DE GRUYTER OPEN

DOI: 10.1515/bile-2017-0001

Biometrical Letters
Vol. 54 (2017), No. 1, 1 - 24

An algorithm for a new method of change-point analysis in the independent Poisson sequence

Chihiro Hirotsu[1], Harukazu Tsuruta[2]

[1]Collaborative Research Center, Meisei University, 2 – 1 – 1, Hodokubo, Hino–city, Tokyo 191 – 8506, Japan, e-mail: hirotsu@ge.meisei – u.ac.jp
[2]Tokyo Metropolitan Institute of Gerontology, 35 –2 Sakae –cho, Itabashi–ku, Tokyo 173 –0015, Japan

SUMMARY

Step change-point and slope change-point models in the independent Poisson sequence are developed based on accumulated and doubly-accumulated statistics. The method for the step change-point model developed in Section 2 is an alternative to the likelihood ratio test of Worsley (1986) and the algorithm for *p*-value calculation based on the first-order Markov property is the same as that given there. Different algorithms for the non-null distribution and inference on the change-point itself are, however, newly developed and a Pascal program is given in the Appendix. These methods are extended to the slope change-point model in Section 3. The approach is essentially the same as that of Section 2 but the algorithm is now based on the second-order Markov property and becomes a little more complicated. The Pascal program related to the slope change-point model is supported on the website, URL: https://corec.meisei-u.ac.jp/labs/hirotsu/.

Key words: Convexity hypothesis; Markov property; Monotone hypothesis; Slope change-point model; Step change-point model

1. Introduction

Spontaneous reporting of adverse events due to a drug is collected daily, weekly or monthly by a medical organization such as PMDA (Pharmaceutical and Medical Device Agency of Japan), see Table 1, for example. These data are the number of events reported per a month from November 2003 to May 2010 at PMDA ; an independent Poisson sequence is assumed. It is important at PMDA to detect a significant change of time series in as short a time as possible.

Table 1. Spontaneous reporting of adverse events per month at PMDA

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
y_k	1	4	1	1	1	1	3	0	4	1	3	0	2	4	3
k	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
y_k	3	2	4	1	4	1	4	2	1	2	2	1	0	1	5
k	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
y_k	1	4	1	4	2	3	7	3	3	4	1	5	4	5	6
k	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
y_k	2	4	9	3	4	1	1	6	3	5	8	1	1	6	3
k	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
y_k	3	1	2	3	1	3	4	3	3	5	2	2	0	4	4
k	76	77	78	79											
y_k	4	2	2	4											

According to Worsley (1986) we consider a fixed segment of a Poisson sequence but the same statistic can be applied also sequentially with a slight modification. For such data, detecting an increasing tendency on the whole is of interest as well as detecting a step change-point. Then the maximal standardized accumulated statistic, max acc. $t1$, has been shown to be an appropriate statistic to detect both an increasing tendency and the change-point simultaneously, see Hirotsu and Marumo (2002) and Hirotsu (2013). It is based on the accumulated efficient score and belongs to the complete class of tests for the monotone alternative in the means of a Poisson sequence (Hirotsu, 1982). It is also asymptotically equivalent to the likelihood ratio test statistic to detect a step change-point in a Poisson sequence given by Worsley (1986). In the change-point analysis there are two parameters of interest, one is the change in mean and the other is the change-point itself. The algorithm for calculating p-values for testing a change is essentially the same as given in Worsley (1986) but the algorithm for the p-value for testing the null hypothesis of change-point itself is novel.

An approach to the step change-point is nicely extended to the slope change-point model and max acc. $t2$ has been proposed in Hirotsu et al. (2016). The probability calculations become a little harder now based on the second-order Markov property but the underlying idea is the same.

Algorithms related to a step change-point are given in Chapter 2 and the related Pascal program is given in the Appendix. The Pascal program related to the slope change-point model is given at URL: https://corec.meisei-u.ac.jp/labs/hirotsu/.

2. Monotone Hypothesis and Step Change-point Model

2.1. Model and Basic Idea

We assume a sequence of variables y_i which are distributed independently as Poisson random variables with distribution $e^{-\Lambda_i}(\Lambda_i)^{y_i}/y_i!$ of mean $\Lambda_i, i = 1, \dots, a$. Then for testing the null hypothesis

$$H_{m0}: \Lambda_1 = \cdots = \Lambda_a \tag{1}$$

against a monotone alternative

$$H_{m1}: \Lambda_1 \leq \cdots \leq \Lambda_a, \textit{ with at least one inequality strong,} \tag{2}$$

max acc. $t1$ has been proposed in Hirotsu (1982) as one of the tests that belongs to the complete class of tests for the monotone hypothesis. It is defined by max acc. $t1 = \max(t_1, \cdots, t_{a-1})$, where

$$t_k(Y_k) = t_k = \left\{\left(\frac{a-k}{ak}\right)\hat{\Lambda}\right\}^{-1/2}\left(\hat{\Lambda} - \frac{Y_k}{k}\right), k = 1, \dots, a-1, \tag{3}$$

with $Y_k = y_1 + \cdots + y_k,\ \hat{\Lambda} = Y_a/a$.

The statistic Y_k is the accumulated efficient score since $\boldsymbol{y} = (y_1, \dots, y_k)'$ is an efficient score vector with respect to $\Lambda = (\Lambda_1, \cdots, \Lambda_a)'$, where throughout this paper a prime implies a transpose of a vector or a matrix. Then $t_k(Y_k)$ is the standardized version of Y_k under the null hypothesis H_{m0}. The accumulated statistic Y_k is shown to be also an efficient score with respect to parameter Δ in the step change-point model,

$$M_{mk}: \begin{cases} \theta_i = log\ \Lambda_i = \theta, i = 1, \cdots, k, \\ \\ \theta_i = log\ \Lambda_i = \theta + \Delta, i = k+1, \dots, a. \end{cases} \tag{4}$$

Therefore the max acc. $t1$ is an appropriate test statistic also for the null hypothesis

$$H_{\Delta}: \Delta = 0, \text{ for all unknown } k, \tag{5}$$

in change-point model (4). It should be noted that the model M_{mk} is a typical example of the monotone hypothesis satisfying H_{m1}. On the contrary, every monotone contrast $\boldsymbol{c} = (c_1, \cdots, c_a)'$, $\boldsymbol{c}'\boldsymbol{j} = 0, c_1 \leq \cdots \leq c_a, \boldsymbol{j} = (1, \cdots, 1)'$ can be expressed by a unique and positive linear combination of step change-point contrasts $(k-a, \cdots, k-a, k, \cdots, k)$ with the first k elements $k-a$ and the last $a-k$ elements $k, k = 1, \ldots, a-1$,thus suggesting a close relationship between the monotone hypothesis and the step change-point model (Hirotsu and Marumo, 2002). The null hypotheses H_{m0} (1) and H_{Δ} (5) are of course equivalent.

For the change-point model M_{mk}(4) we are interested also in testing the null hypothesis that $K+1$ is the change-point,

$$H_0^{K+1}: k+1 = K+1, \tag{6}$$

against the alternative hypothesis

$$H_1^{K+1}: k+1 \neq K+1 \tag{7}$$

asserting $K+1$ not to be a change-point.

2.2. An Algorithm for Calculation of p-value under the Null Model H_{m0} or H_{Δ}

First define the conditional probability given Y_k as $F_k(Y_k, t_0) = \Pr(t_1 < t_0, \cdots, t_k < t_0 | Y_k), k = 1, \ldots, a,$ where $t_a(Y_a)$ is defined to be $-\infty$. Then we have a recursion formula:

$$\begin{aligned} F_{k+1}(Y_{k+1},\ t_0) &= Pr(t_1 < t_0, \cdots, t_k < t_0, t_{k+1} < t_0 | Y_{k+1}) \\ &= \sum_{Y_k} Pr(t_1 < t_0, \cdots, t_k < t_0, t_{k+1} < t_0 | Y_k, Y_{k+1})\, f_k(Y_k | Y_{k+1}) \qquad (8) \\ &= \begin{cases} \sum_{Y_k} F_k(Y_k,\ t_0)\, f_k(Y_k | Y_{k+1}) & if\ t_{k+1}(Y_{k+1}) < t_0\,, \\ 0, & otherwise, \end{cases} \qquad (9) \end{aligned}$$

where $f_k(Y_k|Y_{k+1})$ is a conditional distribution of Y_k given Y_{k+1} and the range of Y_k in the summation is obviously restricted to $0 \le Y_k \le Y_{k+1}$. In this case the conditional distribution is known to be a binomial distribution

$$f_k(Y_k|Y_{k+1}) = \binom{Y_{k+1}}{Y_k}\left(\frac{k}{k+1}\right)^{Y_k}\left(\frac{1}{k+1}\right)^{Y_{k+1}-Y_k}.$$

Equation (8) is due to the law of total probability and equation (9) is due to Markov property of the sequence $\{Y_k\}$. Finally we have the p-value for the observed maximum t_0 at the final step as $p = 1 - F_a(Y_a, t_0)$. It should be noted that the procedure converts the multiple summation into the repetition of a single summation, so that the calculation is feasible for large a. The Pascal program given in the Appendix can be applied to the p-value calculation.

Example 1. Change-point analysis of Table 1. Max acc. $t1$ is applied with the estimate $\hat{\Lambda}$ =2.835. The observed maximum is max $t_k(Y_k) = 3.497$ at April 2006 ($k = 29$) with right one-sided p-value 0.0096 by the recursion formula (9). It should be noted that large $t_k(Y_k)$ suggests a shift of mean between k and $k+1$; we call this a change at $k+1$. Therefore, max acc. $t1$ suggests a shift of mean at $k+1 = 30$. Then it is interesting to confirm whether it changed to a decreasing tendency at some point after that time ($k+1 = 30$). For this purpose, max acc. $t2$ of Section 3.2 can be applied.

2.3. Power Calculation

We fix the change-point model M_{mk} (4) at k as one of the monotone hypotheses and calculate the power of max acc. $t1$ as a function of Δ. Essentially the same recursion formula for the power is obtained as from the p-value calculation except that the conditional distribution is now $f_k^*(Y_k|Y_{k+1})$. It is then an easy calculation to obtain the factorization of the joint conditional distribution given Y_a under model M_{mk} (4) as

$$G(\boldsymbol{y} \mid Y_a, \Delta) = \prod_{i=1}^{a-1} f_i^*(Y_i|Y_{i+1}),$$

where

$$\begin{cases} f_i^*(Y_i|Y_{i+1}) = \binom{Y_{i+1}}{Y_i}\left(\frac{i}{i+1}\right)^{Y_i}\left(\frac{1}{i+1}\right)^{Y_{i+1}-Y_i}, i = 1, \dots, k-1, \\ f_i^*(Y_i|Y_{i+1}) = C_{i+1}^{-1}(Y_{i+1}, \Delta)\binom{Y_{i+1}}{Y_i}\left(\frac{i}{i+1}\right)^{Y_i}\left(\frac{1}{i+1}\right)^{Y_{i+1}-Y_i} e^{-Y_i\Delta}, i = k, \\ f_i^*(Y_i|Y_{i+1}) = C_{i+1}^{-1}(Y_{i+1}, \Delta)C_i(Y_i, \Delta)\binom{Y_{i+1}}{Y_i}\left(\frac{i}{i+1}\right)^{Y_i}\left(\frac{1}{i+1}\right)^{Y_{i+1}-Y_i}, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = k+1, \dots, a-1 \end{cases}$$

with $C_i(Y_i, \Delta), k+1 \le i \le a$ as normalizing constants. The conditional distribution $f_i^*(Y_i|Y_{i+1})$ reduces to the binomial distribution $f_i(Y_i|Y_{i+1})$ under H_Δ. The coefficients of the conditional distribution are determined recursively. For the recursion formula, however, the simpler form

$$\begin{cases} f_i^*(Y_i|Y_{i+1}) = C_{i+1}^{-1}(Y_{i+1})C_i(Y_i)\{(Y_{i+1}-Y_i)!\}^{-1}, i = 1, \dots, k-1, \\ f_k^*(Y_k|Y_{k+1}) = C_{k+1}^{-1}(Y_{k+1}, \Delta)C_k(Y_k)e^{-Y_k\Delta}\{(Y_{k+1}-Y_k)!\}^{-1}, i = k, \\ f_i^*(Y_i|Y_{i+1}) = C_{i+1}^{-1}(Y_{i+1}, \Delta)C_i(Y_i, \Delta)\{(Y_{i+1}-Y_i)!\}^{-1}, i = k+1, \dots, a-1, \end{cases} \tag{10}$$

is made more convenient by slightly changing the definition of the coefficients C_i. Then the coefficients C_i are calculated recursively starting from $C_1 = 1/Y_1!$ by

$$\begin{cases} C_{i+1}(Y_{i+1}) = \sum_{Y_i} C_i(Y_i)\{(Y_{i+1}-Y_i)!\}^{-1}, \ i = 1, \dots, k-1, \\ C_{k+1}(Y_{k+1}, \Delta) = \sum_{Y_k} C_k(Y_k)e^{-Y_k\Delta}\{(Y_{k+1}-Y_k)!\}^{-1}, i = k, \\ C_{i+1}(Y_{i+1}, \Delta) = \sum_{Y_i} C_i(Y_i, \Delta)\{(Y_{i+1}-Y_i)!\}^{-1}, i = k+1, \dots, a-1. \end{cases} \tag{11}$$

Recursion formula (9) is valid as it is, by replacing the conditional distribution $f_i(Y_i|Y_{i+1})$ by $f_i^*(Y_i|Y_{i+1})$ and putting a critical point instead of t_0, and gives a useful method for calculating the power at given Δ. Equation (10) shows also that the accumulated statistic Y_k is the efficient score with respect to Δ for the step change-point model (4). It should be noted that this algorithm can be applied also to the p-value calculation by setting $\Delta = 0$. This formula is different from that of Worsley (1986) who considers two independent Poisson sequences on both sides of the change-point conditionally given Y_k. Our formula extends more conveniently to the slope change-point model in Chapter 3. A Pascal program given in the Appendix can be applied both to the power and $p-$value calculation.

2.4. Inference on the Change-point

In Example 1 we are also interested in the confidence set for the change-point $k+1$. As usual, this is obtained as the set of $K+1$ that are not rejected at level α by the test of the null hypothesis on the change-point H_0^{K+1} (6) against the alternative hypothesis H_1^{K+1} (7) asserting $K+1$ not to be a change-point. An appropriate test statistic is again $\max_k t_k(Y_k)$ of (3) but the maximization is with respect to $k = 1, \dots, a-1, k \neq K$ and its null distribution should be defined under the null hypothesis H_0^{K+1}. This statistic is asymptotically equivalent to the likelihood ratio statistic used by Worsley (1986). In this case the null distribution contains a nuisance parameter Δ. However, according to Worsley, we can make the inference free from Δ by conditioning on the sufficient statistic Y_K under H_0^{K+1}. The conditional null distribution is most easily obtained by running the recursion formulae (9) and (11), fixing Y_K at the observed value. This is easily done by altering the inequality for restricting Y_K to the one point of the observed value at step $i = K$ in running recursion formula (11). Then, the confidence set eventually collects those $K+1$ for which $t_K(Y_K)$ is sufficiently close to the observed $\max_k t_k(Y_k)$, $k = 1, \dots, a-1$. Worsley (1986) proposed a different recursion formula considering independent Markov processes for both sides of the assumed change-point. However, our method is more convenient since the calculations of p-value, power and the confidence set of change-points are performed in the one program given in the Appendix. Our method is also systematically extended to the second order Markov sequence in Section 3.

Example 2. Example 1 continued**.** We test the null hypotheses H_0^{K+1} (6) for $K = 1, \dots, a-1$, applying the recursion formulae (9) and (11), and collect those $K+1$ with two-sided p-value larger than or equal to 0.10. Then the confidence set at confidence coefficient 0.90 is obtained as an interval $27 \leq K+1 \leq 43$.

3. Convexity Hypothesis and Slope Change-point Model

3.1. Model and Basic Idea

The idea of Section 2 is extended to the convexity hypothesis and slope change-point model almost as it is. It should be noted here that the convexity hypothesis depends on the spacing of events, whereas monotonicity is a property independent of spacing. Therefore, we consider here a general case of unequal spacing of events and denote the time or location of the ith event by x_i. Then, the convexity hypothesis was introduced in Hirotsu and Marumo (2002) as

$$H_{c1}:\ \boldsymbol{L}_a^{*\prime}\boldsymbol{\theta} \geq 0,\ \text{with at least one inequality strong,} \tag{12}$$

where $\boldsymbol{L}_a^{*\prime}$ is a second-order differential matrix, defined by

$$\boldsymbol{L}_a^{*\prime} = \begin{bmatrix} \frac{1}{x_2-x_1} & \frac{1}{x_1-x_2}+\frac{1}{x_2-x_3} & \frac{1}{x_3-x_2} & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{x_3-x_2} & \frac{1}{x_2-x_3}+\frac{1}{x_3-x_4} & \frac{1}{x_4-x_3} & 0 & \cdots & 0 \\ & & & & & \cdots & \\ 0 & 0 & 0 & \cdots & \frac{1}{x_{a-1}-x_{a-2}} & \frac{1}{x_{a-2}-x_{a-1}}+\frac{1}{x_{a-1}-x_a} & \frac{1}{x_a-x_{a-1}} \end{bmatrix}_{(a-2)\times a}.$$

The null hypothesis H_{c0} is defined by (12) with all the equalities, H_{c0}: $\boldsymbol{L}_a^{*\prime}\boldsymbol{\theta} = 0$. Corresponding to max acc. $t1$ (3) for the monotone hypothesis, the statistic max acc. $t2$ has been developed based on $(\boldsymbol{L}_a^{*\prime}\boldsymbol{L}_a^{*})^{-1}\boldsymbol{L}_a^{*\prime}\boldsymbol{y}$ by Hirotsu *et al*. (2016). It is defined by max acc. $t2 = \max(s_1^*, \cdots, s_{a-2}^*)$, where

$$s_k^* = \{S_k - E(S_k)\}/V^{1/2}(S_k)$$

is the standardized version of S_k under H_{c0} with

$$S_k = \sum_{i=1}^{k}(x_{i+1}-x_i)Y_i = (x_{k+1}-x_1)y_1 + \cdots + (x_{k+1}-x_k)y_k, \quad k = 1, \ldots, a-2. \tag{13}$$

The statistic S_k is a weighted doubly-accumulated statistic of $\boldsymbol{y} = (y_1, \cdots, y_a)'$, which reduces to

$$S_k = \sum_{i=1}^{k} Y_i = ky_1 + (k-1)y_2 + \cdots + y_k, \quad k = 1, \ldots, a-2,$$

in the case of equal spacing. The formula for calculating moments for standardization is given in Hirotsu *et al*. (2016).

It has been shown by Hirotsu and Marumo (2002) that

$$\boldsymbol{L}_a^* \ (\boldsymbol{L}_a^{*\prime}\boldsymbol{L}_a^*)^{-1} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \dots, \boldsymbol{b}_{a-2}],$$

$$\boldsymbol{b}_k = (\boldsymbol{I} - \boldsymbol{\Pi}_{\boldsymbol{B}}) \ (0, \dots, 0, x_{k+2} - x_{k+1}, \dots, x_a - x_{k+1})',$$

where $\boldsymbol{\Pi}_{\boldsymbol{B}} = \boldsymbol{B}(\boldsymbol{B}'\boldsymbol{B})^{-1}\boldsymbol{B}'$ is a projection matrix on to the column space of

$$\boldsymbol{B} = \begin{pmatrix} 1, \cdots, 1 \\ x_1, \cdots, x_a \end{pmatrix}'.$$

Then,

$$M_{ck}: \ \boldsymbol{\theta} = (\boldsymbol{B} \ \boldsymbol{b}_k)(\beta_0 \ \beta_1 \ \Delta_k)' = \boldsymbol{B}(\beta_0 \ \beta_1)' + \boldsymbol{b}_k\Delta_k$$

is a slope change-point model with change Δ_k at time point x_{k+1} for $k = 1, \dots, a-2$, where a positive Δ_k corresponds to the convexity hypothesis.

More explicitly this can be rewritten as

$$M_{ck}: \begin{cases} \theta_i = log \ \Lambda_i = \beta_0^* + \Delta_k x_{k+1} + (\beta_k^* - \Delta_k)x_i, i = 1, \dots, k+1 \\ \\ \theta_i = log \ \Lambda_i = \beta_0^* + \beta_k^* x_i, i = k+2, \dots, a. \end{cases} \tag{14}$$

in a form similar to M_{mk} (4). The null model H_{c0}: $\boldsymbol{L}_a^{*\prime}\boldsymbol{\theta} = 0$ is obviously equivalent to a linear regression model H_{c0}: $\theta_i = \beta_0 + \beta_1 x_i$. The sufficient statistics under H_{c0} are obviously

$$Y_a = y_1 + y_2 + \cdots + y_a \ \ and \ \ T_a = x_1y_1 + x_2y_2 + \cdots + x_ay_a,$$

and a similar test is constructed conditionally given $(Y_a, \ T_a)$. It is easily shown by derivation of the log likelihood function with respect to $\Delta_k$under the model M_{ck} (14) that the statistic S_k is an efficient score for testing the slope change-point model (14). Thus max acc. *t*2 is an appropriate test statistic simultaneously for the convexity and the slope change-point hypotheses.

For the change-point model M_{ck} we are also interested in testing the null hypothesis that x_{K+1} is the change-point,

$$H_0^{x_{K+1}}: x_{k+1} = x_{K+1}, \tag{15}$$

against the alternative hypothesis,

$$H_1^{x_{K+1}}: x_{k+1} \neq x_{K+1} \tag{16}$$

asserting x_{K+1} not to be a change-point.

3.2. An Algorithm for p-value Calculation under the Null Model H_{c0}

A recursion formula for the p-value has been obtained by Hirotsu *et al.* (2016) based on the second-order Markov property of the sequence $\{S_k\}$. Let us define the conditional probability

$$\begin{aligned} F_k(S_{k-1}, S_k, d) &= Pr(s_1^* < d, \dots, s_k^* < d | S_{k-1}, S_k), \\ &= Pr(S_1 < d_1^*, \dots, S_k < d_k^* | S_{k-1}, S_k), k = 2, \dots, a, \end{aligned} \tag{17}$$

where $d_k^* = E(S_k) + V^{1/2}(S_k)d, \quad k = 1, \dots, a-2$ and s_{a-1}^*, s_a^* are defined as $-\infty$, although their conditional variances are zero so that the inequality always holds. It should be noted that the recursion formula (17) is given in terms of S_k, since the distribution theory has been obtained in terms of S_k by Hirotsu *et al.* (2016). Correspondingly, d_{a-1}^* and d_a^* are set to $S_{a-1} + \delta$ and $S_a + \delta$, respectively, with δ a positive small number, where S_{a-1} and S_a are defined as $S_{a-1} = x_a Y_a - T_a$, $S_a = S_{a-1} + (x_{a+1} - x_a)Y_a$ with x_{a+1} an arbitrary number larger than x_a. It should be noted that S_{a-1} is an extension of the definition (13) to $k = a-1$, but it cannot be done for S_a without introducing a hypothetical value x_{a+1}. For notational convenience anyway, S_{a-1} and S_a are employed as conditioning variables instead of Y_a and T_a. Also, it is sometimes convenient to use S_{a-1} and Y_a as conditioning variables. The one-to-one correspondence among the set of variables (Y_a, T_a), (S_{a-1}, S_a), and (S_{a-1}, Y_a) is obvious. Then, a recursion formula for F_k is obtained as

$$F_{k+1}(S_k, S_{k+1}, d) = Pr(S_1 < d_1^*, \dots, S_k < d_k^*, S_{k+1} < d_{k+1}^* | S_k, S_{k+1})$$

$$= \sum_{S_{k-1}} Pr(S_1 < d_1^*, \dots, S_k < d_k^*, S_{k+1} < d_{k+1}^* | S_{k-1}, S_k, S_{k+1}) \times f_{k-1}(S_{k-1} | S_k, S_{k+1}) \tag{18}$$

$$= \begin{cases} Pr(S_1 < d_1^*, \dots, S_k < d_k^* | S_{k-1}, S_k) \times f_{k-1}(S_{k-1} | S_k, S_{k+1}) & if\ S_{k+1} < d_{k+1}^*, \\ 0, \quad otherwise, \end{cases} \tag{19}$$

where $f_k(S_k|S_{k+1}, S_{k+2})$ is the conditional distribution of S_k given S_{k+1} and S_{k+2}. Equation (18) is due to the law of total probability, and equation (19) is due to the second-order Markov property of S_k. Thus, essentially the recursion formula is obtained as

$$F_{k+1}(S_k, S_{k+1}, d) = \sum_{S_{k-1}} F_k(S_{k-1}, S_k, d) \times f_{k-1}(S_{k-1}|S_k, S_{k+1}).$$

There is no difficulty in extending the formula to F_a and the p-value of the observed maximum s_m^* is obtained at the final step by

$$p = 1 - F_a(S_{a-1}, S_a, d) \ at \ d = s_m^*.$$

It should be noted again that the procedure converts the multiple summation into the repetition of a single summation, so that the calculation is feasible for large a.

The argument for the recursion formula above is similar to that of Section 2.2 except that the conditioning variables become two, reflecting the second order Markov property. In this case, however, the conditional distribution $f_k(S_k|S_{k+1}, S_{k+2})$ is unknown and needs to be calculated.

3.3. Derivation and Calculation of the Conditional Distribution $f_k(S_k|S_{k+1}, S_{k+2})$

The joint conditional probability of $\boldsymbol{y}$ given $\{Y_a, \ T_a\}$ is factorized as

$$G(\boldsymbol{y}|Y_a, T_a) = \prod_{k=1}^{a-2} f_k(S_k|S_{k+1}, S_{k+2})$$

in terms of S_k due to the second-order Markov property of S_k. Then, the kth conditional distribution should be written in the form

$$f_k(S_k|S_{k+1}, S_{k+2}) = C_{k+1}^{-1}(S_{k+1}, S_{k+2}) C_k(S_k, S_{k+1})$$
$$\times \left\{ \left(\frac{S_{k+2} - S_{k+1}}{x_{k+3} - x_{k+2}} - \frac{S_{k+1} - S_k}{x_{k+2} - x_{k+1}} \right)! \right\}^{-1}, \ k = 1, \dots, a-2, \tag{20}$$

where C_{k+1} is the normalizing constant and $\left(\frac{S_{k+2}-S_{k+1}}{x_{k+3}-x_{k+2}}-\frac{S_{k+1}-S_k}{x_{k+2}-x_{k+1}}\right)$ is nothing but y_{k+2}.

The initial constant is defined as

$$C_1(S_1,S_2)=\left\{\left(\frac{S_1}{x_2-x_1}\right)!\right\}^{-1}\times\left\{\left(\frac{S_2-S_1}{x_3-x_2}-\frac{S_1}{x_2-x_1}\right)!\right\}^{-1}.$$

It should further be noted that in equation (20) the random variable S_k is also included as the conditioning variable in the normalizing constant $C_k(S_k,S_{k+1})$ of the previous step. Starting from C_1, all the C_k can be calculated recursively by the equation

$$C_{k+1}(S_{k+1},S_{k+2})=\sum_{S_k}C_k(S_k,S_{k+1})\times\left\{\left(\frac{S_{k+2}-S_{k+1}}{x_{k+3}-x_{k+2}}-\frac{S_{k+1}-S_k}{x_{k+2}-x_{k+1}}\right)!\right\}^{-1}. \quad (21)$$

Then, at the final step the overall normalizing constant $C_{a-1}(S_{a-1},S_a)$ is obtained and the distribution $G(\boldsymbol{y})$ is determined in terms of $S_k, k=1,\dots,a-2$. Note that $C_{a-1}(S_{a-1},S_a)$ is well defined, since $(S_a-S_{a-1})/(x_{a+1}-x_a)$ is simply Y_a, but in what follows the notation $C_{a-1}(S_{a-1},Y_a)$ is employed instead of $C_{a-1}(S_{a-1},S_a)$. For executing the recursion formula efficiently, several inequalities have been introduced in Hirotsu *et al*. (2016) and we give a summary in the following:

Absolute:

$$\max\{0,S_{a-1}-(x_a-x_{k+1})Y_a\}\le S_k\le\frac{x_{k+1}-x_1}{x_a-x_1}S_{a-1},\qquad k=1,\dots,a-2.$$

Relative:

$$\frac{x_{k+1}-x_1}{x_k-x_1}S_{k-1}\le S_k\le\frac{1}{x_a-x_k}\{(x_a-x_{k+1})S_{k-1}+(x_{k+1}-x_k)S_{a-1}\},\qquad k=2,\dots,a-2.$$

The absolute inequality gives restrictions on S_k in terms of S_{a-1} and Y_a, which are constants, and the relative one is useful for the bottom-up procedure to

construct conditional probabilities giving the possible range of S_k in terms of S_{k-1}. This inequality can be rewritten as

$$\frac{1}{x_a-x_{k+1}}\{(x_a-x_k)S_k-(x_{k+1}-x_k)S_{a-1}\} \le S_{k-1} \le \frac{x_k-x_1}{x_{k+1}-x_1}S_k, \quad k=2,\dots,a-2,$$

which is useful for defining the range of S_{k-1} based on S_k.

Example 3: Example 1 continued. In this example we are interested in testing a downturn tendency and therefore we perform a concavity test based on $-s_k^*$. Applying the recursion formula (19) for concavity test, a downturn is detected at October 2007 ($K+1=48$) with observed maximum $-s_m^*=2.858$ and one-sided p-value 0.0093 by the aid of recursion formula (21).

3.4. Power Calculation

We fix the change-point model M_{ck} (14) at k and calculate the power of max acc. $t2$ as a function of Δ_k. An essentially identical recursion formula is obtained as for the p-value calculation except for the conditional distribution $f_k(S_k|S_{k+1},S_{k+2})$. Now, it is an easy calculation to obtain the joint conditional distribution given (Y_a,T_a) under M_{ck} in terms of S_k as

$$G(\boldsymbol{y}|Y_a,T_a,\Delta_k)=C^{-1}(Y_a,T_a,\Delta_k)\prod_{i=1}^{a-2}\left\{\left(\frac{S_{i+2}-S_{i+1}}{x_{i+3}-x_{i+2}}-\frac{S_{i+1}-S_i}{x_{i+2}-x_{i+1}}\right)!\right\}^{-1}e^{S_k\Delta_k}. \tag{22}$$

Equation (22) shows again that the accumulated statistic S_k is the efficient score with respect to Δ_k for the slope change-point model (14). The factorization of $G(\boldsymbol{y}|Y_a,T_a,\Delta_k)$ is obtained by the same idea as equation (10) of Section 2.3, that is, $G(\boldsymbol{y}\mid Y_a,T_a,\Delta_k)=\prod_{i=1}^{a-2}f_i^*(S_i|S_{i+1},S_{i+2})$, where

$$\begin{cases} f_i^*(S_i|S_{i+1},S_{i+2})=C_{i+1}^{-1}(S_{i+1},S_{i+2})C_i(S_i,S_{i+1})\left\{\left(\frac{S_{i+2}-S_{i+1}}{x_{i+3}-x_{i+2}}-\frac{S_{i+1}-S_i}{x_{i+2}-x_{i+1}}\right)!\right\}^{-1}, \\ \qquad 1\le i\le k-1, \\ f_k^*(S_k|S_{k+1},S_{k+2})=C_{k+1}^{-1}(S_{k+1},S_{k+2},\Delta_k)C_k(S_k,S_{k+1})\left\{\left(\frac{S_{k+2}-S_{k+1}}{x_{k+3}-x_{k+2}}-\frac{S_{k+1}-S_k}{x_{k+2}-x_{i+1}}\right)!\right\}^{-1}e^{S_k\Delta_k}, \\ \qquad i=k, \\ f_i^*(S_i|S_{i+1},S_{i+2})=C_{i+1}^{-1}(S_{i+1},S_{i+2},\Delta_k)C_i(S_i,S_{i+1},\Delta_k)\left\{\left(\frac{S_{i+2}-S_{i+1}}{x_{i+3}-x_{i+2}}-\frac{S_{i+1}-S_i}{x_{i+2}-x_{i+1}}\right)!\right\}^{-1}, \\ \qquad k+1\le i\le a-2. \end{cases}$$

We can calculate the normalizing constants $C_i(S_i, S_{i+1})$ recursively just as in equation (21), noting the change of kernel for calculating $C_{k+1}(S_{k+1}, S_{k+2}, \Delta_k)$ at $i = k$. The constant $C(Y_a, T_a, \Delta_k)$ is obtained at the final step as $C_{a-1}(S_{a-1}, S_a, \Delta_k$. The initial value is the same as (21). If the change-point is at $x = x_2$, then the recursion formula starts from the second equation.

3.5. Inference on the Change-point

The confidence set of the change-point x_{k+1} is obtained as the set of x_{K+1} that are not rejected at level α by the test of the null hypothesis on the change-point $H_0^{x_{K+1}}$ (15) against the alternative hypothesis $H_1^{x_{K+1}}$ (16) asserting x_{K+1} not to be a change-point. An appropriate test statistic is again the maximal standardized statistic of S_k in Section 3.1, but the maximization is with respect to $k = 1, \ldots, a-2, k \neq K$ and its null distribution should be defined under the null hypothesis $H_0^{x_{K+1}}$. Again, to be free from the nuisance parameter Δ_K, we make a conditional inference given the sufficient statistic S_K under $H_0^{x_{K+1}}$. The conditional null distribution is most easily obtained by running the recursion formula of Section 3.4 fixing S_K at the observed value. This is just as in Section 2.4.

Example 4 Example 1 continued. In this example we are interested in testing a downturn tendency and therefore we perform a concavity test based on $-s_k^*$. Applying the recursion formula for $K = 1, \ldots, a-2$, for the concavity test, the confidence set at confidence coefficient 0.90 is obtained as an interval $35 \leq x_{K+1} \leq 58$.

4. Concluding Remarks

The calculations of p-value, power and the confidence set of step change-points are performed in one Pascal program, which is given in the Appendix. The calculations for the slope change-point model are almost parallel to those for the step change-point model but are more complicated. Therefore, the related

program becomes too long to present here but is available at URL: https://corec.meisei-u.ac.jp/labs/hirotsu/.

The authors have no conflict of interest.

REFERENCES

Hirotsu, C. (1982). Use of cumulative efficient scores for testing ordered alternatives in discrete models. Biometrika 69:567-577.

Hirotsu, C. (2013). Theory and its application of the cumulative, two-way cumulative and doubly cumulative sum statistics. Jap. J. Appl. Statist. 42:121-143. (In Japanese)

Hirotsu, C. and Marumo, K. (2002). Change point analysis as a method for isotonic inference. Scand. J. Statist. 29:125–138.

Hirotsu, C., Yamamoto, S. and Tsuruta, H. (2016). A unifying approach to the shape and change-point hypotheses in the discrete univariate exponential family. Comput. Statist. Data Anal. 97:33–46.

Worsley, K. J. (1986): Confidence regions and tests for a change-point in a sequence of exponential family of random variables. Biometrika 73:91–104.

APPENDIX

```
program Max_acc_t1_Poisson(input, output);

{$APPTYPE CONSOLE}

uses SysUtils, Math;

{ ------------------------------------------------------------------ }
{   max acc t1 for Poisson sequence (step change-point model)        }
{     1.p-value  2.power  3.confidence set                           }
{ ------------------------------------------------------------------ }
{   This program can be compiled by Delphi(R).                       }
{   The latest version and the executable program are available at    }
{      https://corec.meisei-u.ac.jp/labs/hirotsu/                    }
{ ------------------------------------------------------------------ }
{   Critical points for power calculation are available at the site.  }
{ ------------------------------------------------------------------ }
{                 Notes and Limitations                              }
{ ------------------------------------------------------------------ }
{   Global variables C1 & C2 are used to avoid stack overflow.       }
{   Maximum length of Poisson sequence is 100.                       }
{   Extended precision is used                                       }
{   & the maximum number of total frequency is 1754.                 }
{      (If you use double precision, max number is 170.)             }
{ ------------------------------------------------------------------ }
```

```
{ --- Declarations ------------------------------------------------ }

  const  max_sequence  =                100;
         max_factorial =               1754;      { max of factorial }
         epsilon       =          0.0000001;
         infinity      =  1.797693134862E308;

  type array_type        = array[0..max_sequence]  of integer;
       array_type_R      = array[0..max_sequence]  of extended;
       array_type_long_R = array[0..max_factorial] of extended;

       Ck_type = record
                   p        : extended;
                   feasible : boolean;
                 end;

       option_type = ( get_p_value, get_power, get_CI );

  var  fi,   fo  : text;
       filename  : string;
       option    : option_type;

       y         : array_type;
       C1,  C2   : array[0..max_factorial] of Ck_type; { global var }
       LFact     : array_type_long_R;

       J,
       K0,  n_option,
       a,   Ya,
       change_point   : integer;

       p,   delta,
       critical_point : extended;

  label last_line;

{ ---------------------------------------------------------------- }
     function Imin2( x1, x2 : integer ) : integer;
{ ---------------------------------------------------------------- }
     begin
       if x1 < x2 then Imin2 := x1
                  else Imin2 := x2;
     end;

{ ---------------------------------------------------------------- }
     function feasible( Y1, Y2 : integer ) : boolean;
{ ---------------------------------------------------------------- }
     var yk : integer;

     begin
       yk := Y2 - Y1;

       if (yk >= 0) then feasible := True
```

```
                    else feasible := False;
    end;

{ ------------------------------------------------------------------- }
    function fk( k,  Y1,  Y2,
                 change_point : integer;
                 delta        : extended ) : extended;
{ ------------------------------------------------------------------- }
{         NB: C1 & C2 are gloabal variables                           }
{ ------------------------------------------------------------------- }
    var yk : integer;
        p  : extended;

    begin
      yk := Y2 - Y1;

      if feasible(Y1, Y2) and ((C1[Y1].feasible and C2[Y2].feasible)) then
        p := exp( Ln(C1[Y1].p) - Ln(C2[Y2].p) - LFact[yk] )
      else
        p := 0;

      if (k=change_point) then fk := p * exp(-delta*Y1)
                          else fk := p;
    end;

{ ------------------------------------------------------------------- }
    function get_F( a, K0,
                    change_point   : integer;
                    critical_point,
                    delta          : extended;
                    y              : array_type ) : extended;
{ ------------------------------------------------------------------- }
  var k,   J,
      Y1,  Y2,   Ya,
      low_Y1, up_Y1,
      low_Y2, up_Y2,
      max_k0          : integer;

      max_t0,     t,
      lambda_hat, sum : extended;

      Yk,  low,  up   : array_type;
      E0,  V0,   t0   : array_type_R;
      Fp1, Fp2        : array_type_long_R;

{ ----  Start of get_F  ------------------------------------------- }

  begin
    for J:=0 to max_factorial do begin
      C1[J].p:=0;  C1[J].feasible:=False;  Fp1[J]:=0;
      C2[J].p:=0;  C2[J].feasible:=False;  Fp2[J]:=0;
    end;
```

```
{ --- get accumulated sum Yk's of the observed sequence y ----------- }

                    Yk[0] := 0;
    for k:=1 to a do Yk[k] := Yk[k-1] + y[k];
                    Ya    := Yk[a];

{ --- set the range of Yk's ---------------------------------------- }

    for k:=1 to (a-1) do begin
      low[k] :=  0;
      up[k]  := Ya;
    end;

    low[a] := Ya;
    up[a]  := Ya;

    if (K0>0) then begin
      low[K0] := Yk[K0];
      up[K0]  := Yk[K0];
    end;

{ --- calculate t0 based on the observed sequence ------------------- }

    if (change_point=0) and (K0<=1) then begin
      writeln( fo );
      writeln( fo, 'k,y[k],Yk[k],E[t],V[t],t[k]' );

      writeln;
      writeln( '     k     y[k]    Yk[k]     E[t]        V[t]        t[k]' );
      writeln( '-------------------------------------------------------------' );
    end;

    max_t0     := -infinity;
    max_k0     := 0;
    lambda_hat := Ya/a;

    if (change_point>0) then max_t0 := critical_point
    else begin
      for k:=1 to (a-1) do begin
        E0[k] := lambda_hat - Yk[k]/k;
        V0[k] := (1/k-1/a)*(lambda_hat);
        t0[k] := E0[k] / sqrt(V0[k]);

        if (t0[k] > max_t0) then begin
          max_t0 := t0[k];
          max_k0 := k;
        end;

        if (change_point=0) and (K0<=1) then begin
          writeln( fo,  k:8, ',',
                        y[k]:4,    ',',
                        Yk[k]:6,   ',',
                        E0[k]:20:8,',',
```

```
                    V0[k]:20:8,',',
                    t0[k]:20:8      );
        writeln(    k:6,        '  ',
                    y[k]:6,     '  ',
                    Yk[k]:6,    '  ',
                    E0[k]:10:4,'  ',
                    V0[k]:10:4,'  ',
                    t0[k]:10:4      );
      end;
    end;
  end;

  max_t0 := max_t0 - epsilon;

  if (change_point=0) and (K0<=1) then begin
    writeln( fo );
    writeln( fo,  'max_t0:,',   max_t0:10:6, ',max at K:,', max_k0:6    );
    writeln( fo );

    writeln( '-------------------------------------------------------------' );
    writeln;
    writeln( 'max_t0 =', max_t0:10:4, ' at K = ', max_k0:3 );
    writeln;
  end;

{ --- get C(1)  ------------------------------------------------- }

  for Y1 := low[1] to up[1] do begin
    C1[Y1].p        := exp( -LFact[Y1] );
    C1[Y1].feasible := True;

    if (K0=1) then t := -infinity
              else t := (lambda_hat - Y1/1) / sqrt((1/1- 1/a)*lambda_hat);

    if (t<max_t0) then Fp1[Y1] := 1
                  else Fp1[Y1] := 0;
  end;

{ --- get C(k+1) and then F(Y[k+1],t0) (k=1,..,(a-1)) --------------- }

  for k:=1 to (a-1) do begin              { --- start of loop k --- }

  { ---  Step 1 : get ranges of Y[k] and Y[k+1]  --- }

    low_Y1 := low[k];    up_Y1  := up[k];
    low_Y2 := low[k+1];  up_Y2  := up[k+1];

  { ---  Step 2 : calculate C(k+1)  --- }

    for Y2 := low_Y2 to up_Y2 do begin
      sum := 0;

      for Y1 := low_Y1 to up_Y1 do begin
```

```
          if feasible(Y1, Y2) and C1[Y1].feasible then begin
            C2[Y2].feasible := True;

            if (k=change_point) then
              sum := sum + exp( Ln(C1[Y1].p) - LFact[Y2-Y1]) * exp(-delta*Y1)
            else
              sum := sum + exp( Ln(C1[Y1].p) - LFact[Y2-Y1] );
          end;
        end;

        C2[Y2].p := sum;
      end;

  { ---  Step 3 : calculate F(Y[k+1],t0)  --- }

      for Y2 := low_Y2 to up_Y2 do begin
        if ((k+1)=a) or ((k+1)=K0) then
          t := -infinity
        else
          t := (lambda_hat - Y2/(k+1)) / sqrt((1/(k+1)- 1/a)*lambda_hat);

        if (t<max_t0) then begin
          sum := 0;

          for Y1 := low_Y1 to up_Y1 do
            if feasible(Y1, Y2) then
              sum := sum + Fp1[Y1] * fk( k, Y1, Y2, change_point, delta );

          Fp2[Y2] := sum;
        end else
          Fp2[Y2] := 0;
      end;

  { --- Step 4 : copy C(k+1) & Fp(k+1) for the next iteration --- }

      if ((k+1) < a) then
        for J:=0 to max_factorial do begin
          C1[J].p:=C2[J].p; C1[J].feasible:=C2[J].feasible; Fp1[J]:=Fp2[J];
          C2[J].p:=0;       C2[J].feasible:=False;          Fp2[J]:=0;
        end;
    end;                                        { --- end of loop k --- }

    get_F := Fp2[Ya];
  end;                                           { --- end of get_F --- }

{ ------  M A I N    P A R T  ------------------------------------- }

begin
  K0            := 0;
  delta         := 0;
  change_point  := 0;
  critical_point := -infinity;
```

```
  for J:=0 to max_sequence do y[J] := 0;

{ --- calculate log factorials ------------------------------------ }

                              Lfact[0] := 0;
  for J:=1 to max_factorial do Lfact[J] := Lfact[J-1] + Ln( J );

{ ---  read data   ------------------------------------------------- }

  write('Enter option (1.p-value 2:power 3.confidence set ==>> ');
  readln( n_option );

       if n_option=2 then option := get_power
  else if n_option=3 then option := get_CI
  else                    option := get_p_value;

  write('Enter data file name ==>> '); readln( filename );

  if FileExists( filename ) then
    begin
      assign( fi, filename);          reset(fi);
      assign( fo, filename+'.csv' );  rewrite(fo);
    end
  else
    begin
      writeln;
      writeln('Error : file NOT found. Please try it again!');
      goto last_line;
    end;

  J  := 0;
  Ya := 0;

  repeat
    J := J + 1;
    readln ( fi, y[J] );
    Ya := Ya + y[J];
  until eof(fi) or (J >= max_sequence);

  a := J;  { --- a is the length of the observed sequence y[] --- }

  if not eof(fi) then begin
    writeln('Warning: the length of the sequence is greater than max.' );
    goto last_line;
  end;

  if (Ya > max_factorial) then begin
    writeln('Warning: the sum of yk is greater than max(1754).' );
    goto last_line;
  end;

  close(fi);
```

```
  if (option=get_power) then begin
    write('Enter change point ==>> ');     readln( change_point );
    write('Enter critical point ==>> ');   readln( critical_point );
  end;

  writeln;
  writeln( 'Now calulating ........ ' );
  writeln;

{ ---  top level   ------------------------------------------------- }

  case option of
    get_p_value :
      begin
        p := 1 - get_F( a, 0, 0, 0, 0, y );

        writeln(     'p-value: ', p:10:6 );
        writeln( fo, 'p-value:,', p:10:6 );
      end;

    get_power :
      begin
        writeln( '   delta        p' );
        writeln( '  ----------------------' );
        writeln( fo, 'delta,p' );

        for J:=-200 to 200 do begin
          delta := J / 100;
          p := 1 - get_F( a, 0, change_point, critical_point, delta, y );

          writeln(     delta:8:2, '  ', p:10:6 );
          writeln( fo, delta:8:2, ',',  p:10:6 );
        end;

        writeln( '  ----------------------' );
      end;

    get_CI :
      begin
        for K0:=1 to (a-1) do begin
          p := 1 - get_F( a, K0, 0, 0, 0, y );

          if (K0=1) then begin
            writeln( '      K0        p' );
            writeln( '    ------------------------' );
            writeln( fo, 'K0,p' );
          end;

          write( K0:8, '  ', p:10:6 );
               if (p>=0.1)  then writeln( '   **' )
          else if (p>=0.05) then writeln( '   *' )
          else                   writeln;
```

```
          write( fo, K0:8, ',',  p:10:6 );
               if (p>=0.1)  then writeln( fo, ',**' )
          else if (p>=0.05) then writeln( fo, ',*' )
          else                   writeln( fo );
        end;

        writeln( '    ----------------------' );
      end;
  end;

{ ---  ending  ------------------------------------------------------ }

  close(fo);

last_line:
  writeln;
  write('Hit ENTER key !');
  readln;
end.
```

Worked Example

We give here only a short worked example since the real example of Table 1 is too lengthy.
For Examples 1 and 2 one should be recommended to visit our website.

【Data File】
1
1
1
3
3
3

【Output in the calculation of *p*-value】
Enter option (1.p-value 2:power 3.confidence set ==>> 1
Enter data file name ==>> PO_A
Now calulating .....

k	y[k]	Yk[k]	E[t]	V[t]	t[k]
1	1	1	1.0000	1.6667	0.7746
2	1	2	1.0000	0.6667	1.2247
3	1	3	1.0000	0.3333	1.7321
4	3	6	0.5000	0.1667	1.2247
5	3	9	0.2000	0.0667	0.7746

```
max_t0 =    1.7321 at K =   3
p-value:   0.147437
Hit ENTER key !
```

【Output in the calculation of confidence set of change-point】

```
Enter option (1.p-value 2:power 3.confidence set ==>> 3
Enter data file name ==>> PO_A
Now calulating .....
   k    y[k]   Yk[k]    E[t]       V[t]       t[k]
-------------------------------------------------------------
   1       1       1    1.0000     1.6667     0.7746
   2       1       2    1.0000     0.6667     1.2247
   3       1       3    1.0000     0.3333     1.7321
   4       3       6    0.5000     0.1667     1.2247
   5       3       9    0.2000     0.0667     0.7746
-------------------------------------------------------------
max_t0 =    1.7321 at K =   3
    K0        p
  ----------------------
     1    0.226435   **
     2    0.335275   **
     3    0.565521   **
     4    0.306808   **
     5    0.177867   **
  ----------------------

Hit ENTER key !
```