Computational Methods  (PHYS 2030)

**Instructors:** Prof. Christopher Bergevin (cberge@yorku.ca)

**Schedule:** Lecture: MWF 11:30-12:30 (CLH M)

**Website:** http://www.yorku.ca/cberge/2030W2018.html
% # # # EXresampleLoess.m # # #
clear;
% -----------------------------
N= 100; % # of points for fit (over interval [min(x) max(x)])
xR= [0 1]; % range of x-values
scaleN= 0.3; % scale factor for noise
alpha= 0.17; % loess fit parameter (between 0 and 1)
order= 1; % order of polynomial for fit
M= 20; % # of resamples
range= 2/3; % range to vary span over [0,1]
% -----------------------------
x= linspace(min(xR),max(xR),N); % determine fit x-values
polyS= ceil(10*rand(1)); % randomly determine polynomial order
Pv= 2*rand(polyS,1)-1; % polynomial coefficients
y= polyval(Pv,x)+ rand(1)*sin(2*pi*x)+ scaleN*randn(N,1)'; % pseudo-random 'data'
N= numel(x); % total number of points
% ---
% loess fit to ENTIRE data set
xFit= linspace(min(xR),max(xR),N); % can 'resample' if desired (i.e., need not have xFit=x)
yFitW= loess(x,y,xFit,alpha,order,[],1,0,0.1); % loess fit via external function
% -----------------------------
% Resampling: regression on randomly chosen subsets of the data
for nn=1:M
    temp= randperm(N); % create random resampling index
tINDX= temp(1:ceil(range*N)); % trim to subset of specified size
temp2(nn,:)= loess(x(tINDX),y(tINDX),xFit,alpha,order,[],1,0,0.1);
end
yFitBS= mean(temp2);
errBS= std(temp2);
% ---
% visualize
figure(1); clf; hold on; grid on;
boundedline(xFit,yFitBS,2*errBS,'b'); % trend from bootstrapping
%plot(xFit,yFitBS,'b-','LineWidth',2); % without boundedline.m, uncomment these lines
%plot(xFit,yFitBS-2*errBS,'b-'); plot(xFit,yFitBS+2*errBS,'b-'); % 95% CIs for bootstrapped trend
plot(xFit,yFitW,'r-','LineWidth',2); % trend for whole data set
plot(x,y,'ko');
xlabel('x'); ylabel('y');
legend('resampled CI','mean resampled trend','trend to whole set','data','Location','SouthWest');

Note: This subsampling method does not use replacement (i.e., a given data point can appear once at most in a resampled set)
scaleN = 0.3; % scale factor for noise
alpha = 0.17; % loess fit parameter (between 0 and 1)
M = 20; % # of resamples
range = 2/3; % range to vary span over [0,1]

Very useful idea here: We can determine uncertainty associated with our trends!

Note: We are not bootstrapping here (strictly speaking)
scaleN = 0.3;  % scale factor for noise
alpha = 0.17;  % loess fit parameter (between 0 and 1)
M = 20;  % # of resamples
range = 2/3;  % range to vary span over [0,1]
scaleN = 0.05;  % scale factor for noise
alpha = 0.17;  % loess fit parameter (between 0 and 1)
M = 20;  % # of resamples
range = 2/3;  % range to vary span over [0,1]
scaleN = 0.3;  % scale factor for noise
alpha = 0.07;  % loess fit parameter (between 0 and 1)
M = 20;  % # of resamples
range = 2/3;  % range to vary span over [0,1]
scaleN = 0.6;  % scale factor for noise
alpha = 0.07;  % loess fit parameter (between 0 and 1)
M = 20;       % # of resamples
range = 2/3;  % range to vary span over [0,1]
scaleN = 0.6; % scale factor for noise
alpha = 0.3; % loess fit parameter (between 0 and 1)
M = 20; % # of resamples
range = 2/3; % range to vary span over [0,1]
There is another method for computing standard errors called the jackknife... It is less computationally expensive than the bootstrap but it is less general... However, unlike the bootstrap, the jackknife does not produce consistent estimates of standard error of sample quantiles.”
% ### EXresampleLoess2.m ###
clear;

% -----------------------------
N= 100; % # of points for fit (over interval [min(x) max(x)])
xR= [0 1]; % range of x-values
scaleN= 0.3; % scale factor for noise
alpha= 0.17; % loess fit parameter (between 0 and 1)
order= 1; % order of polynomial for fit
M= 20; % # of resamples
range= 2/3; % range to vary span over [0,1]
replace= 1; % use replacement? [0-no, 1-yes]

% -----------------------------

x= linspace(min(xR),max(xR),N); % determine fit x-values
polyS= ceil(10*rand(1)); % randomly determine polynomial order
Pv= 2*rand(polyS,1)-1; % polynomial coefficients
y= polyval(Pv,x) + rand(1)*sin(2*pi*x) + scaleN*randn(N,1); % pseudo-random 'data'
N= numel(x); % total number of points
% ---
% loess fit to ENTIRE data set
xFit= linspace(min(xR),max(xR),N); % can 'resample' if desired (i.e., need not have xFit=x)
yFitW= loess(x,y,xFit,alpha,order,[],1,0,0.1); % loess fit via external function
% ---
% Resampling: regression on randomly chosen subsets of the data
for nn=1:M
    % create randomresampling index
    if replace==0
        temp= randperm(N); % each entry is unique
    elseif replace==1
        temp= floor(N*rand(1,N))+1; % allow for possible repeated values
    end
    tINDX= temp(1:ceil(range*N)); % trim to subset of specified size
    temp2(nn,:) = loess(x(tINDX),y(tINDX),xFit,alpha,order,[],1,0,0.1);
end
yFitBS= mean(temp2);
errBS= std(temp2);
% ---
% visualize
figure(1); clf; hold on; grid on;
boundedline(xFit,yFitBS,2*errBS,'b'); % trend from bootstrapping
%plot(xFit,yFitBS,'b--','LineWidth',2); % without boundedline.m, uncomment these lines
%plot(xFit,yFitBS-2*errBS,'b--'); plot(xFit,yFitBS+2*errBS,'b--'); % 95% CIs for bootstrapped trend
plot(xFit,yFitW,'r--','LineWidth',2); % trend for whole data set
plot(x,y,'ko');
xlabel('x'); ylabel('y');
legend('resampled CI','mean resampled trend','trend to whole set','data','Location','SouthWest');

Note: This version **does allow for replacement** (i.e., if replace= 1 and range= 1, then this would be *bootstrapping*).
**Be careful!**

→ A normal distribution doesn’t necessarily tell you where an individual measurement lies, but the mean value across a set of measurements (i.e., what happens for compiling across REPEATED measurements)
Bonus: Falling snowflakes

letitsnow.m
(http://www.mathworks.com/matlabcentral/fileexchange/34349-let-it-snow--falling-snow-on-every-figure-/content/letitsnow.m)
% - For snowflakes -
num_snowflakes = 120;
snowflakes_size = 11;
snowflakes_color = sf_color.*[1 1 1];
height_range = 3;

snow_timer = timer('TimerFcn', @updateSnow_Callback_nest,...
    'ExecutionMode','fixedRate',... 
    'Period', 1./fps, 'BusyMode', 'queue');

% Position of snowflakes
snowflakes_x = rand(1, num_snowflakes);
snowflakes_y = rand(1, num_snowflakes);
snowflakes_y = (snowflakes_y.* height_range.^3).^(1/3)+1;
snowflakes_pos = [snowflakes_x; snowflakes_y];

% Velocity of snowflakes
snowflakes_v = rand(1, num_snowflakes).*0.1 + 0.1;  % (distance per second)
snowflakes_theta = (rand(1, num_snowflakes).*60 + 240).*pi/180;
snowflakes_vel = [snowflakes_v.*cos(snowflakes_theta);
snowflakes_v.*sin(snowflakes_theta)];
Post-class exercises

- Fiddle around with the random # generator codes (i.e., the ones that don’t explicitly make use of rand.m or rand.m). Can you get something reasonable working?

- Modify EXintegrateMC1.m so to recreate these types of normal distributions

- Create a normally distributed set of numbers and verify the relationship between the area and associated STD

- Can you modify the random walker codes (1-D or 2-D) in some fashion to create ‘anomalous diffusion’ conditions?

- What is the connection between the MSD for an ensemble of random walkers and the STD of a normal distribution?

- What difference does ‘with’ or ‘without replacement’ make when resampling? How does such depend upon sample size?