.0383 -0.0818
[5,] -0.1636  0.0622 -0.059143 -1.0601 -0.1904


Phi:
       [,1]    [,2]    [,3]   [,4]   [,5]
[1,]  1.000  0.1448 -0.1262 -0.127 -0.157
[2,]  0.145  1.0000 -0.0974  0.191 -0.216
[3,] -0.126 -0.0974  1.0000 -0.067  0.281
[4,] -0.127  0.1910 -0.0670  1.000 -0.282
[5,] -0.157 -0.2165  0.2815 -0.282  1.000
```

Looking Under the Hood

One of the benefits of using R is that it is often possible to inspect the function code to learn "where the numbers come from." For instance, to see how R calculates maximum likelihood factor analysis, we can type `factanal` at the prompt sign (">").

```
> factanal

function (x, factors, data = NULL, covmat = NULL, n.obs = NA,
    subset, na.action, start = NULL, scores = c("none", "regression",
        "Bartlett"), rotation = "varimax", control = NULL, ...)
{
    sortLoadings <- function(Lambda) {
        cn <- colnames(Lambda)
```

```
    Phi <- attr(Lambda, "covariance")
    ssq <- apply(Lambda, 2L, function(x) -sum(x^2))
    Lambda <- Lambda[, order(ssq), drop = FALSE]
    colnames(Lambda) <- cn
    neg <- colSums(Lambda) < 0
    Lambda[, neg] <- -Lambda[, neg]
    if (!is.null(Phi)) {
        unit <- ifelse(neg, -1, 1)
        attr(Lambda, "covariance") <- unit %*% Phi[order(ssq),
            order(ssq)] %*% unit
    }
    Lambda
}
cl <- match.call()
na.act <- NULL
if (is.list(covmat)) {
    if (any(is.na(match(c("cov", "n.obs"), names(covmat)))))
        stop("'covmat' is not a valid covariance list")
    cv <- covmat$cov
    n.obs <- covmat$n.obs
    have.x <- FALSE
}
else if (is.matrix(covmat)) {
    cv <- covmat
    have.x <- FALSE
}
else if (is.null(covmat)) {
    if (missing(x))
        stop("neither 'x' nor 'covmat' supplied")
    have.x <- TRUE
```

```
    if (inherits(x, "formula")) {
        mt <- terms(x, data = data)
        if (attr(mt, "response") > 0)
            stop("response not allowed in formula")
        attr(mt, "intercept") <- 0
        mf <- match.call(expand.dots = FALSE)
        names(mf)[names(mf) == "x"] <- "formula"
        mf$factors <- mf$covmat <- mf$scores <- mf$start <- mf$rotation <- mf$contro
        mf[[1L]] <- as.name("model.frame")
        mf <- eval.parent(mf)
        na.act <- attr(mf, "na.action")
        if (.check_vars_numeric(mf))
            stop("factor analysis applies only to numerical variables")
        z <- model.matrix(mt, mf)
    }
    else {
        z <- as.matrix(x)
        if (!is.numeric(z))
            stop("factor analysis applies only to numerical variables")
        if (!missing(subset))
            z <- z[subset, , drop = FALSE]
    }
    covmat <- cov.wt(z)
    cv <- covmat$cov
    n.obs <- covmat$n.obs
}
else stop("'covmat' is of unknown type")
scores <- match.arg(scores)
if (scores != "none" && !have.x)
    stop("requested scores without an 'x' matrix")
```

```
p <- ncol(cv)
if (p < 3)
    stop("factor analysis requires at least three variables")
dof <- 0.5 * ((p - factors)^2 - p - factors)
if (dof < 0)
    stop(gettextf("%d factors is too many for %d variables",
        factors, p), domain = NA)
sds <- sqrt(diag(cv))
cv <- cv/(sds %o% sds)
cn <- list(nstart = 1, trace = FALSE, lower = 0.005)
cn[names(control)] <- control
more <- list(...)[c("nstart", "trace", "lower", "opt", "rotate")]
if (length(more))
    cn[names(more)] <- more
if (is.null(start)) {
    start <- (1 - 0.5 * factors/p)/diag(solve(cv))
    if ((ns <- cn$nstart) > 1)
        start <- cbind(start, matrix(runif(ns - 1), p, ns -
            1, byrow = TRUE))
}
start <- as.matrix(start)
if (nrow(start) != p)
    stop(gettextf("'start' must have %d rows", p), domain = NA)
nc <- ncol(start)
if (nc < 1)
    stop("no starting values supplied")
best <- Inf
for (i in 1L:nc) {
    nfit <- factanal.fit.mle(cv, factors, start[, i], max(cn$lower,
        0), cn$opt)
```

```
    if (cn$trace)
        cat("start", i, "value:", format(nfit$criteria[1L]),
            "uniqs:", format(as.vector(round(nfit$uniquenesses,
              4))), "\n")
    if (nfit$converged && nfit$criteria[1L] < best) {
        fit <- nfit
        best <- fit$criteria[1L]
    }
}
if (best == Inf)
    stop("unable to optimize from these starting value(s)")
load <- fit$loadings
if (rotation != "none") {
    rot <- do.call(rotation, c(list(load), cn$rotate))
    load <- if (is.list(rot))
        rot$loadings
    else rot
}
fit$loadings <- sortLoadings(load)
class(fit$loadings) <- "loadings"
fit$na.action <- na.act
if (have.x && scores != "none") {
    Lambda <- fit$loadings
    zz <- scale(z, TRUE, TRUE)
    switch(scores, regression = {
        sc <- zz %*% solve(cv, Lambda)
        if (!is.null(Phi <- attr(Lambda, "covariance")))
            sc <- sc %*% Phi
    }, Bartlett = {
        d <- 1/fit$uniquenesses
```

```
            tmp <- t(Lambda * d)
            sc <- t(solve(tmp %*% Lambda, tmp %*% t(zz)))
        })
        rownames(sc) <- rownames(z)
        colnames(sc) <- colnames(Lambda)
        if (!is.null(na.act))
            sc <- napredict(na.act, sc)
        fit$scores <- sc
    }
    if (!is.na(n.obs) && dof > 0) {
        fit$STATISTIC <- (n.obs - 1 - (2 * p + 5)/6 - (2 * factors)/3) *
            fit$criteria["objective"]
        fit$PVAL <- pchisq(fit$STATISTIC, dof, lower.tail = FALSE)
    }
    fit$n.obs <- n.obs
    fit$call <- cl
    fit
}
<environment: namespace:stats>
```

Notice, that `factanal` does not actually perform the heavy lifting: the actual mle loadings are estimated in a function called `factanal.fit.mle`. Unfortunately, if we try the previous method to look at the code of this function we will be disappointed.

```
>factanal.fit.mle

Error: object 'factanal.fit.mle' not found
```

Nada! Fortunately, there is a convenient way to access the code using the `getAnywhere` function.

```
> getAnywhere(factanal.fit.mle)
```

```
A single object matching 'factanal.fit.mle' was found
It was found in the following places
  namespace:stats
with value

function (cmat, factors, start = NULL, lower = 0.005, control = NULL,
    ...)
{
    FAout <- function(Psi, S, q) {
        sc <- diag(1/sqrt(Psi))
        Sstar <- sc %*% S %*% sc
        E <- eigen(Sstar, symmetric = TRUE)
        L <- E$vectors[, 1L:q, drop = FALSE]
        load <- L %*% diag(sqrt(pmax(E$values[1L:q] - 1, 0)),
            q)
        diag(sqrt(Psi)) %*% load
    }
    FAfn <- function(Psi, S, q) {
        sc <- diag(1/sqrt(Psi))
        Sstar <- sc %*% S %*% sc
        E <- eigen(Sstar, symmetric = TRUE, only.values = TRUE)
        e <- E$values[-(1L:q)]
        e <- sum(log(e) - e) - q + nrow(S)
        -e
    }
    FAgr <- function(Psi, S, q) {
        sc <- diag(1/sqrt(Psi))
        Sstar <- sc %*% S %*% sc
        E <- eigen(Sstar, symmetric = TRUE)
        L <- E$vectors[, 1L:q, drop = FALSE]
```

```
        load <- L %*% diag(sqrt(pmax(E$values[1L:q] - 1, 0)),
            q)
        load <- diag(sqrt(Psi)) %*% load
        g <- load %*% t(load) + diag(Psi) - S
        diag(g)/Psi^2
    }
    p <- ncol(cmat)
    if (is.null(start))
        start <- (1 - 0.5 * factors/p)/diag(solve(cmat))
    res <- optim(start, FAfn, FAgr, method = "L-BFGS-B", lower = lower,
        upper = 1, control = c(list(fnscale = 1, parscale = rep(0.01,
            length(start))), control), q = factors, S = cmat)
    Lambda <- FAout(res$par, cmat, factors)
    dimnames(Lambda) <- list(dimnames(cmat)[[1L]], paste("Factor",
        1L:factors, sep = ""))
    p <- ncol(cmat)
    dof <- 0.5 * ((p - factors)^2 - p - factors)
    un <- res$par
    names(un) <- colnames(cmat)
    class(Lambda) <- "loadings"
    ans <- list(converged = res$convergence == 0, loadings = Lambda,
        uniquenesses = un, correlation = cmat, criteria = c(objective = res$value,
            counts = res$counts), factors = factors, dof = dof,
        method = "mle")
    class(ans) <- "factanal"
    ans
}
<environment: namespace:stats>
```