EASTIRS : Evolutionary Algorithm Slicing Timeseries Into Related Segments

(User Notes by Richard Forsyth, Sept 2014)

This program identifies sections of sequential data with similar values, thus chopping a series into segments. It has been written in Python3 and is released under the GNU Public License for general usage. (Current version is eastirs3.py: the trailing digit in the program name is a version number that may change if the software is revised.)

Why I Wrote this Software

I originally wrote a C version of this program in 1994. The recent Python edition is, I believe, a slight improvement. EASTIRS is a special-purpose clustering program. It tries to identify natural break-points in sequential data, thus partitioning a series into segments, or phases, with similar values.

In essence it performs constrained clustering, but does much the same job more commonly described by statisticians as change-point analysis. Finding natural subsections of temporally organized data is a task with many applications in economics, medicine, process control and other fields. Personally, I am motivated to apply it to stylochronometry, i.e. changes in an author's verbal habits over time which, if regular, could allow estimating the dates of undated works.

EASTIRS expects to be given 2 input sequences: a time-stamp variable (t) and a response variable (y). Thus it looks for break-points in a function y = f(t). Actually it isn't restricted to traditional time-series: the t-variable doesn't have to be spaced at equal intervals and it doesn't even have to be a time marker. It can be any variable that makes sense as defining an ordering on the y values. (For practical convenience, if your data has only a single data stream of y values, the program will attach the positive integers as a default t-variable.)

At present the program deals only with univariate time series, but I hope to develop and release a multivariate version eventually.

Setting Up

First you need Python3. If you don't have it already, the latest version can be downloaded and installed from the Python website: www.python.org. This is usually quite straightforward. The only snag is if you have Python2 and want to keep using it. Then you'll probably have to set up a specific command to run whichever version you use less frequently.

Next step is to unpack the eastirs.zip file. After unpacking it (into a top-level folder called "eastirs", unless you want to do quite a lot of editing), you should find the following subfolders.

datasets
op
p3
parapath

The programs are in p3. Sample data sets for testing will be found in datasets. Subfolder op is the default location for output files and parapath is a convenient place for storing parameter files, which will be explained later. In Windows, it is most convenient to install eastirs at the top level of the C:\ drive, at least to start with; otherwise you'll have to edit the sample parameter files to make sure their datfile parameters point to the correct locations.

Data Format

The program expects to read its input values from data files such as can be exported from R (R Core Team, 2013) or Excel, with a header line giving column names and using the tab character as a delimiter. Data files can also be created in a text editor such as Notepad++ (http://notepad-plus-plus.org/), preferably in utf-8 encoding.

The first five lines of the sample datafile lynxdat.dat are listed below to illustrate this format.

```
year  lynx  diff
1821  269   0
1822  321   52
1823  585   264
1824  871   286
....
```

This dataset is a well-studied time series. It records trappings of lynx in Canada from 1821 to 1934. If you examine it you will see that it consists of repeated boom and bust cycles, presumably as the lynx population expands to unsustainable levels and then crashes.

The first line names the variables (columns); subsequent lines give values for each of these 3 variables, separated by tabs.

With this example, the obvious t-variable is the year, but there are 2 possible y-variables, lynx (the number of lynx caught that year) and diff (the difference in lynx trappings from year to year). In other words, one can look for phases either in the levels themselves or in the changes of level.

Preparing a Parameter File

When you run eastirs3.py it will ask for the name of a parameter file. Below is a listing of parameter file lynx.txt which comes with the distribution in parapath.

```
comment  applying eastirs to canadian lynx trappings data :
jobname  lynx
datfile  c:\eastirs\datasets\lynxdat.dat
##targvar  lynx
##targvar  diff
##choose 1 of above.
timevar  year
varmode  vc
ntrials  65536
```

A parameter file is just a plain text file with one item per line. Each line should begin with the parameter name, then 1 or more blank spaces, then the parameter value. The following table interprets the above parameter file, line by line.

Parameter	Default value	Function
comment	[None]	This (or in fact any unrecognized parameter name, e.g. "##") can be used to insert reminders about what the file is meant to do.
jobname	eastirs	This gives the job a name. Any text string can be the value. It isn't necessary but it is useful as the jobname will be used as a prefix to the program's output files, so it can be seen that they form a group.
datfile	[None]	This should be the full file specification of a file where the input data is stored (in tab-delimited form with a header line naming the

		columns).
timevar	[None]	This gives the column name of the variable to be used as the time-stamp that defines the sequence of the y-values.
targvar	[None]	This gives the name of the column in the data file that contains the target variable, i.e. the values of the series itself.
varmode	vc	This selects which 'fitness function' (see below) is to be used. Several functions have been implemented but at present I don't recommend any except "vc", which is the default value.
ntrials	49152	EASTIRS uses an evolutionary algorithm to optimize the serial subdivision of the data, according to the fitness function selected by varmode. This parameter specifies how many evolutionary trials to make, i.e. how many gene-strings will be created during the optimization process.

In the example above, the lines beginning "##targvar" are comments. To activate either choice of target (y) variable you could simply delete the "##" at the start of one of these lines, although this isn't strictly necessary, since the program will ask for a target variable if none is specified in the parameter file. (Likewise with time-stamp variable.)

Running EASTIRS

When you run eastirs3.py it will ask for a parameter file. If this file is in the same directory as the program or in the parapath subfolder you won't have to give the full path specification, just its name (.txt extension will be presumed if no extension is given).

You should see on screen something like the listing below, which comes from a run using the spots101.txt parameter file. This runs the program on another of the supplied sample sets, mean daily sunspot numbers from 1913 to 2013 inclusive.

```
c:\myfolder>python c:\eastirs\p3\eastirs3.py

C:\eastirs\p3\eastirs3.py 3.3 Fri Sep  5 16:13:23 2014
command-line args. = 1
prepath : C:\eastirs\p3
working folder:  C:\myfolder
script usage:  python C:\eastirs\p3\eastirs3.py <parafile>
please give parameter file name : c:\eastirs\parapath\spots101
Paths to search for parameter file :
['C:\\\\parapath',  'C:\\myfolder',  '..',  '.',  'C:\\Users\\Richard\\parapath',
'C:\\Users\\Richard']
c:\eastirs\parapath spots101
trying to open : c:\eastirs\parapath\spots101.txt
c:\eastirs\parapath\spots101.txt opened for reading.
102 2
data rows = 101
data cols = 2
['midyear', 'spotnum']
Usable variables :
0 midyear
1 spotnum
Time-stamp chosen is column 0 midyear
midyear
[1913.5, 1914.5, 1915.5, 1916.5, 1917.5, 1918.5, 1919.5, 1920.5, 1921.5, 1922.5]
....
[2004.5, 2005.5, 2006.5, 2007.5, 2008.5, 2009.5, 2010.5, 2011.5, 2012.5, 2013.5]
time-series y-variable :
1 spotnum
basic stats of spotnum :
```

```
id        0
mean      63.001
std1      47.8434
std2      48.082
var1      2288.99
var2      2311.8799

[some lines omitted here to save space .... ]

C:\eastirs\p3\eastirs3.py done on Fri Sep  5 16:15:08 2014
after 78.484 seconds.
```

The only user input required here has been marked in **bold face**, i.e. the program launch command and the name of the parameter file, spots101, which is found in folder c:\eastirs\parapath\ with .txt extension (both defaults).

Interpreting the output

The listing below shows output derived from another dataset supplied with the distribution, aircraft.dat. This contains 103 examples of military aircraft used in World War II. (More details are given in aircraft.txt in the datasets subfolder). This is not a temporal sequencing problem; rather, in this case, the program is examining the relationship between the variable tip2tail (length of the aircraft in metres) and its wingspan (also in metres). Longer aircraft tend to have greater wingspans: the program is in effect looking for change-points in this relationship.

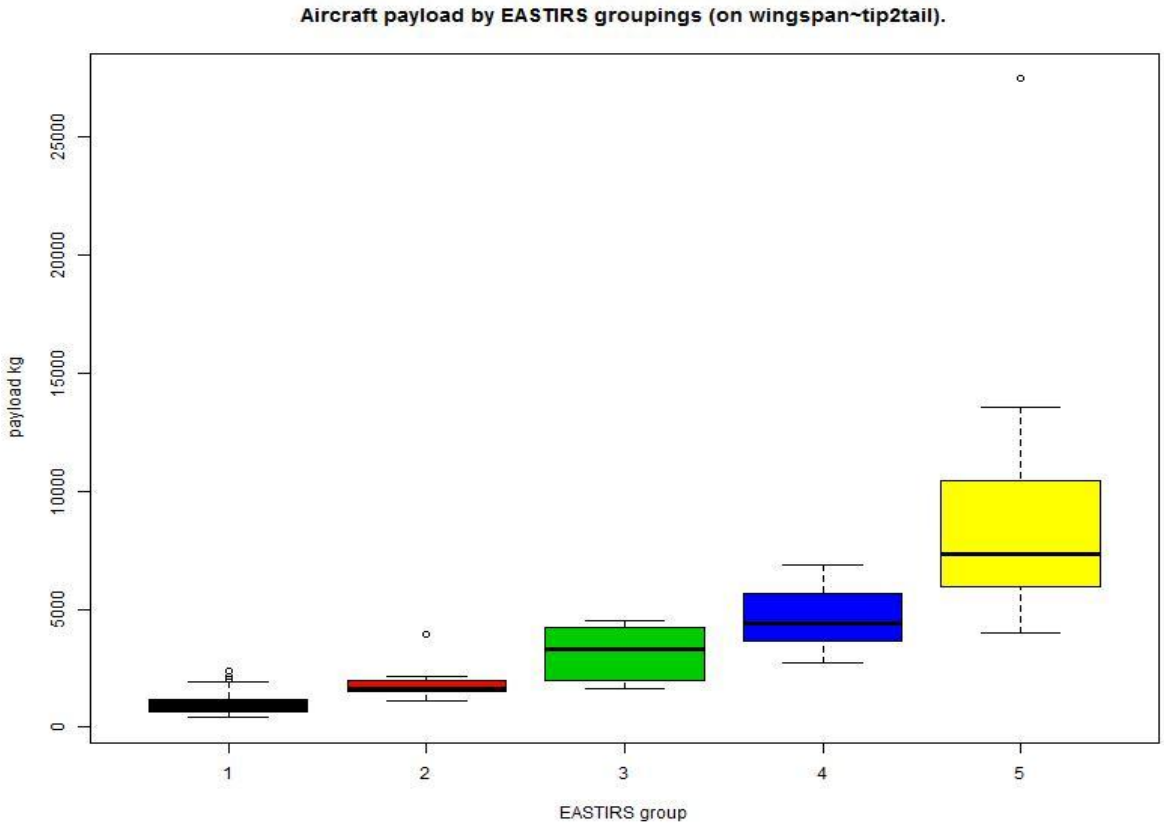
```
C:\eastirs\p3\eastirs3.py
Fri Sep  5 15:52:05 2014

Source file name : c:\eastirs\datasets\aircraft.dat
Sequence variable : tip2tail
Response variable : wingspan
wingspan mean = 16.6274757
wingspan variance = 50.7165588

Fitness mode : vc
Best score =  0.8552925

Best partition :
Rank  Case  Gene  Group      tip2tail       wingspan
   1    43     1      1        5.6900         9.3000
   2    54     0      1        6.1300         9.1800
   3    53     0      1        6.3000        10.0000
   4    46     0      1        7.5500        11.4800
   5     1     0      1        7.7800        11.0600
   6     8     0      1        7.7900        10.9700
   7    12     0      1        7.9500        11.0000
   8    82     0      1        8.0300        10.6700
   9    51     0      1        8.1500        10.3000
  10     3     0      1        8.1600        10.6200
  11    10     0      1        8.2000        10.5800
  12     5     0      1        8.2100        11.0100
  13     7     0      1        8.2500         9.7000
  14    37     0      1        8.3000        10.5000
  15    67     0      1        8.3600         9.8300
  16    49     0      1        8.4600         9.8000
  17    57     0      1        8.5000         9.2100
  18    26     0      1        8.7500         9.5000
  19     2     0      1        8.7600        10.2100
  20    91     0      1        8.7600        11.5800
  21    31     0      1        8.8400        10.4900
  22    11     0      1        8.8500        10.5800
  23    15     0      1        8.8800        12.0000
  24    50     0      1        8.9000         9.8000
  25    25     0      1        8.9200        10.8300
```

```
  26    35     0      1        9.0000         7.2000
  27    41     0      1        9.0400         9.9200
  28    18     0      1        9.0600        12.0000
  29    17     0      1        9.1600        12.0000
  30    79     0      1        9.1900        10.3700
  31    85     0      1        9.5100        11.3900
  32    76     0      1        9.5700        11.2300
  33    71     0      1        9.5800        12.1900
  34    19     0      1        9.7000        10.8000
  35    72     0      1        9.7300        12.6700
  36   100     0      1        9.8200        11.3000
  37    27     0      1        9.9200        11.2400
  38    77     0      1        9.9600        11.2300
  39    90     0      1       10.0700        12.6600
  40   103     0      1       10.1700        12.4700
  41    14     0      1       10.2000        14.3650
  42    32     0      1       10.2000        10.4900
  43    28     0      1       10.2200        11.5000
  44    92     1      2       10.2400        13.0600

[.... 43 lines omitted to save space ....]

  88     6     0      4       18.4000        24.8000
  89    30     0      4       18.5000        19.0000
  90    38     1      5       18.9000        29.2500
  91    89     0      5       19.4400        29.1300
  92    84     0      5       19.4700        31.7000
  93    21     0      5       19.6300        24.8900
  94    83     0      5       20.4700        33.5300
  95    59     0      5       21.1500        31.0900
  96    78     0      5       21.3000        25.0600
  97    70     0      5       21.3600        31.7500
  98    58     0      5       21.5000        25.6000
  99    80     0      5       22.8000        31.6200
 100    33     0      5       23.4600        30.8550
 101    74     0      5       26.0300        34.3800
 102    75     0      5       26.5900        30.2000
 103    81     0      5       30.1800        43.0500

Segment statistics :

Segment   1 at 5.6900  n =   43 :
median =   10.8000  mean =   10.8192  variance =     1.4387

Segment   2 at 10.2400  n =   14 :
median =   14.2350  mean =   14.2721  variance =     2.0334

Segment   3 at 12.1900  n =   12 :
median =   16.5100  mean =   16.6350  variance =     3.0975

Segment   4 at 13.5100  n =   20 :
median =   20.3650  mean =   20.7935  variance =     3.4605

Segment   5 at 18.9000  n =   14 :
median =   30.9725  mean =   30.8646  variance =    21.0163

Summed subgroup variance = 31.0463918
```

The listing begins by recording information about the run's settings, including the 2 variables chosen for analysis as time-stamp and response. (There are 16 variables in this dataset.) The main body of the listing (abbreviated by omitting 43 rows in the middle) gives details of the best segmentation found. This is ordered by the t-variable values and gives the corresponding value of the y-variable for each t-value. It also shows the genestring that defines the resulting segmentation and the group (numbered from 1) to which each data pair belongs. The listing concludes by giving some summary statistics of each grouping. (See Upton & Cook (2006) if uncertain about statistical terminology.)

In this example, the program has decided that there are five subgroups, from very small aircraft (43 of them, mostly fighters) to very large (14 of them, 13 being heavy bombers).

An analyst would doubtless want to explore further how these groups differ, e.g. by seeing if the x-y gradient in the first group differs from that in the last group. To enable such further analyses, the program produces 2 output files: a list file (indicated by a name ending in "_list.txt" unless specified otherwise), and a dump file (indicated by a name ending in "_dump.dat" unless specified otherwise). The list file is what has been shown above. The dump file is just the input data with an extra column appended that contains the numeric segment label for each case, in tab-delimited format with a header line. This can be read into Excel or R for further processing. The graph below is an example of the sort of post-processing that can be done (in R for this example) with a dump file.



This graph is a boxplot generated from the aircraft.dat data, with the groupings derived from EASTIRS appended (i.e. the "_dump.dat" file read back into R). It shows the distributions of payload (laden less unladen weight in kilograms) in each subgroup. Payload was not part of the grouping process, which was based on fuselage length and wingspan; it is derived from the difference of 2 other variables. Yet only 2 of these five derived groupings overlap in their middle 50% range on payload (groups 3 and 4). Thus the groupings based on one pair of variables are predictive in terms of a different pair of variables. Of course we know that longer aeroplanes with greater wing spans can normally carry heavier payloads, but that is why this is a test dataset. In general, this kind of structural relationship will be unknown, so the fact that the partitioning process reveals it can be seen as a sign that the process has value in data exploration.

How the program works

The problem of constrained clustering is peculiarly well suited to solution by an evolutionary

approach, since a clustering can be described by a binary string in which 1 signifies the start of a new segment at a particular point and 0 indicates the continuation of a segment. Bitstrings of this type are exactly what evolutionary/genetic algorithms work with most effectively. They can be chopped up and recombined or mutated very simply to produce fresh candidate solutions, thus searching the space of all possible segmentations.

The evolutionary optimizer used here can be found in the py3seal.py library, on folder p3. It is a streamlined version of a technique described in section 4.1 of Forsyth (1996).
http://www.richardsandesforsyth.net/pubs/ppsn1996.pdf

To apply evolutionary optimization to this kind of problem, a 'fitness function' must be defined. This gives a quantitative evaluation of the quality of any given segmentation. I have experimented with several fitness functions but so far the only one I consider satisfactory is selected by giving "vc" as the value of the varmode parameter. (This is the default if no varmode parameter is supplied.)

This fitness function (which the program tries to maximize) yields a numerical value which can be summarized as

1.0 - ((dev1+dev2) / max(1,n-g)) / var2

where var2 is the overall variance of the y-variable; dev1 is the summed squared deviation of each y-value in each group (segment) from its local mean; dev2 is the sum of the squared deviations of the arithmetic means of each group from the overall mean of the y-variable; n is the number of data points in the series, and g is the number of segments in the candidate solution.

The underlying idea is that dev1 quantifies the lack of homogeneity in the groups, while dev2 quantifies the lack of parsimony in the segmentation. Both these are treated as costs, then added together and divided by a number corresponding to the degrees of freedom of the partitioning. This result is then scaled by the overall variance and subtracted from 1.0 since the evolutionary function is a maximizer.

Acknowledgements

Thank you for reading this far. :-)

References

Forsyth, R.S. (1996). IOGA: an instance-oriented genetic algorithm. In: Voight, H.-M., Ebeling, W., Rechenberg, I. & Schwefel, H.-P. (eds.) *Parallel Problem Solving from Nature -- PPSN IV.* Berlin: Springer.
http://www.richardsandesforsyth.net/pubs/ppsn1996.pdf

R Core Team (2013). R: A language and environment for statistical computing. *R Foundation for statistical Computing*, Vienna, Austria.
http://www.R-project.org/.

Upton, G. & Cook, I. (2006). *Oxford Dictionary of Statistics*, second ed. Oxford: Oxford University Press.

Appendix 1 : Parameter Files

Parameters used by **eastirs3**.py are described below. Normally only the first five in this table need to be specified by a user, though for long sequences you might want to increase ntrials.

Parameter	Default value	Function
comment	[None]	This (or in fact any unrecognized parameter name, e.g. "##") can be used to insert reminders about what the file is meant to do.
jobname	eastirs	This gives the job a name. Any text string can be the value. It isn't necessary but it is useful as the jobname will be used as a prefix to the program's output files, so it can be seen that they form a group.
datfile	[None]	This should be the full file specification of a file that indicates where the input data is stored (in tab-delimited form with a header line naming the columns).
timevar	[None]	This gives the column name of the variable to be used as the time-stamp that defines the sequence of the y-values.
targvar	[None]	This gives the name of the column in the data file that contains the target variable, i.e. the values of the series itself.
outpath	.\op\	Directory to receive output files. By default this is the op subfolder of the parent directory of the program.
varmode	vc	This selects which 'fitness function' (see below) is to be used. Several functions have been implemented but at present I don't recommend any except "vc", which is the default value.
ntrials	49152	EASTIRS uses an evolutionary algorithm to optimize the subdivision of the data, according to the fitness function selected by varmode. This parameter specifies how many evolutionary trials to make, i.e. how many gene-strings will be created during the optimization process. [*]
skipvars	[None]	A list of names of variables to be ignored, separated by commas. Won't really be needed until/unless the program becomes multivariate.
listfile	[None]	File for human-readable output. If none is specified, the file will have the jobname followed by "_list.txt".
dumpfile	[None]	File to receive extended input data with column appended (called seg_ment) giving the subgroup number of each case, in tab-delimited form suitable to read into R with the read.delim() function for post-processing. If none is given, the file will have the jobname followed by "_dump.dat".
outfile	eastirs.txt	File that will receive parameter setting information, for reference.

[*] With modest-sized series, up to about 200 data points, EASTIRS is surprisingly effective. However, the search space expands exponentially as 2 to the power of the number of data points, so theory as well as practical experience suggests that with sequences of thousands of data points it is unlikely to converge on a near-optimal solution in a reasonable time.

N.B. At present, if you have more than 1 blank lines in a parameter file, the input routine chokes. I do plan to fix this at some stage; in the mean time, it is quite easy to delete empty lines using a text editor.

Appendix 2 : Sample Datafiles

These are provided in the datasets subfolder.

aircraft.dat
This dataset contains information (16 columns) concerning 103 makes of military aircraft used in World War II. For further details, see aircraft.txt in the same subfolder.

lynxdat.dat
This is the classic Canadian lynx time series, as obtained from the lynx dataset provided with the R package.
http://www.R-project.org/.

michuron.dat
This data records the annual mean level of Lake Huron (which is the same as Lake Michigan) over the years 1918 to 2013, obtained from the U.S. National Oceanic and Atmospheric Administration website.
http://www.glerl.noaa.gov/data/now/wlevels/levels.html

spots101.dat
This data series records the mean daily sunspot numbers observed during the period 1913 to 2013 inclusive. Original source: WDC-SILSO, Royal Observatory of Belgium, Brussels.
http://www.sidc.be/silso/datafiles