Module 28.1: nag_fac_analysis
Factor Analysis and Principal Component

`nag_fac_analysis` contains a procedure that performs principal component analysis for multivariate data.

Contents

Introduction

Let the n by p data matrix consist of p variables, $x_1, x_2, \ldots, x_p$, observed on n objects or individuals. *Factor analysis* and *principal component analysis* are variable-directed methods that examine the linear relationship between all the variables with the aim of reducing the dimensionality of the problem.

Principal component analysis finds new variables which are linear combinations of the observed variables, are orthogonal and have the maximum variation in the smallest number of variables. The principal components can be shown to be the eigenvectors of the variance-covariance matrix and the eigenvalues give the amount of variation for each component. The first principal component is the eigenvector associated with the largest eigenvalue and so explains the largest amount of variation. Ideally, a small number of principal components will explain most of the variation in the original data. Examining the eigenvalues will give an indication of how many components are needed to give a reasonable representation of the data.

Instead of using the variance-covariance matrix a standardised version such as the correlation matrix can be used. An alternate method of computing the principal components is to use the singular value decomposition of the scaled mean adjusted data matrix. The squared singular values are the eigenvalues given above, the right singular vectors give the loading matrix, which shows how the original variables relate to the principal components, and the left singular vectors give the principal component scores which give the observations in term of the principal components.

Factor analysis can be performed by dividing the variables in a principal component analysis by a suitable scaling factor. In the simplest case the scaling can be fixed as the inverse of the diagonal elements of the inverse of the variance-covariance matrix to give principal factor analysis.

Only one procedure is available at this release.

- `nag_prin_comp` performs a principal component analysis on a data matrix. Principal component analysis is often used to reduce the dimension of a data set, replacing a large number of correlated variables with a smaller number of orthogonal variables that still contain most of the information in the original data set.

Procedure: nag_prin_comp

1 Description

nag_prin_comp calculates the principal component loadings and scores for a given (optionally weighted) data matrix.

2 Usage

```
USE nag_fac_analysis
CALL nag_prin_comp(data, prin_var  [, optional arguments])
```

3 Arguments

Note. All array arguments are assumed-shape arrays. The extent in each dimension must be exactly that required by the problem. Notation such as '$\mathbf{x}(n)$' is used in the argument descriptions to specify that the array x must have exactly n elements.

This procedure derives the values of the following problem parameters from the shape of the supplied arrays.

$m \geq 1$ — the number of variables in the data matrix.

$p \geq 1$ — the number of variables included in the analysis. If the optional argument var_in_comp is not present then $p = m$, otherwise $p =$ COUNT(var_in_comp).

$n > p$ — the number of observations in the data matrix.

3.1 Mandatory Arguments

data(n, m) — real(kind=*wp*), intent(in)

Input: data(i, j) must contain the ith observation for the jth variable, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$.

prin_var$(p, 3)$ — real(kind=*wp*), intent(out)

Output: prin_var$(i, 1)$ contains the eigenvalue, γ_i^2, associated with the ith principal component. prin_var$(i, 2)$ contains the proportion of the variation explained by the ith principal component. prin_var$(i, 3)$ contains the cumulative proportion of the variation explained by the first i principal components.

3.2 Optional Arguments

Note. Optional arguments must be supplied by keyword, not by position. The order in which they are described below may differ from the order in which they occur in the argument list.

matrix — character(len=1), intent(in), optional

Input: specifies which type of principal component analysis is to be carried out.

If matrix = 'C' or 'c', it is for a correlation matrix;

if matrix = 'S' or 's', it is for a standardised matrix with standardisations given by the input array s;

if matrix = 'U' or 'u', it is for a sum of squares and cross-products matrix;

if matrix = 'V' or 'v', it is for a variance-covariance matrix.

Default: matrix = 'C'.

std — character(len=1), intent(in), optional

Input: specifies how the principal component scores are to be standardised.

If `std` = `'U'` or `'u'`, the sum of the squares of the scores for each principal component is equal to the corresponding eigenvalue;

if `std` = `'S'` or `'s'`, the sum of the squares of the scores for each principal component is equal to 1.0;

if `std` = `'E'` or `'e'`, the variance of the scores for each principal component is equal to the corresponding eigenvalue;

if `std` = `'Z'` or `'z'`, the variance of the scores for each principal component is unity.

Default: `std` = `'U'`.

var_in_comp(m) — logical, intent(in), optional

Input: the variables to be included in the model.

If `var_in_comp(`i`)` = `.true.`, the ith variable is *included* in the calculations;

if `var_in_comp(`i`)` = `.false.`, the ith variable is *excluded* from the calculations.

Default: all variables are included in the calculations.

wt(n) — real(kind=*wp*), intent(in), optional

Input: the weights, $w_i, i = 1, 2, \ldots, n$, that are associated with the observations.

Default: an unweighted analysis is performed.

Constraints: `wt` ≥ 0.

index(p) — integer, intent(out), optional

Output: the indices of the variables included in the calculations.

s(m) — real(kind=*wp*), intent(in), optional

Input: the standardisations to be used on the selected variables, if any. If `matrix` = `'S'` or `'s'`, then `s(`i`)`, where $1 \leq i \leq m$, contains the standardisation to be used for the ith column of the input array `data`.

Constraints: `s` must be present if `matrix` = `'s'` or `'S'`. `s(`i`)` > 0.0, for $i = 1, 2, \ldots, m$ and `var_in_comp(`i`)` = `.true.`.

score(n, p) — real(kind=*wp*), intent(out), optional

Output: the principal component scores. The jth column contains the scores for jth principal component, and `score(`i, j`)` contains the score for the ith observation on the jth principal component.

loading(p, p) — real(kind=*wp*), intent(out), optional

Output: the principal component loadings. The jth column contains the loadings for the jth principal component, and `loading(`i, j`)` contains the loading of the ith variable included in the calculations (the original `index(`i`)` variable), on the jth principal component.

chi_stat(p) — real(kind=*wp*), intent(out), optional

Output: `chi_stat(`i`)` contains the χ^2-statistic for the ith principal component, that is the test for the equality of the $i, i+1, \ldots, p$ eigenvalues.

sig_chi_stat(p) — real(kind=*wp*), intent(out), optional

Output: `sig_chi_stat(`i`)` contains significance level for the χ^2-statistic for the ith principal component.

chi_df(*p*) — integer, intent(out), optional

Output: `chi_df(`i`)` contains the number of degrees of freedom associated with the ith χ^2-statistic.

error — type(nag_error), intent(inout), optional

The NAG *fl*90 error-handling argument. See the Essential Introduction, or the module document `nag_error_handling` (1.2). You are recommended to omit this argument if you are unsure how to use it. If this argument is supplied, it *must* be initialized by a call to `nag_set_error` before this procedure is called.

4 Error Codes

Fatal errors (error%level = 3):

error%code	Description
301	An input argument has an invalid value.
302	An array argument has an invalid shape.
303	Array arguments have inconsistent shapes.
305	Invalid absence of an optional argument.
320	The procedure was unable to allocate enough memory.

Failures (error%level = 2):

error%code	Description
201	Cannot compute the principal components.
	The singular value decomposition has failed to converge.

5 Examples of Usage

A complete example of the use of this procedure appears in Example 1 of this module document.

6 Further Comments

Principal component analysis is the same as the discrete Karhunen–Loeve expansion and therefore, within a signal processing/pattern recognition context, this procedure can be used to perform data compression, optimal pattern representation and feature extraction.

6.1 Mathematical Background

Let X be the n by p mean-centred data matrix derived from the original n by m data array `data` (n observations on m variables). The n observations on the ith variable, x_i, form the ith column of X and the p by p sum of squares and cross-products matrix is $C = X^T X$, and the sample variance-covariance matrix is $S = C/(n-1)$.

The first principal component, $z_1 = \sum_{i=1}^{p} a_{1i} x_i$, is the linear combination of the variables that gives the maximum variation. The vector a_1 is such that $a_1^T S a_1$ is maximized subject to $a_1^T a_1 = 1.0$. A second principal component, $z_2 = \sum_{i=1}^{p} a_{2i} x_i$, is found such that $a_2^T S a_2$ is maximized subject to $a_2^T a_2 = 1.0$ and $a_2^T a_1 = 0.0$. This gives the linear combination of variables that is orthogonal to the first principal component and gives the maximum variation. Further principal components are derived in a similar way.

The vectors $a_1, a_2, \ldots, a_k$ are the eigenvectors of the matrix S and associated with each eigenvector is the eigenvalue, γ_i^2. The value of $\gamma_i^2/\Sigma\gamma_i^2$ gives the proportion of variation explained by the ith principal component. Some authors define the loading to be $a_j\gamma_j$. Alternatively, the a_i can be considered as the right singular vectors in a singular value decomposition (SVD), with singular values γ_i, of the scaled, mean-centred data matrix $X/\sqrt{(n-1)}$. This latter approach is used in this procedure.

Principal component analysis is often used to reduce the dimension of a data set, replacing a large number of correlated variables with a smaller number of orthogonal variables that still contain most of the information in the original data set.

The choice of the number of dimensions required is usually based on the amount of variation accounted for by the leading principal components. If k principal components are selected then a test of the equality of the remaining $p-k$ eigenvalues is

$$(n-(2p+5)/6)\left(-\sum_{i=k+1}^{p}\log(\gamma_i)+(p-k)\log\left(\sum_{i=k+1}^{p}\gamma_i/(p-k)\right)\right)$$

which has, asymptotically, a χ^2-distribution with $\frac{1}{2}(p-k-1)(p-k+2)$ degrees of freedom.

The case $k=0$ tests for equality of all eigenvalues; if all eigenvalues are not significantly different, then the original variables are independent, and principal component analysis has no advantage over examining the original variables. For $k=1,2,\ldots,p-2$, the test for quality of the remaining eigenvalues indicates that if any more principal components are to be considered then they all should be considered; that is, there is no advantage in including only some of the remaining components.

Instead of variance-covariance matrix the correlation matrix may be used. This means that the variables are standardised to have the same variance and so can be useful if the variables are measured on different scales. If the correlation matrix is used, the χ^2-approximation for the statistic given above is not valid. Alternatively the unscaled sum of squares and cross-products matrix or an externally standardised sums of squares and cross-products matrix may be used.

The principal component scores are the values of the principal component variables for the observations. These scores can be standardised so that either their variance or the sum of the squares for each principal component is equal to 1.0 or the corresponding eigenvalue.

Weights can be used with the analysis, in which case the data matrix is first centred about the weighted means and then each row is scaled by an amount $\sqrt{w_i}$, where w_i is the weight for the ith observation. For the variance-covariance matrix the divisor $(n-1)$ is replaced by $\sum_{i=1}^{n} w_i - 1$.

6.2 Algorithmic Detail

The loadings and scores are obtained by performing an SVD on the appropriately scaled n by p data matrix, $\tilde{X}$. For instance if the mean-centred data matrix is X and `matrix = 'U'`, then $\tilde{X} = X$, while if `matrix = 'V'`, then $\tilde{X} = X/\sqrt{(n-1)}$ in the unweighted case.

Let $\tilde{X} = U\Sigma V^T$, where Σ is a diagonal matrix containing the singular values γ_j, U is a n by p orthogonal matrix, and V is a p by p orthogonal matrix. The columns of V, a_j, are the eigenvectors of $\tilde{X}^T\tilde{X}$ and the columns of U are the scores standardised so that their sum of squares equals unity. Other standardisations are obtained by scaling U as appropriate.

6.3 Accuracy

As this procedure uses SVD of the data matrix, it will be less affected by ill conditioned problems than traditional methods which use an eigenvalue decomposition of the variance-covariance matrix.

Example 1: Calculation of the Principal Components and Scores

A data set is taken from Cooley and Lohnes [1]; it consists of ten observations on three variables. The unweighted principal components based on the variance-covariance matrix are computed and the unstandardised principal component scores requested.

1 Program Text

Note. The listing of the example program presented below is double precision. Single precision users are referred to Section 5.2 of the Essential Introduction for further information.

```
PROGRAM nag_fac_analysis_ex01

  ! Example Program Text for nag_fac_analysis
  ! NAG fl90, Release 4. NAG Copyright 2000.

  ! .. Use Statements ..
  USE nag_fac_analysis, ONLY : nag_prin_comp
  USE nag_examples_io, ONLY : nag_std_in, nag_std_out
  USE nag_write_mat, ONLY : nag_write_gen_mat
  ! .. Implicit None Statement ..
  IMPLICIT NONE
  ! .. Intrinsic Functions ..
  INTRINSIC KIND
  ! .. Parameters ..
  INTEGER, PARAMETER :: wp = KIND(1.0D0)
  ! .. Local Scalars ..
  INTEGER :: i, k, m, n
  CHARACTER (1) :: matrix
  ! .. Local Arrays ..
  INTEGER, ALLOCATABLE :: chi_df(:)
  REAL (wp), ALLOCATABLE :: chi_stat(:), data(:,:), loading(:,:), &
   prin_var(:,:), score(:,:), sig_chi_stat(:)
  ! .. Executable Statements ..
  WRITE (nag_std_out,*) &
   'Example Program Results for nag_fac_analysis_ex01'

  ! Skip heading in data file

  READ (nag_std_in,*)
  READ (nag_std_in,*) matrix, n, m
  k = m                         ! all variables are inculded in the
  ! computation
  ALLOCATE (data(n,m),score(n,k),loading(k,k),prin_var(k,3),chi_df(k), &
   chi_stat(k),sig_chi_stat(k)) ! Allocate storage

  READ (nag_std_in,*) (data(i,:),i=1,n)

  CALL nag_prin_comp(data,prin_var,matrix=matrix,score=score, &
   loading=loading,chi_stat=chi_stat,chi_df=chi_df, &
   sig_chi_stat=sig_chi_stat)

  WRITE (nag_std_out,*)
  WRITE (nag_std_out,*) &
   'Eigenvalues   Fractional   Cumulative     Chisq       Sig        DF'
  WRITE (nag_std_out,*) '               variation    variation'

  DO i = 1, k
    WRITE (nag_std_out,'(5f12.4,i8)') prin_var(i,:), chi_stat(i), &
     sig_chi_stat(i), chi_df(i)
  END DO
```

```
      WRITE (nag_std_out,*)

      CALL nag_write_gen_mat(loading,format='f12.4',title='Loadings')

      WRITE (nag_std_out,*)

      CALL nag_write_gen_mat(score,format='f12.4',title= &
       'Principal component scores')

      DEALLOCATE (data,score,loading,prin_var,chi_stat,sig_chi_stat, &
       chi_df)                    ! Deallocate storage

    END PROGRAM nag_fac_analysis_ex01
```

2 Program Data

```
Example Program Data for nag_fac_analysis_ex01
V 10 3      : matrix (type of analysis), n, m
7.0 4.0 3.0
4.0 1.0 8.0
6.0 3.0 5.0
8.0 6.0 1.0
8.0 5.0 7.0
7.0 2.0 9.0
5.0 3.0 3.0
9.0 5.0 8.0
7.0 4.0 5.0
8.0 2.0 2.0  :data
```

3 Program Results

```
 Example Program Results for nag_fac_analysis_ex01

 Eigenvalues   Fractional   Cumulative     Chisq       Sig        DF
                variation    variation
      8.2739      0.6515      0.6515      8.6127      0.1255       5
      3.6761      0.2895      0.9410      4.1183      0.1276       2
      0.7499      0.0590      1.0000      0.0000      0.0000       0

 Loadings
        0.1376      0.6990     -0.7017
        0.2505      0.6609      0.7075
       -0.9583      0.2731      0.0842

 Principal component scores
        0.7171     -0.0577      0.0356
       -1.2681     -0.9625      0.1701
       -0.0511     -0.3290      0.0898
        1.5688      0.4338      0.2172
       -0.4313      0.7597      0.1497
       -1.3664      0.0479     -0.2677
        0.5419     -0.7440      0.2676
       -0.7048      1.0837     -0.0561
        0.0783      0.1243      0.0917
        0.9155     -0.3563     -0.6980
```

References

[1] Cooley W C and Lohnes P R (1971) *Multivariate Data Analysis* Wiley

[2] Gnanadesikan R (1977) *Methods for Statistical Data Analysis of Multivariate Observations* Wiley

[3] Hammarling S (1985) The singular value decomposition in multivariate statistics *ACM Signum Newsletter* **20(3)** 2–25

[4] Kendall M G and Stuart A (1976) *Advanced Theory of Statistics, Vol 3* Griffin

[5] Krzanowski W J (1988) *Principles of Multivariate Analysis* Oxford University Press

[6] Morrison D F (1967) *Multivariate Statistical Methods* McGraw-Hill