Durand Claire

ECE 3522: Stochastic Processes in Signals and Systems

Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA 1912

# Problem Statement

This familiarization with reading and processing time series data in MATLAB involved loading and plotting two sets of data, Google’s stock prices since its inception and an audio file sampled at 8 kHz. Each set of data was then analyzed in order to obtain the minimum and maximum values, the mean, the median, and the variance of each. These values were then to be found again using a loop over pre-defined windows of data with pre-defined frames. The values to be used for the Google stock data were frames of 1, 7, 14, and 30 with windows of 7 and 30. The values to be used for the speech signal data were frames of 40, 80, and 160 with windows of 160 and 240.

# Approach and Results

In order to plot the steps created by plotting the mean using window-frame analysis, the center of each frame was found. The equation to find the center of the frame is as follows: , where is the center of the frame and is the number of points in the current frame, found in the M matrix. Then the window was centered about that point. If the number of points in the frame and the window are both even or both odd, the equations to find the minimum and maximum indexes of the window are as follows: and , where and are respectively the minimum and maximum indexes of the window and is the number of points in the current window. However, if the number of points in the frame and the window are not both even or both odd, then these equations become: and.

Once the proper window was obtained, the mean was found for that window, and plotted for the length of the frame in an output vector. The length of the frame was added to the center of the old frame in order to find the center of the next frame. This was repeated until the center of the frame surpassed the length of the data. Note that catch-alls were placed in order to prevent the window to seep into the non-positive index or past the last data point. In the cases where the center of the frame was too close to the edges of data for the window, the window was shortened instead of zero-stuffed in order to preserve the integrity of the data.

# MATLAB Code

The code was separated by the data that was loaded in. First Google’s stock data was analyzed as follows.

Computer Assignment No. 1 : Simple Statistics 1

Load and plot the data for Google's stock prices 1

Compute some global statistics on these signals (min & max values, mean, median, variance) 2

Windows and frames 3

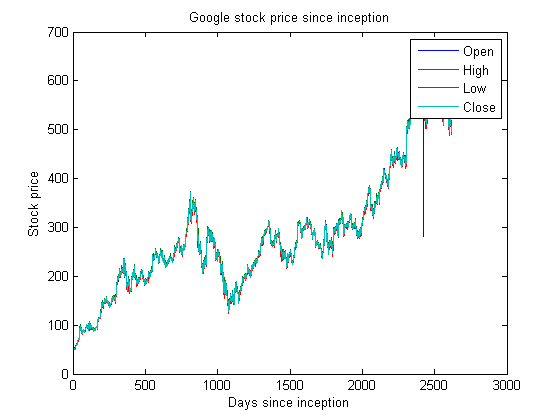
## Computer Assignment No. 1 : Simple Statistics

Part 1 - Google stock prices

clear; clc; clf;  
close all

## Load and plot the data for Google's stock prices

% Load the data  
stocks = 'google\_v00.csv'; % Create a variable for the file name  
G = csvread(stocks, 1,1); % Read in the data, excluding the text parts.  
  
% From the matrix loaded with the data, extract each "signal"  
Open = G(:,1); % Opening price  
High = G(:,2); % Highest price  
Low = G(:,3); % Lowest price  
Close = G(:,4); % Closing price  
  
% Obtain data dimensions  
[m,n] = size(G); % Obtain the size of the matrix (how many days of data)  
d = 1:m; % Create a time vector d for each day of stock data  
  
% Plot the data as a function of the days  
figure(1)  
plot(d, Open, d, High, d, Low, d, Close),  
xlabel('Days since inception'), ylabel('Stock price'),  
legend('Open', 'High', 'Low', 'Close'),  
title('Google stock price since inception')



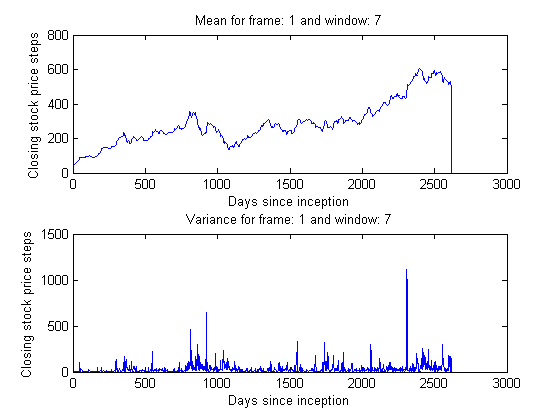
## Compute some global statistics on these signals (min & max values, mean, median, variance)

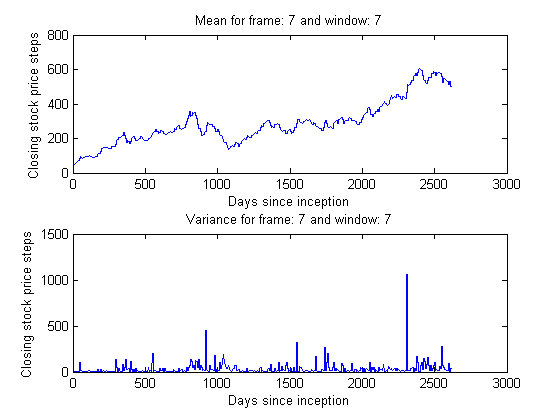
Close\_min = min(Close) % Find the min value in Close data  
Close\_max = max(Close) % Find the max value in Close data  
Close\_mean = mean(Close) % Find the mean in Close data  
Close\_median = median(Close) % Find the median in Close data  
Close\_variance = var(Close) % Find the variance in Close data

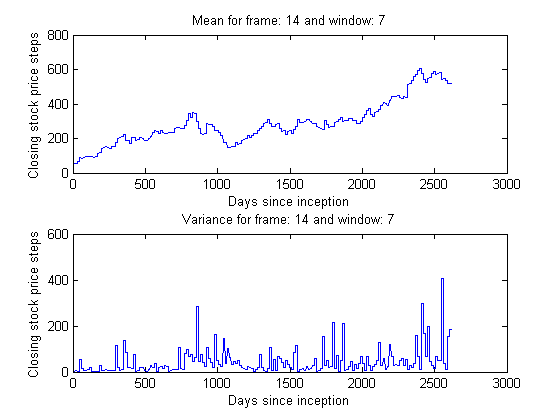
Close\_min =  
  
 49.9500  
  
  
Close\_max =  
  
 609.4700  
  
  
Close\_mean =  
  
 286.7374  
  
  
Close\_median =  
  
 264.8250  
  
  
Close\_variance =  
  
 1.6194e+04

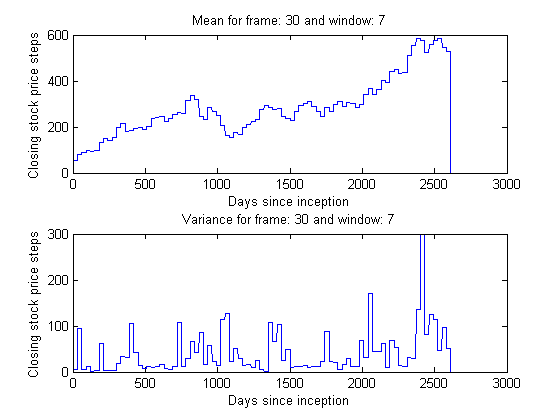
## Windows and frames

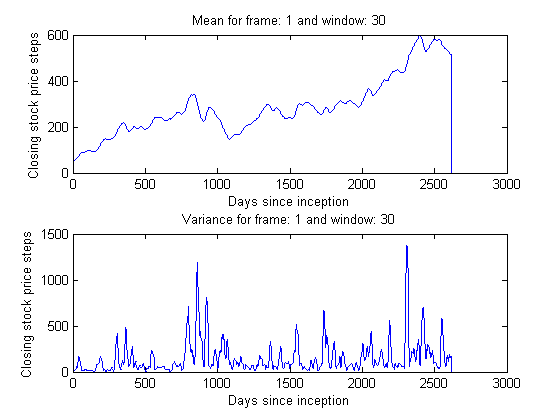
% SIGNAL: =================================================================  
% FRAMES: |---.---|---.---|---.---|---.---|---.---|---.---|---.---|---.---|  
% WINDOWS: <---------------> <--------------->  
% <---------------> <--------------->  
% <---------------> <--------------->  
%  
  
% Define two key parameters:  
% M: frame duration in samples - how often we compute an output  
% N: window duration in samples - how much data we use in each computation  
  
M = [1,7,14,30]; % days  
N = [7,30]; % days  
  
fig = 1;  
  
n = 1;  
  
for n = 1:length(N)  
  
 m = 1;  
  
 for m = 1:length(M)  
  
 % Create a matrix to store the output mean  
 G\_output\_mean = zeros(length(Close),1);  
 G\_output\_var = zeros(length(Close),1);  
  
 % Find the center of the first frame  
 Mc = (M(m)+1)/2;  
  
 while (Mc<length(Close))  
 % Establish the bounds of the window subarray we're looking at  
 %  
 % Case where M and N are both either even or odd  
 % then case where M and N are not both even or odd  
 % Note that this will tilt the mean calculation toward the future  
 % however we will remain consistent in order to avoid disparities.  
 if (mod(M(m),2) == mod(N(n),2))  
 Nmin = Mc - ((N(n)-1)/2);  
 Nmax = Mc + ((N(n)-1)/2);  
 else  
 Nmin = Mc - (N(n)/2) - 1;  
 Nmax = Mc + (N(n)/2);  
 end  
  
 % Handle the beginning and end  
 if Nmin < 1  
 Nmin = 1;  
 end  
  
 if Nmax > length(Close)  
 Nmax = length(Close);  
 end  
  
 Window = Close(Nmin:Nmax);  
  
 % Establish the bounds of the output subarray in which we're placing results  
 Mmin = Mc - ((M(m)-1)/2);  
 Mmax = Mc + ((M(m)-1)/2);  
  
 % Don't let your output matrix run larger than our input matrix  
 if (Mmax > length(G\_output\_mean))  
 Mmax = length(G\_output\_mean);  
 end  
  
 % Place the mean value of the window subarray in every value of the output  
 % frame subarray  
 for Mi = Mmin:Mmax  
 G\_output\_mean(Mi) = mean(Window);  
 G\_output\_var(Mi) = var(Window);  
 end  
  
 Mc = Mc + M(m); % Find the new frame center  
 end  
  
 fig = fig+1;  
  
 % Plot the data as a function of the days  
 figure(fig)  
 subplot(2,1,1)  
 plot(d, G\_output\_mean),  
 xlabel('Days since inception'), ylabel('Closing stock price steps'),  
 str = sprintf('Mean for frame: %d and window: %d', M(m),N(n));  
 title(str)  
 subplot(2,1,2)  
 plot(d, G\_output\_var),  
 xlabel('Days since inception'), ylabel('Closing stock price steps'),  
 str = sprintf('Variance for frame: %d and window: %d', M(m),N(n));  
 title(str)  
 end  
end

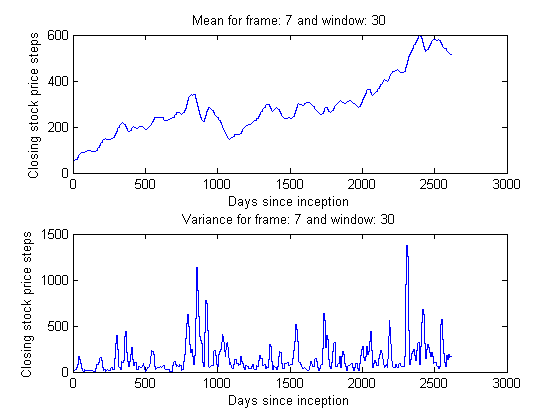


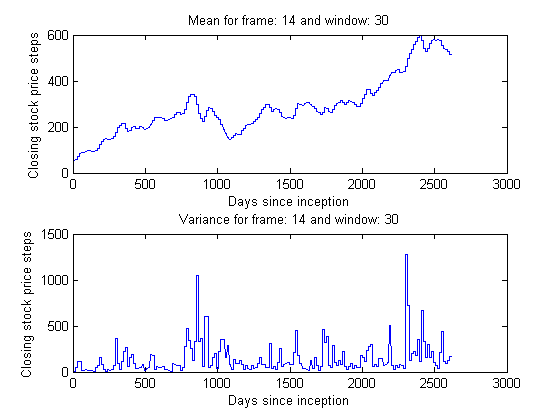


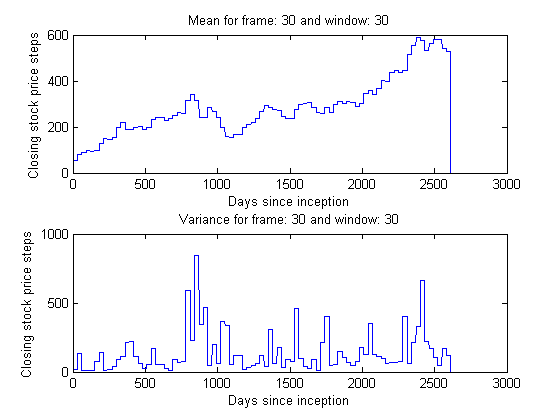












[*Published with MATLAB® R2014a*](http://www.mathworks.com/products/matlab)

# Note that the lines going to zero at the end of some of the signals can be ignored. These are created by a missing value at the end of the vector, which MATLAB simply assumes is zero when plotting the vector.

# The speech signal data was analyzed in the same fashion.

Computer Assignment No. 1 : Simple Statistics 1

Load and plot the speech signal data 1

Compute some global statistics on the signal (min & max values, mean, median, variance) 2

Windows and frames 3

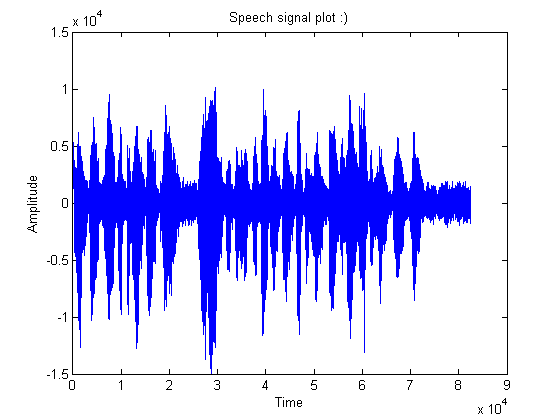
## Computer Assignment No. 1 : Simple Statistics

Part 1 - Speech signal

clear; clc; clf;  
close all

## Load and plot the speech signal data

% Open the file  
fp = fopen('rec\_01\_speech.raw','r');  
  
% Load the data.  
% It is 16-bit sampled data, so we must load it as short integers.  
sig = fread(fp,inf,'int16');  
  
% Close the file  
fclose(fp);  
  
% Obtain data dimensions  
[p,q] = size(sig); % Obtain the size of the matrix (how many points of data)  
data\_pts = 1:p; % Create a time vector for each point of speech data  
  
% Plot the data as a function of the points  
figure(1)  
plot(data\_pts, sig),  
xlabel('Time'), ylabel('Amplitude'),  
title('Speech signal plot :)')



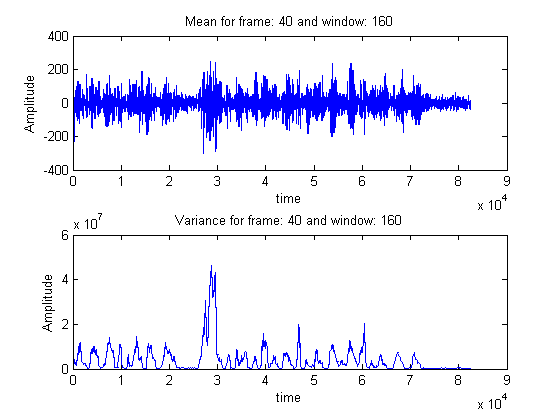
## Compute some global statistics on the signal (min & max values, mean, median, variance)

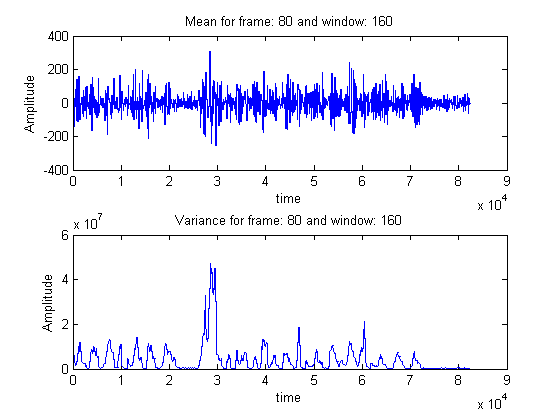
Signal\_min = min(sig) % Find the min value in speech signal data  
Signal\_max = max(sig) % Find the max value in speech signal data  
Signal\_mean = mean(sig) % Find the mean of the speech signal data  
Signal\_med = median(sig) % Find the median of the speech signal data  
Signal\_var = var(sig) % Find the variance of the speech signal data

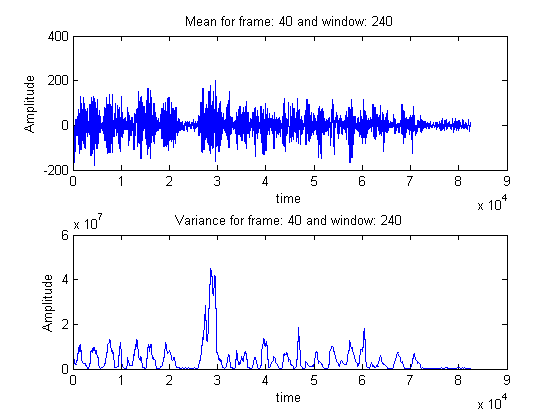
Signal\_min =  
  
 -14993  
  
  
Signal\_max =  
  
 10104  
  
  
Signal\_mean =  
  
 -0.3891  
  
  
Signal\_med =  
  
 83  
  
  
Signal\_var =  
  
 4.1394e+06

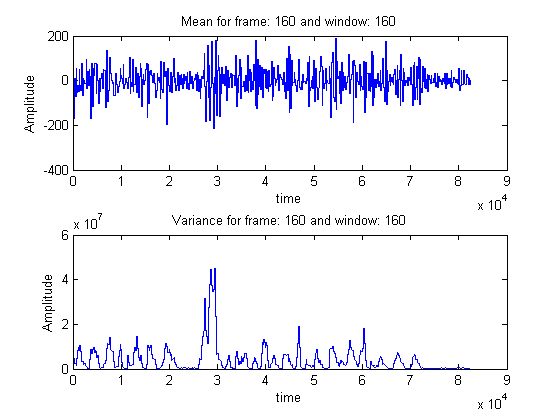
## Windows and frames

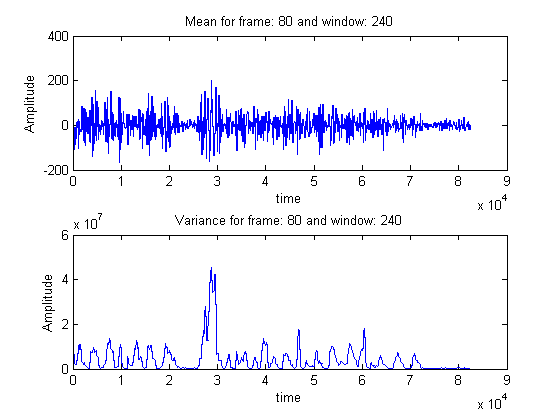
% SIGNAL: =================================================================  
% FRAMES: |---.---|---.---|---.---|---.---|---.---|---.---|---.---|---.---|  
% WINDOWS: <---------------> <--------------->  
% <---------------> <--------------->  
% <---------------> <--------------->  
%  
  
% Define two key parameters:  
% M: frame duration in samples - how often we compute an output  
% N: window duration in samples - how much data we use in each computation  
  
M = [40,80,160]; % days  
N = [160,240]; % days  
  
fig = 1;  
  
n = 1;  
  
for n = 1:length(N)  
  
 m = 1;  
  
 for m = 1:length(M)  
  
 % Create a matrix to store the output  
 sig\_output\_mean = zeros(length(sig),1);  
 sig\_output\_var = zeros(length(sig),1);  
  
 % Find the center of the first frame  
 Mc = (M(m)+1)/2;  
  
 while (Mc<length(sig))  
 % Establish the bounds of the window subarray we're looking at  
 %  
 % Case where M and N are both either even or odd  
 % then case where M and N are not both even or odd  
 % Note that this will tilt the mean calculation toward the future  
 % however we will remain consistent in order to avoid disparities.  
 if (mod(M(m),2) == mod(N(n),2))  
 Nmin = Mc - ((N(n)-1)/2);  
 Nmax = Mc + ((N(n)-1)/2);  
 else  
 Nmin = Mc - (N(n)/2) - 1;  
 Nmax = Mc + (N(n)/2);  
 end  
  
 % Handle the beginning and end  
 if Nmin < 1  
 Nmin = 1;  
 end  
  
 if Nmax > length(sig)  
 Nmax = length(sig);  
 end  
  
 Window = sig(Nmin:Nmax);  
  
 % Establish the bounds of the output subarray in which we're placing results  
 Mmin = Mc - ((M(m)-1)/2);  
 Mmax = Mc + ((M(m)-1)/2);  
  
 % Don't let your output matrix run larger than our input matrix  
 if (Mmax > length(sig\_output\_mean))  
 Mmax = length(sig\_output\_mean);  
 end  
  
 % Place the mean value of the window subarray in every value of the output  
 % frame subarray  
 for Mi = Mmin:Mmax  
 sig\_output\_mean(Mi) = mean(Window);  
 sig\_output\_var(Mi) = var(Window);  
 end  
  
 Mc = Mc + M(m); % Find the new frame center  
 end  
  
 fig = fig+1;  
  
 % Plot the data as a function of the days  
 figure(fig)  
 subplot(2,1,1)  
 plot(data\_pts, sig\_output\_mean),  
 xlabel('time'), ylabel('Amplitude'),  
 str = sprintf('Mean for frame: %d and window: %d', M(m),N(n))  
 title(str);  
 subplot(2,1,2)  
 plot(data\_pts, sig\_output\_var),  
 xlabel('time'), ylabel('Amplitude'),  
 str = sprintf('Variance for frame: %d and window: %d', M(m),N(n))  
 title(str);  
 end  
end

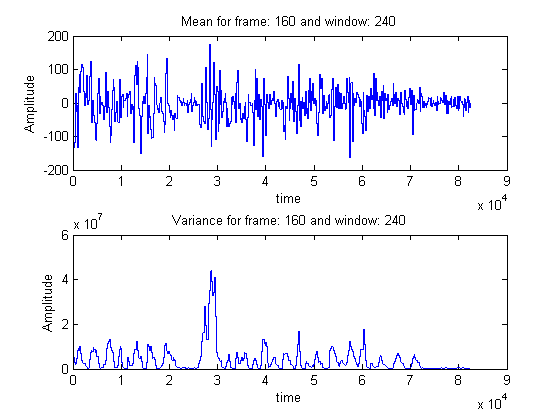












[*Published with MATLAB® R2014a*](http://www.mathworks.com/products/matlab)

# Conclusions

Unsurprisingly, window-frame analysis allows for a more faithful representation of the signal since the smaller the frame and the smaller the window, the closer the output resembles the input. Lengthening the frame makes the output “choppier” since it breaks the signal into varying number of same points. Lengthening the window smoothens the output since it takes the average over a wider number of points and therefore makes the output less sensible to sudden changes and peaks in the input signal. The mean calculated in the second part is equivalent to using a window that was as long as the data. As discussed last semester, the window and frame sizes should be carefully chosen in order to show more or less specific trends in the signal.