"MATLAB"

Random (and irregular) snippets of code

Saturday, 4 April 2015

Alternative function: Moving average

Download

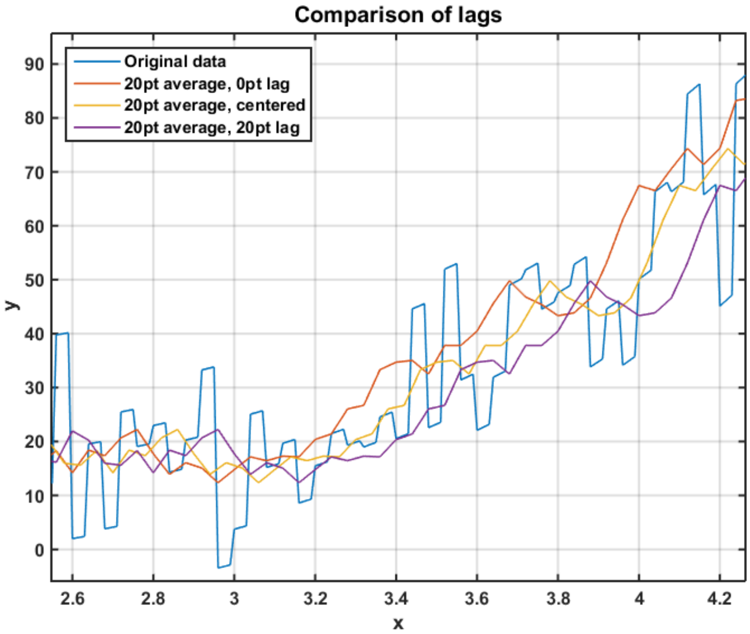
movAv2.m

This is a slightly more advanced version of movAv and provides an alternative to movavg and tsmovavg ("time-series moving average"). This version allows weighting of the mean and setting of the lag value (rather than just centring the average automatically). It also shows examples of basic management of the variables in a functions workspace, using switch and case for convenient string logic, and how to perform a weighted mean calculation. See also movAv for a couple of examples when moving averages may or may not be appropriate.

Description

Compared to movAv.m, this version adds two features:

- Lag - when performing a moving average with length **n**, **n** points in total are lost from the output. The lag value positions the output in a vector of the same length as in the input, by determining the number of NaNs at the start and end of the vector.



- Weights - Included are a few random weights as examples in the script as examples, although the only one that might be useful is 'simExp', which exponentially weights the values in the centre of the average window higher than the outer points. Weights can also be supplied as a vector the same length as the average window (**n**). The absolute values of the weights are unimportant, as they're normalised to relative values that sum to 1 in the script.

Search This Blog

Search

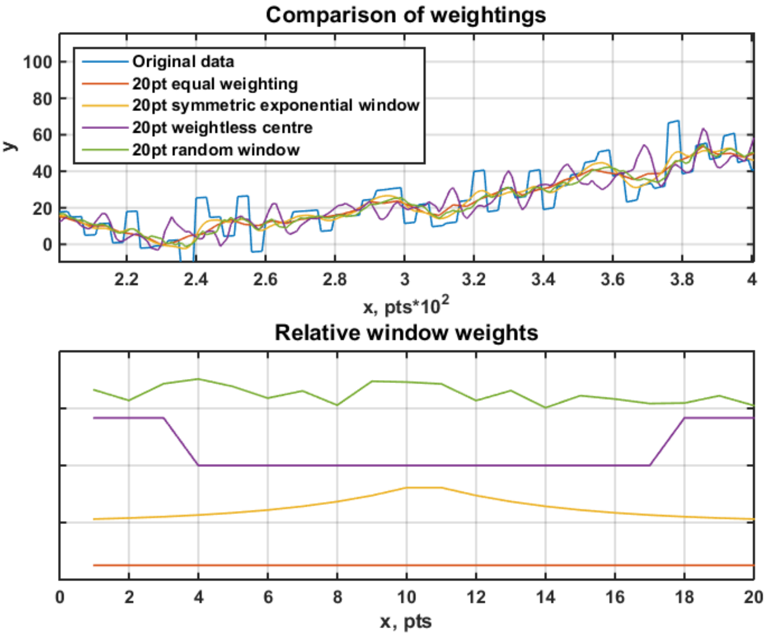
Subscribe To

Posts

Comments

Labels

- 2014b (2)
- 2AFC (2)
- Additional functions (8)
- Alternative functions (16)
- Base conversion (3)
- Coursera (5)
- Curve Fitting (5)
- Data acquisition (2)
- Data Analysis (18)
- Figures (2)
- Finance (5)
- General Linear Model (1)
- Graphics (2)
- Handles (2)
- Importing (1)
- logit (1)
- MATLAB (20)
- Modelling (2)
- Multisensory integration (1)
- Neuroscience (4)
- OpenGL (1)
- Plotting (1)
- Prediction (2)
- Psychometric curve (2)
- Psychophysics (4)
- Race model inequality (1)
- Signal detection theory (2)
- Spikes (1)
- Stimulus presentation (1)
- Time value of money (4)
- Tutorials (6)



movAv2.m code run-through

```
function [output, weights] = movAv2(y,n,lag,weights)
```

movAv2 takes 4 inputs; the data to be averaged (vector, **y**), the number of points over which to perform the average (scaler, **n**), the lag (scaler, **lag**) of the output vector relative to the input vector, and a vector of weights to use (**weights**). The weights should either be a vector of length the average window (**n**), or a string that matches any of the examples in side the function (**'exp'**, **'symExp'**, **'linInc'**, or **'rand'**). The weights used are also returned for reference in the output **weights**.

```
% Calculate lag
lead = n-(n-lag);
```

First **lead** is calculated from the **lag** and **n** values. If **lag** is 2 and n is 10, **lead** will be 8 meaning the output vector will have 8 NaNs at the start, and 2 at the end.

```
% If no weights specified, assume equal
if ~exist('weights', 'var')
    weights = ones(1,n);
end
```

If weights aren't specified the variable **weights** won't exist in the functions workspace, which can be checked using the **exist** function. If **weights** doesn't exist it set as all ones (of length **n**), so weighting is equal over all points in the window.

```
if ischar(weights)
    switch weights
        case {'exp', 'Exp'}
            weights=logspace(0,1,round(n));
        case 'symExp'
            weights=[logspace(0,1,round(n/2)),logspace(1,0,round(n/2))];
        case 'linInc'
            weights=1:n;
        case 'rand'
            weights=rand(1,n);
        otherwise
disp('Invalid weights requested, weighting disabled')
            weights = ones(1,n);
    end
else
    % Weights specified as a vector
end
```

The next **if** and then **switch** check if **weights** has been specified as a vector or as a string. If **weights** is already a vector, nothing is done. If it's a string **ischar(weights)** is true (note also that **ischar(weights)==1** is also true, but the extra typing is unnecessary because it's like asking if true==true), the **switch/case** block checks for which string it is, and replaces **weights** with a vector of length **n** with appropriate values. Note how multiple stings can be checked by each case using the syntax **case {'string1', 'string2'}** - if either string is true, the **case** is entered. If **weights** is a sting but not matched by any of the cases, the **otherwise** statement sets all the weights as equal.

```
% Normalise weights to 0 to 1;
weights = weights./sum(weights);
```

Blog Archive
August (1)
August (1)
May (7)
April (5)
March (4)
February (1)
August (1)
April (4)
December (1)
November (1)

Followers
Seguidores (2)
Seguir

Google Analytics

The weights are normalised before calculating the average by dividing each weight by the sum of all the weights. This makes all the weights sum to 1, and the absolute value irrelevant.

```
% Preallocate output
output=NaN(1,numel(y));
% For length of input, minus n
for a = 1:length(y)-n
   % Find index range to take average over
   b=a+n-1;
   % Calculate mean
   output(a+lead) = sum(y(a:b).*weights);
end
```

Finally, the mean is calculated in a similar way as in movAv, the differences being how the running average is placed in the output vector according to the **lead** value (instead of just centred) and weighted average is calculated by the sum of the average window (**y(a:b)**) multiplied by the normalised **weights**.

Code for figures (note, uses og and ng)

```
% Generate data
x=1:0.01:5;
noiseReps = 4;
noise = repmat(randn(1,ceil(numel(x)/noiseReps)),noiseReps,1);
noise = reshape(noise, 1, length(noise)*noiseReps);
y=exp(x)+10*noise(1:length(x));
% Figure 1
og
figure
y1=movAv2(y,20,0);
y2=movAv2(y,20,round(20/2));
y3=movAv2(y,20,20);
plot(x, [y', y1', y2', y3'])
legend('Original data', ...
   '20pt average, 0pt lag', ...
   '20pt average, centered', ...
   '20pt average, 20pt lag')
xlabel('x')
ylabel('y')
title('Comparison of lags')
ng
% Figure 2
og
figure
n=20;
w0 = NaN(1,n);
[y1, w1]=movAv2(y,20,10);
[y2, w2]=movAv2(y,20,10,'symExp');
[y3, w3]=movAv2(y,20,10,[10 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 10 10]);
[y4, w4]=movAv2(y,20,10,'rand');
subplot(2,1,1), plot(x, [y', y1', y2', y3', y4']);
title('Comparison of weightings')
ylabel('y')
xlabel('x, pts*10^2')
legend('Original data', ...
   '20pt equal weighting', ...
   '20pt symmetric exponential window', ...
   '20pt weightless centre', ...
   '20pt random window')
subplot(2,1,2), plot(1:n, [w0', w1', w2'+0.2, w3'+0.4, w4'+0.6])
title('Relative window weights')
xlabel('x, pts')
a=gca;
a.YTickLabel=[];
ng
```

Posted by Matbloggs G+1 Recommend this on Google

Labels: Alternative functions, Data Analysis, MATLAB

No comments:

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)