

NAG Toolbox

nag_tsa_cp_pelt_user (g13nb)

1 Purpose

nag_tsa_cp_pelt_user (g13nb) detects change points in a univariate time series, that is, the time points at which some feature of the data, for example the mean, changes. Change points are detected using the PELT (Pruned Exact Linear Time) algorithm using one of a provided set of cost function.

2 Syntax

```
[tau, user, ifail] = nag_tsa_cp_pelt_user(n, beta, k, costfn, 'minss', minss,
'user', user)
[tau, user, ifail] = g13nb(n, beta, k, costfn, 'minss', minss, 'user', user)
```

3 Description

Let $y_{1:n} = \{y_j : j = 1, 2, \dots, n\}$ denote a series of data and $\tau = \{\tau_i : i = 1, 2, \dots, m\}$ denote a set of m ordered (strictly monotonic increasing) indices known as change points with $1 \leq \tau_i \leq n$ and $\tau_m = n$. For ease of notation we also define $\tau_0 = 0$. The m change points, τ , split the data into m segments, with the i th segment being of length n_i and containing $y_{\tau_{i-1}+1:\tau_i}$.

Given a user-supplied cost function, $C(y_{\tau_{i-1}+1:\tau_i})$ nag_tsa_cp_pelt_user (g13nb) solves

$$\underset{m, \tau}{\text{minimize}} \sum_{i=1}^m (C(y_{\tau_{i-1}+1:\tau_i}) + \beta) \quad (1)$$

where β is a penalty term used to control the number of change points. This minimization is performed using the PELT algorithm of Killick *et al.* (2012). The PELT algorithm is guaranteed to return the optimal solution to (1) if there exists a constant K such that

$$C(y_{(u+1):v}) + C(y_{(v+1):w}) + K \leq C(y_{(u+1):w}) \quad (2)$$

for all $u < v < w$

4 References

Chen J and Gupta A K (2010) *Parametric Statistical Change Point Analysis With Applications to Genetics Medicine and Finance Second Edition* Birkhäuser

Killick R, Fearnhead P and Eckley I A (2012) Optimal detection of changepoints with a linear computational cost *Journal of the American Statistical Association* **107:500** 1590–1598

5 Parameters

5.1 Compulsory Input Parameters

1: **n** – INTEGER

n , the length of the time series.

Constraint: $n \geq 2$.

2: **beta** – REAL (KIND=nag_wp)

β , the penalty term.

There are a number of standard ways of setting β , including:

SIC or BIC

$$\beta = p \times \log(n)$$

AIC

$$\beta = 2p$$

Hannan-Quinn

$$\beta = 2p \times \log(\log(n))$$

where p is the number of parameters being treated as estimated in each segment. The value of p will depend on the cost function being used.

If no penalty is required then set $\beta = 0$. Generally, the smaller the value of β the larger the number of suggested change points.

- 3: **k** – REAL (KIND=nag_wp)

K , the constant value that satisfies equation (2). If K exists, it is unlikely to be unique in such cases, it is recommended that the largest value of K , that satisfies equation (2), is chosen. No check is made that K is the correct value for the supplied cost function.

- 4: **costfn** – SUBROUTINE, supplied by the user.

The cost function, C . **costfn** must calculate a vector of costs for a number of segments.

```
[c, user, info] = costfn(ts, r, user, info)
```

Input Parameters

- 1: **ts** – INTEGER

A reference time point.

- 2: **r(nr)** – INTEGER array

Time points which, along with **ts**, define the segments being considered, $0 \leq \mathbf{r}(i) \leq n$ for $i = 1, 2, \dots, nr$.

- 3: **user** – INTEGER array

costfn is called from nag_tsa_cp_pelt_user (g13nb) with the object supplied to nag_tsa_cp_pelt_user (g13nb).

- 4: **info** – INTEGER

info = 0.

Output Parameters

- 1: **c(nr)** – REAL (KIND=nag_wp) array

The cost function, C , with

$$\mathbf{c}(i) = \begin{cases} C(y_{r_i:t}) & \text{if } t > r_i, \\ C(y_{t:r_i}) & \text{otherwise.} \end{cases}$$

where $t = \mathbf{ts}$ and $r_i = \mathbf{r}(i)$.

It should be noted that if $t > r_i$ for any value of i then it will be true for all values of i . Therefore the inequality need only be tested once per call to **costfn**.

- 2: **user** – INTEGER array

3: **info** – INTEGER

Set **info** to a nonzero value if you wish nag_tsa_cp_pelt_user (g13nb) to terminate with **ifail** = 51.

5.2 Optional Input Parameters

1: **minss** – INTEGER

Default: 2

The minimum distance between two change points, that is $\tau_i - \tau_{i-1} \geq \text{minss}$.

Constraint: **minss** ≥ 2 .

2: **user** – INTEGER array

user is not used by nag_tsa_cp_pelt_user (g13nb), but is passed to **costfn**. Note that for large objects it may be more efficient to use a global variable which is accessible from the m-files than to use **user**.

5.3 Output Parameters

1: **tau**(*ntau*) – INTEGER array

The dimension of the array **tau** will be *ntau*

The location of the change points. The *i*th segment is defined by $y_{(\tau_{i-1}+1)}$ to y_{τ_i} , where $\tau_0 = 0$ and $\tau_i = \text{tau}(i)$, $1 \leq i \leq m$.

2: **user** – INTEGER array

3: **ifail** – INTEGER

ifail = 0 unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 11

Constraint: **n** ≥ 2 .

ifail = 31

Constraint: **minss** ≥ 2 .

ifail = 51

User requested termination.

ifail = -99

An unexpected error has been triggered by this routine. Please contact NAG.

ifail = -399

Your licence key may have expired or may not have been installed correctly.

ifail = -999

Dynamic memory allocation failed.

7 Accuracy

Not applicable.

8 Further Comments

nag_tsa_cp_pelt (g13na) performs the same calculations for a cost function selected from a provided set of cost functions. If the required cost function belongs to this provided set then nag_tsa_cp_pelt (g13na) can be used without the need to provide a cost function routine.

9 Example

This example identifies changes in the scale parameter, under the assumption that the data has a gamma distribution, for a simulated dataset with 100 observations. A penalty, β of 3.6 is used and the minimum segment size is set to 3. The shape parameter is fixed at 2.1 across the whole input series.

The cost function used is

$$C(y_{\tau_{i-1}+1:\tau_i}) = 2an_i(\log S_i - \log(an_i))$$

where a is a shape parameter that is fixed for all segments and $n_i = \tau_i - \tau_{i-1} + 1$.

9.1 Program Text

```

function g13nb_example

fprintf('g13nb example results\n\n');

% Input series
y = [ 0.00; 0.78; 0.02; 0.17; 0.04; 1.23; 0.24; 1.70; 0.77; 0.06;
      0.67; 0.94; 1.99; 2.64; 2.26; 3.72; 3.14; 2.28; 3.78; 0.83;
      2.80; 1.66; 1.93; 2.71; 2.97; 3.04; 2.29; 3.71; 1.69; 2.76;
      1.96; 3.17; 1.04; 1.50; 1.12; 1.11; 1.00; 1.84; 1.78; 2.39;
      1.85; 0.62; 2.16; 0.78; 1.70; 0.63; 1.79; 1.21; 2.20; 1.34;
      0.04; 0.14; 2.78; 1.83; 0.98; 0.19; 0.57; 1.41; 2.05; 1.17;
      0.44; 2.32; 0.67; 0.73; 1.17; 0.34; 2.95; 1.08; 2.16; 2.27;
      0.14; 0.24; 0.27; 1.71; 0.04; 1.03; 0.12; 0.67; 1.15; 1.10;
      1.37; 0.59; 0.44; 0.63; 0.06; 0.62; 0.39; 2.63; 1.63; 0.42;
      0.73; 0.85; 0.26; 0.48; 0.26; 1.77; 1.53; 1.39; 1.68; 0.43];

% The cost function is a function of the sum of Y, so for
% efficiency we will calculate the cumulative sum
% It should be noted that this may introduce some rounding issues
% with very extreme data, we also pre-pend a value of 0
csy = [0.0; cumsum(y)];

% Shape parameter used in the cost function
a = 2.1;

% The value of K is defined by the cost function being used
% in this example a value of 0.0 is the required value
k = 0;

% The cumulative sum of the input series and shape parameter
% constitute the information that needs to be passed to the
% costfun, so pack them together into a cell array which will
% get passed through the NAG function
user = {csy; a};

% Length of the input series
n = nag_int(numel(y));

% Penalty term
beta = 3.4;

% Drop small regions
minss = nag_int(3);

```

```
[tau] = g13nb( ...
    n, beta, k, @costfn, 'minss', minss, 'user', user);

% Print the results
fprintf(' -- Change Points --\n');
fprintf(' Number      Position\n');
fprintf(' ======\n');
for i = 1:numel(tau)
    fprintf(' %4d      %6d\n', i, tau(i));
end

% Plot the results
fig1 = figure;

% Plot the original series
plot(y,'Color','red');

% Mark the change points, drop the last one as it is always
% at the end of the series
xpos = transpose(double(tau(1:end-1))*ones(1,2));
ypos = diag(ylim)*ones(2,numel(tau)-1);
line(xpos,ypos,'Color','black');

% Add labels and titles
title({'{\bf g13nb Example Plot}', ...
    'Simulated time series and the corresponding changes in scale b', ...
    'assuming y ~ Ga(2.1,b)'});
xlabel('{\bf Time}');
ylabel('{\bf Value}');

function [c,user,info] = costfn(ts, r, user, info)
    % Cost function, C. This cost function is based on the likelihood of
    % the gamma distribution
    csy = user{1};
    a = user{2};

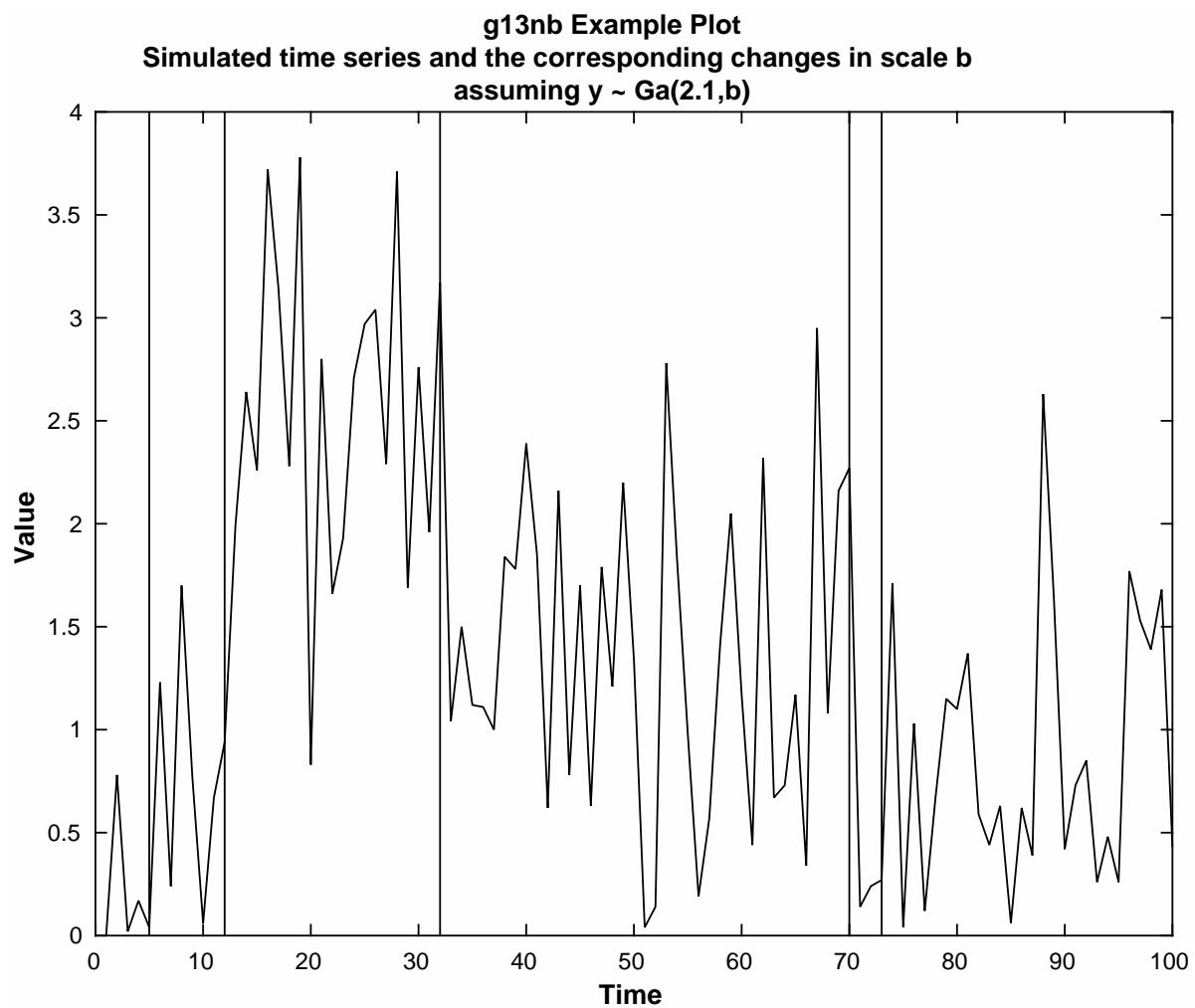
    % Only need to test which way around ts and r are once
    if (ts < r(1))
        si = csy(r+1) - csy(ts+1);
        dn = double(r - ts);
    else
        si = csy(ts+1) - csy(r+1);
        dn = double(ts - r);
    end
    c = (2*dn*a) .* (log(si) - log(dn*a));

    % Set info nonzero to terminate execution for any reason
    info = nag_int(0);
```

9.2 Program Results

g13nb example results

```
-- Change Points --
Number      Position
=====
1          5
2         12
3         32
4         70
5         73
6        100
```



This example plot shows the original data series and the estimated change points.
