Module 29.1: nag_tsa_identify
Time Series Analysis – Identification

`nag_tsa_identify` contains procedures for analysing time series data.

Contents

Introduction

1 Sample Autocorrelations

If a time series is stationary then the correlations ρ_k between terms x_t, x_{t+k} separated by lag k, give an adequate description of the statistical structure and are estimated by the sample autocorrelation function (acf) r_k, for $k = 1, 2, \ldots, m$, where m is the maximum lag. In the case of a random process, with zero mean and independent identically distributed normal errors, it can be shown (Moran and Murphy [4]) that the estimated acf ($\hat{\rho}(k)$) has variance given by

$$\mathrm{var}(\hat{\rho}(k)) = \frac{n-k}{n(n+2)}.$$

2 Partial Autocorrelations

The coefficients of the partial autocorrelation function (pacf) $\phi_{k,k}$, for $k = 1, 2, \ldots,$ measure the correlation between x_t and x_{t+k} conditional upon the intermediate values $x_{t+1}, x_{t+2}, \ldots, x_{t+k-1}$. The corresponding sample values $\hat{\phi}_{k,k}$ give further assistance in the selection of the model. In the case of a random process, with zero mean and independent identically distributed normal errors, it can be shown (Wei [5]) that the estimated pacf ($\hat{\phi}_{k,k}$) has a variance given by

$$\mathrm{var}(\hat{\phi}_{k,k}) \simeq \frac{1}{n}.$$

If x_{t+k} is regressed on its k lagged variables $x_{t+k-1}, x_{t+k-2}, \ldots, x_t$, then

$$x_{t+k} = \phi_{k,1}x_{t+k-1} + \phi_{k,2}x_{t+k-2} + \cdots + \phi_{k,k}x_t + e_{t+k}$$

where $\phi_{k,i}$ denotes the ith regression coefficient and e_{t+k} is an error term uncorrelated with x_{t+k-j} for $j \geq 1$. The partial autocorrelation coefficient between x_t and x_{t+k} is then the regression coefficient associated with x_t.

Procedure: nag_tsa_acf

1 Description

nag_tsa_acf computes the sample autocorrelation function of a time series. It also computes the sample mean, sample variance and a statistic which may be used to test the hypothesis that the true autocorrelation function is zero.

2 Usage

```
USE nag_tsa_identify
CALL nag_tsa_acf(ts, acf  [, 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.

$n > 1$ — the number of values in the time series

$0 < na < n$ — the number of autocorrelation coefficients required

3.1 Mandatory Arguments

ts(n) — real(kind=*wp*), intent(in)

Input: ts(i) must contain the ith value of the time series.

acf(na) — real(kind=*wp*), intent(out)

Output: acf(i) contains the autocorrelation coefficient, r_i, relating to lag i, for $i = 1, 2, \ldots, na$.

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.

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

Output: contains the mean of the input time series ts.

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

Output: contains the variance of the input time series ts.

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

Output: contains the value which can be used to test the hypothesis that the true autocorrelation function is zero.

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

Output: contains the probability, obtained using chi_stat and the upper tail of a χ^2_{na}-distribution, that the autocorrelation function is zero.

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
302	An array argument has an invalid shape.
303	Array arguments have inconsistent shapes.

Failures (error%level = 2):

error%code	Description
201	On entry all the values in the array `ts` are practically identical. The arguments `acf`, `var`, `chi_stat` and `sig_chi_stat` have been set to zero.

5 Examples of Usage

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

6 Further Comments

6.1 Mathematical Background

Let the data consist of n time series, x_i, $i = 1, 2, \ldots, n$.

- The sample mean is defined by:

$$\texttt{mean} = \bar{x} = \sum_{i=1}^{n} x_i / n.$$

- The sample variance (for $n \geq 2$) is defined by:

$$\texttt{var} = s^2 = \sum_{i=1}^{n} (x_i - \bar{x})^2 / (n-1).$$

- The sample autocorrelation coefficient of lag k for $k = 1, 2, \ldots, na$ and $0 < na < n$ is defined by:

$$r_k = \sum_{i=1}^{n-k} (x_i - \bar{x})(x_{i+k} - \bar{x}) / \sum_{i=1}^{n} (x_i - \bar{x})^2$$

  (see page 496 of Box and Jenkins [1] for more details).

- The χ^2-hypothesis test parameter is defined by:

$$\texttt{chi\_stat} = n \sum_{i=1}^{na} r_k^2.$$

  This parameter could be used to test the hypothesis that the autocorrelation function is identically zero.

  If n is large and na is much smaller than n, `chi_stat` has a χ^2_{na}-distribution under the hypothesis of a zero autocorrelation function.

6.2 Accuracy

The computations are believed to be stable.

6.3 Timing

The time taken is approximately proportional to $n \times na$.

Procedure: nag_tsa_pacf

1 Description

nag_tsa_pacf computes the partial autocorrelation coefficients given a set of autocorrelation coefficients. It also calculates the predictor error variance ratios for increasing order of finite lag autoregressive predictor, and the autoregressive parameters associated with the predictor of maximum order.

2 Usage

```
USE nag_tsa_identify
CALL nag_tsa_pacf(acf, pacf  [, 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.

$n > 0$ — the number of supplied autocorrelation coefficients

$0 < np \leq n$ — the number of partial autocorrelation coefficients required

3.1 Mandatory Arguments

acf(n) — real(kind=wp), intent(in)

Input: acf(i) contains the autocorrelation coefficient, r_i, relating to lag i, $i = 1, 2, \ldots, n$.

pacf(np) — real(kind=wp), intent(out)

Output: pacf(l) contains the calculated partial autocorrelation coefficient $(p_{l,l})$ at lag l, for $l = 1, 2, \ldots,$num_val.

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.

num_val — integer, intent(out), optional

Output: contains the number of valid values.

ar(np) — real(kind=wp), intent(out), optional

Output: the autoregressive parameters of maximum order, i.e., $p_{L,j}$, where $L =$ num_val and $j = 1, 2, \ldots,$num_val.

pred_err(np) — real(kind=wp), intent(out), optional

Output: pred_err(l) contains the predictor error variance ratio, ν_l, $l = 1, 2, \ldots,$ num_val.

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**
302	An array argument has an invalid shape.
303	Array arguments have inconsistent shapes.
320	The procedure was unable to allocate enough memory.

Warnings (error%level = 1):

error%code	**Description**
101	Recursion has been prematurely terminated. The supplied autocorrelation coefficients do not form a positive-definite sequence. The optional argument `num_val` returns the number of valid values computed for `pacf` and the optional arguments `ar` and `pred_err`.

5 Examples of Usage

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

6 Further Comments

6.1 Algorithmic Detail

The parameters are determined from the autocorrelation coefficients by the Yule–Walker equations

$$r_i = p_{l,1}r_{i-1} + p_{l,2}r_{i-2} + \cdots + p_{l,l}r_{i-l}, \ \ i = 1, 2, \ldots, l$$

taking $r_j = r_{|j|}$ when $j < 0$, and $r_0 = 1$.

The predictor error variance ratio $v_l = \operatorname{var}(e_{l,t})/\operatorname{var}(x_t)$ is defined by

$$v_l = 1 - p_{l,1}r_1 - p_{l,2}r_2 - \cdots - p_{l,l}r_l.$$

The above sets of equations are solved by a recursive method (the Durbin–Levinson algorithm). The recursive cycle applied for $l = 1, 2, \ldots, (L-1)$, where L is the number of partial autocorrelation coefficients required, is initialized by setting $p_{1,1} = r_1$ and $v_1 = 1 - r_1^2$. Then

$$\begin{aligned} p_{l+1,l+1} &= (r_{l+1} - p_{l,1}r_l - p_{l,2}r_{l-1} - \cdots - p_{l,l}r_1)/v_l \\ p_{l+1,j} &= p_{l,j} - p_{l+1,l+1}p_{l,l+1-j}, \ j = 1, 2, \ldots, l \\ v_{l+1} &= v_l(1 - p_{l+1,l+1})(1 + p_{l+1,l+1}). \end{aligned}$$

If the condition $|p_{l,l}| \geq 1$ occurs, say when $l = l_0$, it indicates that the supplied autocorrelation coefficients do not form a positive-definite sequence, and the recursion is not continued. The autoregressive parameters are overwritten at each recursive step, so that upon completion the only available values are $p_{L,j}$, for $j = 1, 2, \ldots, L$ or $p_{l_0-1,j}$ if the recursion has been prematurely halted.

6.2 Accuracy

The computations are believed to be stable.

6.3 Timing

The time taken is approximately proportional to $(\texttt{num\_val})^2$.

Example 1: Analysis of Sunspot Data

This example program evaluates the first 30 autocorrelations and partial autocorrelations for the Wolfer sunspot numbers from 1700 to 1983. A square root transformation is applied to the data.

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_tsa_identify_ex01

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

  ! .. Use Statements ..
  USE nag_tsa_identify, ONLY : nag_tsa_acf, nag_tsa_pacf
  USE nag_examples_io, ONLY : nag_std_in, nag_std_out
  ! .. Implicit None Statement ..
  IMPLICIT NONE
  ! .. Intrinsic Functions ..
  INTRINSIC KIND, SQRT
  ! .. Parameters ..
  INTEGER, PARAMETER :: wp = KIND(1.0D0)
  ! .. Local Scalars ..
  INTEGER :: i, n, na
  ! .. Local Arrays ..
  REAL (wp), ALLOCATABLE :: acf(:), pacf(:), ts(:)
  ! .. Executable Statements ..
  WRITE (nag_std_out,*) &
   'Example Program Results for nag_tsa_identify_ex01'

  READ (nag_std_in,*)          ! Skip heading in data file
  READ (nag_std_in,*) n, na

  ALLOCATE (ts(n),acf(na),pacf(na)) ! Allocate storage

  READ (nag_std_in,*) ts
  ts = SQRT(ts)

  CALL nag_tsa_acf(ts,acf)

  CALL nag_tsa_pacf(acf,pacf)

  WRITE (nag_std_out,*)
  WRITE (nag_std_out,*) '   lag       acf       pacf'
  WRITE (nag_std_out,*)
  DO i = 1, na
    WRITE (nag_std_out,'(I6,2X,2F10.2)') i, acf(i), pacf(i)
  END DO

  DEALLOCATE (ts,acf,pacf)     ! Deallocate storage

END PROGRAM nag_tsa_identify_ex01
```

2 Program Data

```
Example Program Data for nag_tsa_identify_ex01
 284 30                                                                : n, na
  5.0  11.0  16.0  23.0  36.0  58.0  29.0  20.0  10.0   8.0   3.0   0.0
  0.0   2.0  11.0  27.0  47.0  63.0  60.0  39.0  28.0  26.0  22.0  11.0
 21.0  40.0  78.0 122.0 103.0  73.0  47.0  35.0  11.0   5.0  16.0  34.0
 70.0  81.0 111.0 101.0  73.0  40.0  20.0  16.0   5.0  11.0  22.0  40.0
 60.0  80.9  83.4  47.7  47.8  30.7  12.2   9.6  10.2  32.4  47.6  54.0
 62.9  85.9  61.2  45.1  36.4  20.9  11.4  37.8  69.8 106.1 100.8  81.6
 66.5  34.8  30.6   7.0  19.8  92.5 154.4 125.9  84.8  68.1  38.5  22.8
 10.2  24.1  82.9 132.0 130.9 118.1  89.9  66.6  60.0  46.9  41.0  21.3
 16.0   6.4   4.1   6.8  14.5  34.0  45.0  43.1  47.5  42.2  28.1  10.1
  8.1   2.5   0.0   1.4   5.0  12.2  13.9  35.4  45.8  41.1  30.1  23.9
 15.6   6.6   4.0   1.8   8.5  16.6  36.3  49.6  64.2  67.0  70.9  47.8
 27.5   8.5  13.2  56.9 121.5 138.3 103.2  85.7  64.6  36.7  24.2  10.7
 15.0  40.1  61.5  98.5 124.7  96.3  66.6  64.5  54.1  39.0  20.6   6.7
  4.3  22.7  54.8  93.8  95.8  77.2  59.1  44.0  47.0  30.5  16.3   7.3
 37.6  74.0 139.0 111.2 101.6  66.2  44.7  17.0  11.3  12.4   3.4   6.0
 32.3  54.3  59.7  63.7  63.5  52.2  25.4  13.1   6.8   6.3   7.1  35.6
 73.0  85.1  78.0  64.0  41.8  26.2  26.7  12.1   9.5   2.7   5.0  24.4
 42.0  63.5  53.8  62.0  48.5  43.9  18.6   5.7   3.6   1.4   9.6  47.4
 57.1 103.9  80.6  63.6  37.6  26.1  14.2   5.8  16.7  44.3  63.9  69.0
 77.8  64.9  35.7  21.2  11.1   5.7   8.7  36.1  79.7 114.4 109.6  88.8
 67.8  47.5  30.6  16.3   9.6  33.2  92.6 151.6 136.3 134.7  83.9  69.4
 31.5  13.9   4.4  38.0 141.7 190.2 184.8 159.0 112.3  53.9  37.5  27.9
 10.2  15.1  47.0  93.8 105.9 105.5 104.5  66.6  68.9  38.0  34.5  15.5
 12.6  27.5  92.5 155.4  32.27 54.25 59.65 63.62                       : ts(1:n)
```

3 Program Results

```
 Example Program Results for nag_tsa_identify_ex01

    lag        acf       pacf

     1         0.81       0.81
     2         0.45      -0.62
     3         0.06      -0.16
     4        -0.25      -0.02
     5        -0.41      -0.08
     6        -0.40       0.18
     7        -0.21       0.22
     8         0.09       0.15
     9         0.40       0.29
    10         0.61       0.02
    11         0.65       0.02
    12         0.50      -0.05
    13         0.22      -0.06
    14        -0.07       0.09
    15        -0.28      -0.01
    16        -0.37      -0.09
    17        -0.35      -0.08
    18        -0.21      -0.12
    19         0.00       0.00
    20         0.21      -0.05
    21         0.35       0.05
    22         0.37      -0.02
    23         0.26      -0.11
    24         0.05      -0.08
    25        -0.16       0.02
    26        -0.31      -0.03
    27        -0.37       0.01
```

```
   28       -0.32      0.05
   29       -0.20     -0.05
   30       -0.03      0.01
```

References

[1] Box G E P and Jenkins G M (1976) *Time Series Analysis: Forecasting and Control* Holden-Day (Revised Edition)

[2] Chatfield C (1982) *The Analysis of Time Series: An Introduction* Chapman and Hall

[3] Durbin J (1960) The fitting of time series models *Rev. Inst. Internat. Stat.* **28** 233

[4] Moran M A and Murphy B J (1979) A closer look at two alternative methods of statistical discrimination *Appl. Statist.* **28** 223–232

[5] Wei W W S (1990) *Time Series Analysis: Univariate and Multivariate Methods* Addison-Wesley