

Subject: Re: Low pass filter

Posted by [Martin Schultz](#) on Fri, 25 Feb 2000 08:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

Mike wrote:

>
> Hi,
>
> I was wondering if anyone could point me in the direction of a library
> that produces a lowpass filter to filter high frequency data from an
> hourly time series? Any ideas or suggestions are welcomed. Thanks in
> advance
>
> Mike
>
> --
> "Looking North West out over the Irish sea."

Mike,

you could try out my run_av.pro (attached). I wrote this specifically for something similar, so it handles missing values as well as irregularly spaced time intervals.

Hope it works out,

Martin

```
--  
[[ Dr. Martin Schultz Max-Planck-Institut fuer Meteorologie [[  
[[           Bundesstr. 55, 20146 Hamburg [[  
[[           phone: +49 40 41173-308 [[  
[[           fax: +49 40 41173-298 [[  
[[ martin.schultz@dkrz.de [[  
[[-----  
; $Id: run_av.pro,v 1.10 1999/01/22 20:12:17 mgs Stab $  
;+  
; NAME:  
;     RUN_AV (function)  
;  
; PURPOSE:  
;     Compute running average or running total of a  
;     data vector. Compared to the IDL function TS_SMOOTH,  
;     this function takes into account missing values or
```

```
; gaps in an optional x vector, and it allows for  
; even bandwidths. It can also be used to compute cumulative  
; totals.  
;  
; CATEGORY:  
;   math  
;  
; CALLING SEQUENCE:  
;   result = RUN_AV(Y [,X] [,keywords] )  
;  
; INPUTS:  
;   Y -> the data vector (a 2-D array will be treated as a vector)  
;  
;   X -> an optional X vector defining e.g. the sample times.  
;       This only has an effect when the DELTAX keyword is specified.  
;       X must be monotonically increasing and have the same  
;       number of elements as Y.  
;  
; KEYWORD PARAMETERS:  
;   WIDTH -> The number of points to use for the average or total  
;           Default is 1, i.e. Y is returned unchanged.  
;  
;   MINWIDTH -> The minimum number of points that must be valid  
;                 in order to return a average or total for the given point.  
;                 Default is MINWIDTH=WIDTH, i.e. all points must be valid  
;                 (and if X and DELTAX are specified, all points must lie  
;                 within WIDTH*DELTAX).  
;  
;   MIN_VALID -> The minimum value for valid data. Data with less than  
;                 MIN_VALID will be considered missing. MIN_VALID is also used  
;                 to indicate invalid totals or averages (1% is subtracted).  
;  
;   DELTAX -> The maximum gap between two consecutive x values.  
;           Only effective when X is given.  
;  
;   COUNT -> A named variable will return the number of points used  
;             in each average or total.  
;  
;   /TOTAL -> Set this keyword to compute running totals instead  
;                 of running averages.  
;  
; OUTPUTS:  
;   The function returns a vector with running averages or totals.  
;   The number of elements in the result vector always equals the  
;   number of elements in Y (unless an error occurs).  
;  
; SUBROUTINES:  
;
```

```

; REQUIREMENTS:
;
; NOTES:
; This function can also be used to compute accumulative totals.
; Simply set WIDTH to n_elements(Y) and MINWIDTH to 1 and use
; the /TOTAL keyword. However, this is very uneffective for large
; data vectors!
;
; EXAMPLE:
; y = findgen(20)
; print,run_av(y,width=4)
; ; IDL prints: -1E31 -1E31 -1E31 1.5 2.5 3.5 4.5 ...
;
; print,run_av(y,width=4,/TOTAL)
; ; IDL prints: -1E31 -1E31 -1E31 6 10 14 18 ...
;
; ; (cumulative total)
; print,run_av(y,width=n_elements(y),minwidth=1,/TOTAL)
; ; IDL prints: 0 1 3 ... 190
;
; x = [ 0, 2, 4, 6, 16, 20, 24, 25, 26, 27, 28, 29, 30, 32, 33 ]
; y = fltarr(n_elements(x)) + 1.
; print,run_av(y,x,width=4,count=c)
; ; IDL prints: -1E31 -1E31 -1E31 1 1 1 1 ...
; print,c
; ; IDL prints: 1 2 3 4 4 4 4 4 4 4 4 4 4 4
;
; print,run_av(y,x,deltax=2,width=4,count=c)
; ; IDL prints: -1E31 -1E31 -1E31 1 -1E31 -1E31 -1E31
; ; -1E31 -1E31 -1E31 1 1 1 1
; print,c
; ; IDL prints: 1 2 3 4 3 2 1 1 2 3 4 4 4 4 4
;
; MODIFICATION HISTORY:
; mgs, 21 Oct 1998: VERSION 1.00
;
;
;-
; Copyright (C) 1998, Martin Schultz, Harvard University
; This software is provided as is without any warranty
; whatsoever. It may be freely used, copied or distributed
; for non-commercial purposes. This copyright notice must be
; kept with any copy of this software. If this software shall
; be used commercially or sold as part of a larger package,
; please contact the author to arrange payment.
; Bugs and comments should be directed to mgs@io.harvard.edu
; with subject "IDL routine run_av"
;----- -

```

```

function run_av,y,x,width=width,min_valid=min_valid,deltax=deltax, $
minwidth=minwidth,count=rcount,total=ctotal

result = 0.
if (n_elements(y) eq 0) then return,result

; =====
; set up result array and temporary storage
; =====

average = not keyword_set(ctotal)

if (n_elements(width) eq 0) then width = 1 $  

else width = fix(abs(width[0]))

if (n_elements(minwidth) eq 0) then minwidth = width $  

else minwidth = minwidth < width ; no larger than width!

if (width eq 0) then begin  

  message,'WIDTH must be greater or equal 1!',/Cont  

  return,result
endif

accu = fltarr(width)
count = intarr(width)
result = fltarr(n_elements(y))
rcount = intarr(n_elements(y))
ic = 0

if (n_elements(min_valid) eq 0) then min_valid = -9.99E30

; =====
; VERSION 1: no x array given
; =====

if (n_elements(x) eq 0) then begin
  ; loop through y vector and accumulate
  for i = 0L,n_elements(y)-1 do begin

    if ( (i-ic) ge width ) then ic = ic + width

    ; add current y value to all buffer elements
    ; if greater min_valid
    ; and increment counter
    if (y[i] gt min_valid) then begin
      accu[*] = accu[*] + y[i]

```

```

        count[*] = count[*] + 1
    endif

    ; read out ith buffer value and reset ith buffer
    rcount[i] = count[i-ic]
    if (count[i-ic] ge minwidth) then begin
        result[i] = accu[i-ic]
        if (average) then result[i] = result[i]/rcount[i]
    endif else begin
        result[i] = min_valid
    endelse

    accu[i-ic] = 0.
    count[i-ic] = 0

endfor

return,result
endif

; =====
; VERSION 2: with x array
; same as above, but needs to take care of min x steps
; =====

if (n_elements(x) ne n_elements(y)) then begin
    message,'X and Y must have same number of elements!',/Cont
    return,0.
endif

if (n_elements(deltax) eq 0) then begin
    xdiff = x - shift(x,1)
    deltax = max(xdiff[1:*])
endif

; loop through y vector and accumulate
for i = 0L,n_elements(y)-1 do begin

    if ( (i-ic) ge width ) then ic = ic + width

    ; add current y value to all buffer elements
    ; if greater min_valid
    ; and increment counter
    if (y[i] gt min_valid and x[i]-x[(i-1)>0] le deltax) then begin
        accu[*] = accu[*] + y[i]
        count[*] = count[*] + 1

```

```
endif

; read out ith buffer value and reset ith buffer
rcount[i] = count[i-ic]
if (count[i-ic] ge minwidth) then begin
    result[i] = accu[i-ic]
    if (average) then result[i] = result[i]/rcount[i]
endif else begin
    result[i] = min_valid
endelse

accu[i-ic] = 0.
count[i-ic] = 0

endfor

return,result
```

```
end
```

File Attachments

- 1) [run_av.pro](#), downloaded 76 times
-