```
%% Overview
% The purpose of this code is to take text files from the local directory
% that are spectral data from spectroscopic or microspectrocopic techniques
% and convert it into a format whereby feature extraction can take place
% for ML applications. The code transforms the data into a matrix that can
% be rescaled to relative proportions, converts to absorbance from
% transmittance, reduces the range of wave numbers, plot the resulting
% data, and gathers the necessary predictors from MATLAB's in-built feature
% extraction functions and a few nove; feature extractions that looks at
% the behavior of the spectra via a "cumulative intensity function" (20th to principal
% wave number). NOTE: May need tweaking if there is a limited amount of
% maxima

% Change labels and switch cases, at will, to expedite feature extraction!
% Ex: cvar sets up the categorization of the matrices and should be labeled
% accordingly

%% Loads files, labels, and converts to a matrix

swc = "on"; %Turn switch on or off for table write

ds = tabularTextDatastore('Unknown (.txt)\Unknown Spectra Master CSV.csv');
k = 1;
data_l = read(ds);
data_l = table2array(data_l);
%% Switch case for data type (NOTE: Arrays must be matching in length for ALL files)
v = 1; % Comment/uncomment for debugging/code building purposed. (SET TO DESIRED MODULE↙
TO BYPASS INPUT
% v = input('Is your data from OS or elsewhere? (Type "1" for OS or "2" for elsewhere):↙
');
switch v
    case 1
        l_files = length(data_l(1,:)) - 1;

        for i = 1:l_files
            B(:,k:k+1) = [data_l(:,1) data_l(:,i+1)];
            k = k + 2;
        end

        reset(ds)
    case 2
        l_files = length(data_l);

        for i = 1:l_files
            B(:,k:k+1) = [data_l(:,k) data_l(:,k+1)];
            k = k + 2;
        end

        reset(ds)
    otherwise
        disp('Error: Incorrect input.')
```

```
end

%% Absolute to Relative Intensity Conversion
v = 2; % Comment/uncomment for debugging/code building purposed. (SET TO DESIRED MODULE↙
TO BYPASS INPUT
% v = input('Conversion from absolute to relative intensity needed? (Type: "1" for yes or↙
"2" for no):')
switch v
    case 1
        for i = 1:l_files
            B(:,2*i) = rescale(B(:,2*i));
        end
    case 2
        disp('Moving on...')
end
%% Transmittance to Absorbance
v = 2; % Comment/uncomment for debugging/code building purposed. DELETE WHEN DONE.
% v = input('Need conversion from transmittance to absorbance (Type: "1" for "yes" or "2"↙
for "no")? ');

switch v
    case 1
        for i = 1:l_files
            B(:,2*i) = 1 - B(:,2*i);
        end
    case 2
        disp('Continuing on...')
    otherwise
        disp('Error: Incorrect input. Continuing on...')
end


%% Reduce Dataset to input LB to UB
v = 2; % Comment/uncomment for debugging/code building purposed. (SET TO DESIRED MODULE↙
TO BYPASS INPUT
% v = input('Do you need to reduce your wavelength range (1/cm)? (Type: 1 for "yes" or↙
"2" for no):');

switch v
    case 1
        LB = input('What is your lower bound?:');
        UB = input('What is yout upper bound?:');

        x = 1;
        i = 1;

        while x == 1
            if B(i,1) < LB
                B(i,:) = [];
            end
            if B(i,1) >= LB
```

```
                if B(end,1) > UB || B(end,1) == 0
                    B(end,:) = [];
                end
                if B(end,1) <= UB && B(end,1) ~= 0
                    break
                end
            end
        end

    case 2
        disp('Moving on...');
    otherwise
        disp('Input error for "Dataset Reduction"')
end

%% Plot for Data
v = 2; % Comment/uncomment for debugging/code building purposed. (SET TO DESIRED MODULE↙
TO BYPASS INPUT
% v = input('Plot matrix data? (Type: "1" for yes or "2" for no):');
switch v
    case 1

    j = 1;
    figure(1)
    for i = 1:l_files
        plot(B(:,j),B(:,j+1))
        j = j + 2;
        hold on
    end

    case 2
        disp('Moving on...')
    otherwise
        disp('Error: Plotting module.')
end


%% Artificial Dataset (Random Interpolation)
v = 2; % Comment/uncomment for debugging/code building purposed. (SET TO DESIRED MODULE↙
TO BYPASS INPUT
% v = input('Artificial dataset? (Type: "1" for yes, "2" for no):');

% Be sure to rename variables before files are saved in .mat format. Simply
% concatenate tables AFTER feature extraction into a master table. Cvar
% should be uniquely identifiable

l = 2;
switch v
    case 1
    filenum = input('How many synthetic spectra is needed?:');
```

```
    C = zeros(length(B(:,1)),l_files);
    D = zeros(length(B(:,1)),filenum);
    r = rand(1,l_files);
    r = r/sum(r);

    for n = 1:filenum
        for i = 1:l_files-1
            C(:,i) = B(:,l)*r(i);
            l = l + 2;
        end
        D(:,n) = sum(C,2); %New spectra
        r = rand(1,l_files);
        r = r/sum(r);
        l = 2;
        disp(n)
    end
    D = [B(:,1) D];
    case 2
    disp('Moving on...')
    otherwise
        disp('Error: Artificial dataset')
end



%% Feature Extraction
v = 3; % Comment/uncomment for debugging/code building purposed. (SET TO DESIRED MODULE↙
TO BYPASS INPUT
% v = input('Version 1 or 2? (Type: "1" for v1, "2" for v2, or "3" for v3):')


l_generatedfiles = l_files;

abs_max = zeros(l_generatedfiles,1);
sec_max = zeros(l_generatedfiles,1);
thr_max = zeros(l_generatedfiles,1);
width = zeros(l_generatedfiles,1);
width_2 = zeros(l_generatedfiles,1);
width_3 = zeros(l_generatedfiles,1);
prominence = zeros(l_generatedfiles,1);
prominence_2 = zeros(l_generatedfiles,1);
prominence_3 = zeros(l_generatedfiles,1);
location = zeros(l_generatedfiles,1);
location_2 = zeros(l_generatedfiles,1);
location_3 = zeros(l_generatedfiles,1);
absvalint_secderiv = zeros(l_generatedfiles,1);
avg_abs_over_absolute_max = zeros(l_generatedfiles,1);
lth = length(B(:,1));

switch v
    case 1
```

```
        disp('Moving on...')
    case 2
        int = zeros(20,1);
        pmod_par_aggreg = zeros(l_generatedfiles,1);
        RSS_pmod = zeros(l_generatedfiles,1);
        corr_pmod = zeros(l_generatedfiles,1);

        powers = zeros(19,1);
        smod_par_aggreg = zeros(l_generatedfiles,1);
        RSS_smod = zeros(l_generatedfiles,1);
        corr_smod = zeros(l_generatedfiles,1);
    case 3
        int = zeros(20,1);
        nsigmod_par_aggreg = zeros(l_generatedfiles,1);
        RSS_nsigmod = zeros(l_generatedfiles,1);
        corr_nsigmod = zeros(l_generatedfiles,1);

        powers = zeros(19,1);
        meanpw_5_15 = zeros(l_generatedfiles,1);
        tspower = zeros(l_generatedfiles,1);

        pos = zeros(l_generatedfiles,1);

    otherwise
        disp('Error: Setting zero arrays. Defaulted to v1')
end




cvar = repmat('Unk_v3',[l_generatedfiles, 1]); %CHANGE!!!!

m = 1;

for i = 1:l_generatedfiles %loop that extracts feat from spectra and 3 most principal↙
wavenumbers
    [pks,locs,w,p] = findpeaks(B(:,2*i));
    abs_max(i,:) = max(pks);
    unq = unique(pks);
    sec_max(i,:) = unq(end-1);
    thr_max(i,:) = unq(end-2);
    index = find(pks==abs_max(i,:));
    index_2 = find(pks==sec_max(i,:));
    index_3 = find(pks==thr_max(i,:));
    index_loc = find(B(:,2*i)==abs_max(i,:));
    index_loc_2 = find(B(:,2*i)==sec_max(i,:));
    index_loc_3 = find(B(:,2*i)==thr_max(i,:));
    width(i,:) = w(index);
    width_2(i,:) = w(index_2);
    width_3(i,:) = w(index_3);
    prominence(i,:) = p(index);
```

```
    prominence_2(i,:) = p(index_2);
    prominence_3(i,:) = p(index_3);
    location(i,:) = B(index_loc,m);
    location_2(i,:) = B(index_loc_2,m); %May have to specify index due to multiple
locations possessing the same intensity
    location_3(i,:) = B(index_loc_3,m);

    switch v
        case 1
        case 2
            % Power and sine reg
            % power reg
            for j = 0:19
                int(j+1,:) = unq(end-j)/abs_max(i,:);
            end
            int = flip(int); %intensities of 20th to 1st principal wave number
            principality = 1:20; %least to greatest principal wave number: 20th to 1st

            pft = fittype
('a*principality^b','independent','principality','dependent','int');
            pmod = fit(principality',int,pft);

            CI_pmod = confint(pmod,.95);
            CI_A = abs(CI_pmod(2,1) - CI_pmod(1,1));
            CI_B = abs(CI_pmod(2,2) - CI_pmod(1,2));

            pmod_par_aggreg(i,:) = abs(pmod.a/(CI_A)) + abs(pmod.b/(CI_B)); %Aggregate
parameter for first regression


            y_hat_p = (pmod.a.*principality.^pmod.b)';
            RSS_pmod(i,:) = sum((int-y_hat_p).^2); %Checks of model fits well.
            corr_pmod(i,:) = corr(int,y_hat_p); %

            % exp-sine reg of serial exponentiation (powers from 20th to 1st int by
            % nat log)

            for j = 1:18
                powers(j,:) = log(int(j+1))/log(int(j));
            end

            position = 1:19; %least to greatest intensity

            sft = fittype('a*exp(-b*position)*(z*sin(c*position-d))
+e','independent','position','dependent','powers');
            smod = fit(position',powers,sft);
            CI_smod = confint(smod,.95);

            CI_a = abs(CI_smod(2,1) - CI_smod(1,1)); %Range for the CIs for every
parameter (Updated for correct order)
            CI_b = abs(CI_smod(2,2) - CI_smod(1,2));
```

```
            CI_c = abs(CI_smod(2,3) - CI_smod(1,3));
            CI_d = abs(CI_smod(2,4) - CI_smod(1,4));
            CI_e = abs(CI_smod(2,5) - CI_smod(1,5));
            CI_z = abs(CI_smod(2,6) - CI_smod(1,6));

            smod_par_aggreg(i,:) = abs(smod.a/CI_a) + abs(smod.b/CI_b) + abs(smod.z/CI_z)↙
+ abs(smod.c/CI_c) + abs(smod.d/CI_d) + abs(smod.e/CI_e);

            y_hat_s = (smod.a*exp(-smod.b.*position).*(smod.z.*sin(smod.c.*position-smod.↙
d))+smod.e)';
            RSS_smod(i,:) = sum((powers-y_hat_s).^2); %Checks for error
            corr_smod(i,:) = corr(powers,y_hat_s);
            figure (2)
            subplot(2,1,1);
            plot(principality',y_hat_p,'b',principality',int,'g-o');% Checking models↙
fitness

            xlabel('Principality (Reverse)')
            ylabel('Intensity')
            legend('Regression','Observed')
            title('Power Regression Model (Intensity) - Unknown')
            hold on

            subplot(2,1,2);
            plot(position',y_hat_s,'k',position',powers,'b-o');

            xlabel('Powers (Reverse)')
            ylabel('Intensity')
            legend('Regression','Observed')
            title('Sine-Exponential Regression Model (Powers) - Unknown')
            hold on

        case 3 %UNDER CONSTRUCTION

          for j = 0:19
                int(j+1,:) = unq(end-j)/abs_max(i,:);
          end

          int = flip(int);
          principality = 1:20;

          comb = [principality' int];
          n = principality;
          for n = principality
              if comb(n,2) >= .5
                  pos(i,:) = comb(n,1);
                  break
              else
              end
          end
```

```
        % nsft = fittype('a_1/[1+exp(-b_1*principality+c_1)] + a_2/[1+exp(-
b_2*principality+c_2)] + a_3/[1+exp(-
b_3*principality+c_3)]','independent','principality','dependent','int');
        % options = fitoptions(nsft);
        % options.Lower = [.05,.05,.05,.1,.1,.1,1,1,1];
        % options.Upper = [.95,.95,.95,40,40,40,20,20,20];
        % nsigmod = fit(principality',int,nsft);
        %
        % y_hat_ns = ((nsigmod.a_1./(1+exp(-nsigmod.b_1.*principality+nsigmod.c_1)))+
(nsigmod.a_2./(1+exp(-nsigmod.b_2.*principality+nsigmod.c_2)))+(nsigmod.a_3./(1+exp(-
nsigmod.b_3.*principality+nsigmod.c_3))))';
        % RSS_nsigmod(i,:) = sum((int-y_hat_ns).^2);
        % corr_nsigmod(i,:) = corr(int,y_hat_ns);

        % plot(principality',y_hat_ns,'b',principality',int,'g-o') Will
        % plot BOTH parameters together
        %
        % CI_nsigmod = confint(nsigmod,.95); % STOPPING PLACE
        %
        % CI_a_1 = abs(CI_nsigmod(2,1) - CI_nsigmod(1,1)); %Range for the CIs for every
parameter (Updated for correct order)
        % CI_a_2 = abs(CI_nsigmod(2,2) - CI_nsigmod(1,2));
        % CI_a_3 = abs(CI_nsigmod(2,3) - CI_nsigmod(1,3));
        % CI_b_1 = abs(CI_nsigmod(2,4) - CI_nsigmod(1,4));
        % CI_b_2 = abs(CI_nsigmod(2,5) - CI_nsigmod(1,5));
        % CI_b_3 = abs(CI_nsigmod(2,6) - CI_nsigmod(1,6));
        % CI_c_1 = abs(CI_nsigmod(2,7) - CI_nsigmod(1,7));
        % CI_c_2 = abs(CI_nsigmod(2,8) - CI_nsigmod(1,8));
        % CI_c_3 = abs(CI_nsigmod(2,9) - CI_nsigmod(1,9));

        % if isnan(CI_a_1)
        %     [CI_a_1, CI_a_2, CI_a_3] = deal(.9,.9,.9); %CI fits bounds specified if
not stated
        %     [CI_b_1, CI_b_2, CI_b_3] = deal(39.9,39.9,39.9);
        %     [CI_c_1, CI_c_2, CI_c_3] = deal(19,19,19);
        % end
        %
        % nsigmod_par_aggreg(i,:) = abs(nsigmod.a_1/CI_a_1) + abs(nsigmod.a_2/CI_a_2) +
abs(nsigmod.a_3/CI_a_3) + abs(nsigmod.b_1/CI_b_1) + abs(nsigmod.b_2/CI_b_2) + abs
(nsigmod.b_3/CI_b_3) + abs(nsigmod.c_1/CI_c_1) + abs(nsigmod.c_2/CI_c_2) + abs(nsigmod.
c_3/CI_c_3);
        % disp(i) %COUNTER
        %
        % a_1(i,:) = nsigmod.a_1;
        % b_1(i,:) = nsigmod.b_1;
        % c_1(i,:) = nsigmod.c_1;
        % a_2(i,:) = nsigmod.a_2;
        % b_2(i,:) = nsigmod.b_2;
        % c_3(i,:) = nsigmod.c_2;
        % a_3(i,:) = nsigmod.a_3;
        % b_3(i,:) = nsigmod.b_3;
```

```
            % c_3(i,:) = nsigmod.c_3;

            for j = 1:18
                  powers(j,:) = log(int(j+1))/log(int(j));
            end

            meanpw_5_15(i,:) = mean(powers(5:15)); % Only looking at unique region in the middle of the powers curve

            % figure (2)
            % plot(principality',y_hat_ns,'b',principality',int,'g-o');% Checking models fitness
            %
            % xlabel('Principality (Reverse)')
            % ylabel('Intensity')
            % legend('Regression','Observed')
            % title('Triple Sigmoidal Model (Intensity) - BLANK')
            % hold on

            otherwise
        end

        avg_abs_over_absolute_max(i,:) = mean(B(:,2*i))/abs_max(i,:);
        dAdx = diff(B(:,2*i))./diff(B(:,m));
        dAdx(lth) = 0;
        d_dAdx = diff(dAdx)./diff(B(:,m));
        d_dAdx(lth) = 0;
        absvalint_secderiv(i,:) = trapz(B(:,m),abs(d_dAdx));

        m = m + 2;
    end

    %% Writing to Table
    switch v
        case 1
            Unk_proc_v2 = table(sec_max,thr_max,width,width_2,width_3, ...
            prominence,prominence_2,prominence_3,location,location_2,location_3, ...
            avg_abs_over_absolute_max,absvalint_secderiv,cvar);
        case 2
            Unk_v2 = table(sec_max,thr_max,width,width_2,width_3, ...
            prominence,prominence_2,prominence_3,location,location_2,location_3, ...
            avg_abs_over_absolute_max,absvalint_secderiv,pmod_par_aggreg,smod_par_aggreg, cvar);
            disp('Median of Correlations and RSS for Power Regression - Unknown')
            median(corr_pmod)
            median(RSS_pmod)
            disp('Median of Correlations and RSS for Sin-Exp Regression - Unknown')
            median(corr_smod)
            median(RSS_smod)
        case 3
            Unk_v3 = table(sec_max,thr_max,width,width_2,width_3, ...
```

```
        prominence,prominence_2,prominence_3,location,location_2,location_3, ...
        avg_abs_over_absolute_max,absvalint_secderiv,pos,meanpw_5_15,cvar);
    otherwise
        disp('Input error: table writing.')
end

%Relative int nullifies the importance of "abs_max" since all abs_max
%for all classes are 1.
```