Supporting Information:

Spectral Normalisation by Error Minimisation for Prediction of Conversion in Solvent-Free Catalytic Chain Transfer Polymerisations

Samuel J. Richardson,[a] Idriss Blakey,[a,b] Kristofer J. Thurecht,[a,b,c] Derek J. Irvine,[d] Andrew K. Whittaker*,[a,b,c]

[a]*Australian Institute for Bioengineering and Nanotechnology, University of Queensland, St Lucia, Queensland, 4072*

[b]*Centre for Advanced Imaging, University of Queensland, St Lucia, Queensland, 4072*

[c]*ARC Centre of Excellence in Convergent Bio-Nano Science and Technology*

[d]*National Centre for Industrial Microwave Processing, Department of Chemical and Environmental Engineering, University of Nottingham, Nottingham, NG7 2RD, U.K.*

*Corresponding Author

E-mail: a.whittaker@uq.edu.au. Phone +61 7 334 63885

Table S1. Extents of conversion of MMA-A set calculated by NMR.

Total Conversion	Conversion to Dimer	Conversion to Trimer
0	0	0
0	0	0
0	0	0
0.041	0.041	0
0.26	0.13	0.089
0.38	0.19	0.11
0.39	0.21	0.13
0.41	0.20	0.071
0.44	0.21	0.12
0.45	0.22	0.099
0.47	0.25	0.11
0.49	0.26	0.12
0.51	0.26	0.13

Table S2. Extents of conversion of MMA-B set calculated by NMR

Total Conversion	Conversion to Dimer	Conversion to Trimer
0	0	0
0.0039	0.0048	0
0.0098	0.0086	0
0.012	0.0082	0.0045
0.013	0.0095	0.0068
0.056	0.030	0.019
0.11	0.056	0.034
0.15	0.079	0.046
0.18	0.090	0.051
0.22	0.11	0.072
0.23	0.12	0.079
0.29	0.14	0.092
0.29	0.15	0.11

Table S3. Extents of conversion of MMA-C set calculated by NMR

Total Conversion	Conversion to Dimer	Conversion to Trimer
0	0	0
0.058	0.024	0.023
0.15	0.074	0.055
0.21	0.11	0.074
0.29	0.16	0.094
0.33	0.19	0.11
0.40	0.19	0.12
0.45	0.21	0.13
0.51	0.24	0.15
0.54	0.26	0.16
0.55	0.25	0.17
0.55	0.26	0.16
0.57	0.27	0.18

Table S4. Extents of conversion of HEMA set calculated by NMR

Total Conversion
0
0.17
0.40
0.48
0.52
0.60
0.59
0.66
0.63
0.64
0.66

Table S5. Extents of conversion of t-BMA set calculated by NMR

Total Conversion
0
0.12
0.23
0.29
0.33
0.37
0.4
0.42
0.45
0.46
0.47
0.48
0.48

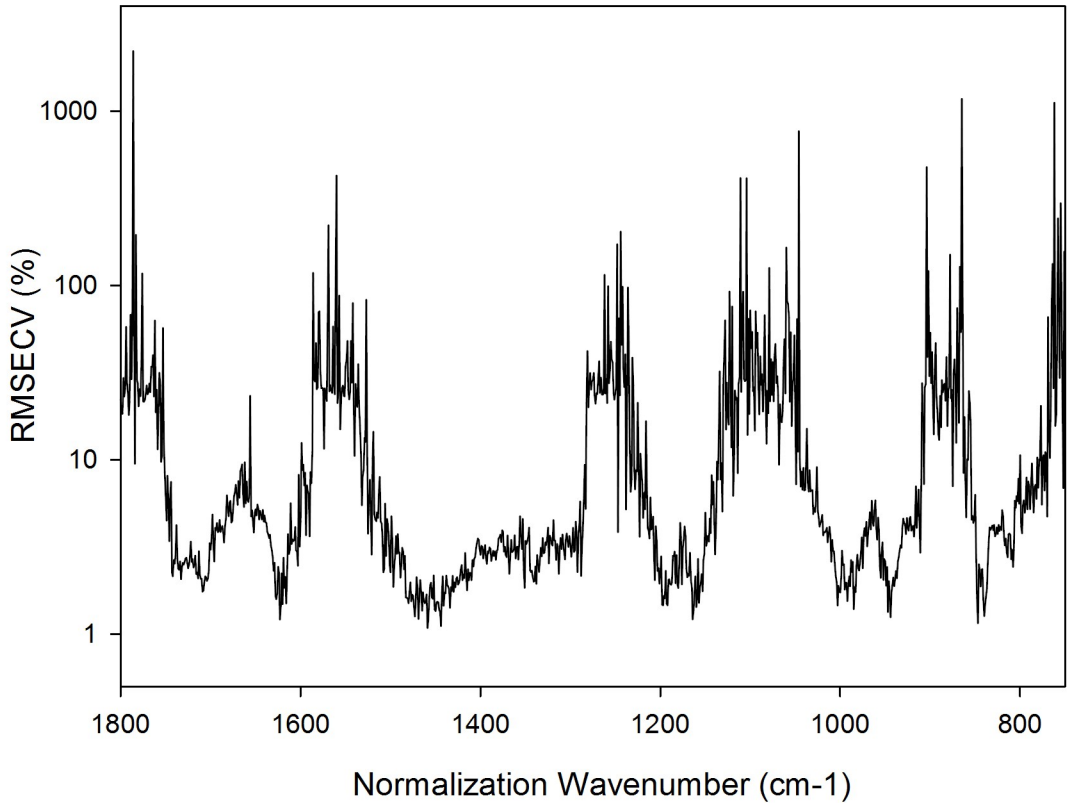

Figure S1. RMSECV as a function of normalization wavenumber for model based on MMA-A.

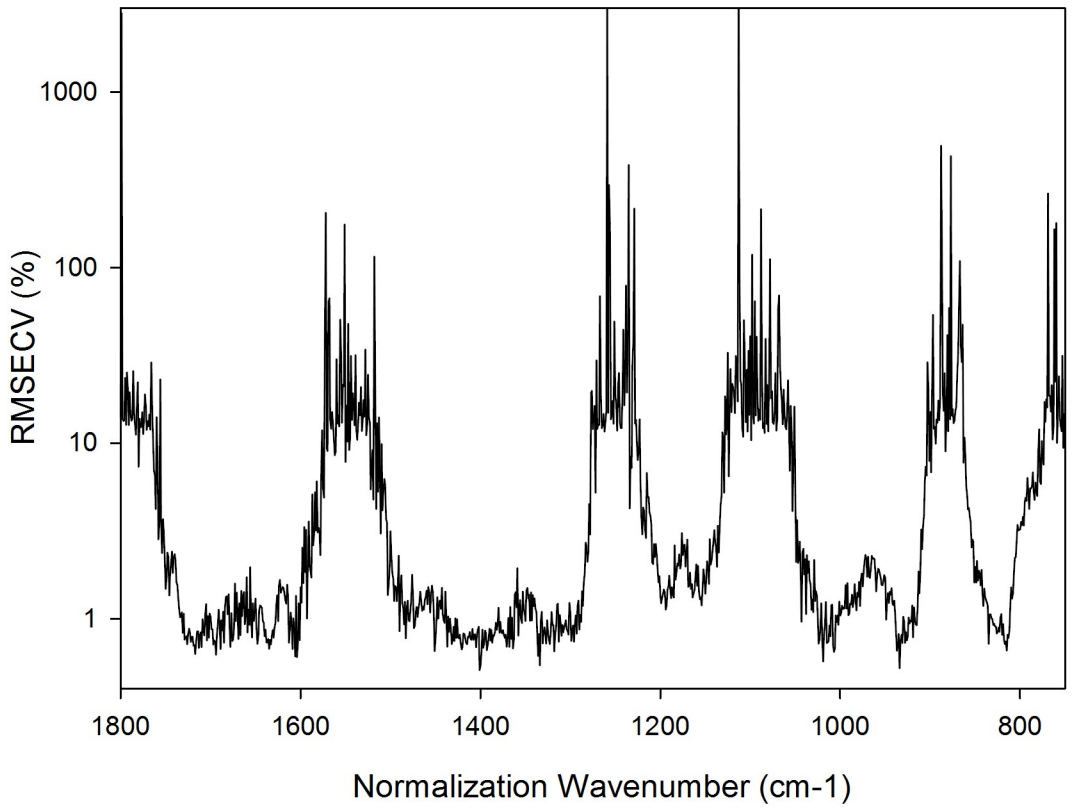

Figure S2. RMSECV as a function of normalization wavenumber for model based on MMA-B.

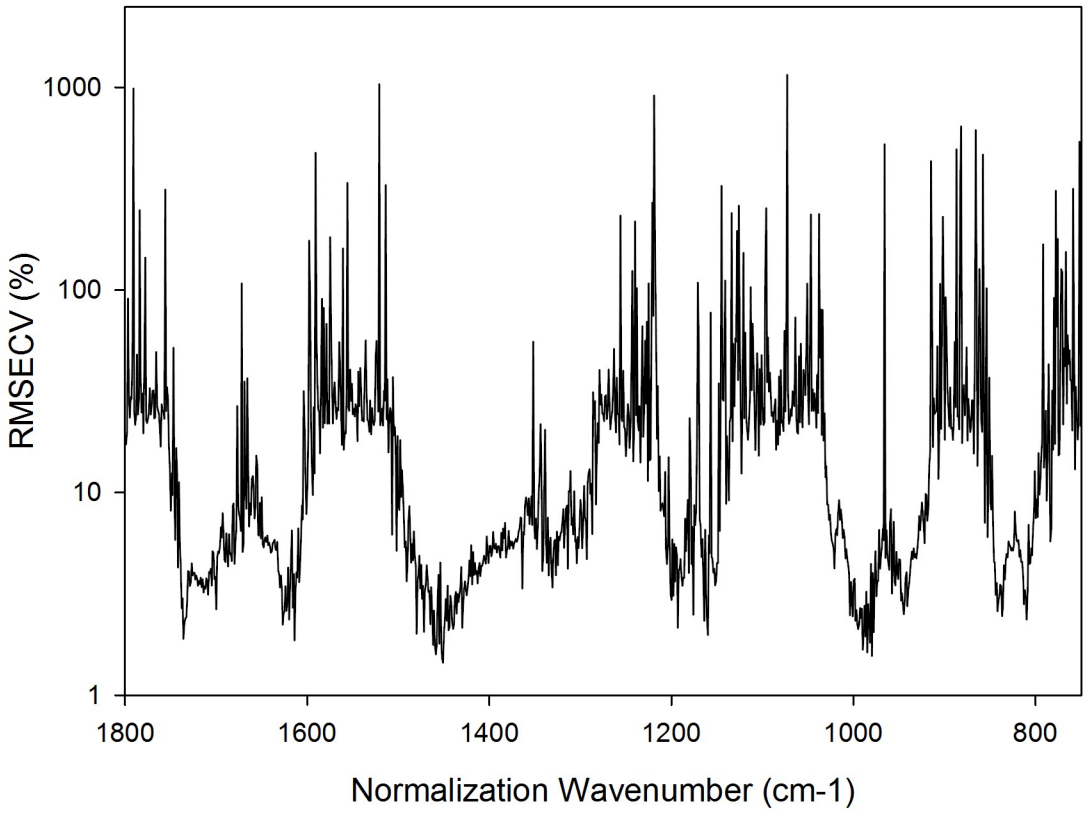

Figure S3. RMSECV as a function of normalization wavenumber for model based on MMA-C.

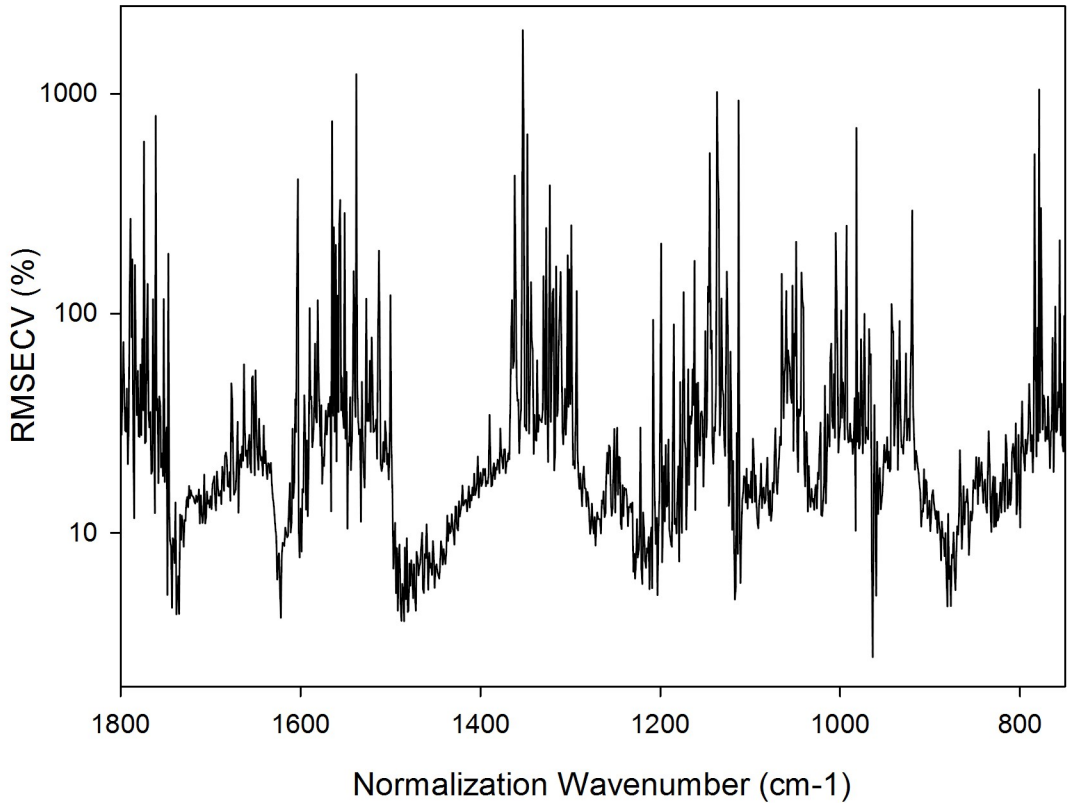

Figure S4. RMSECV as a function of normalization wavenumber for model based on HEMA.

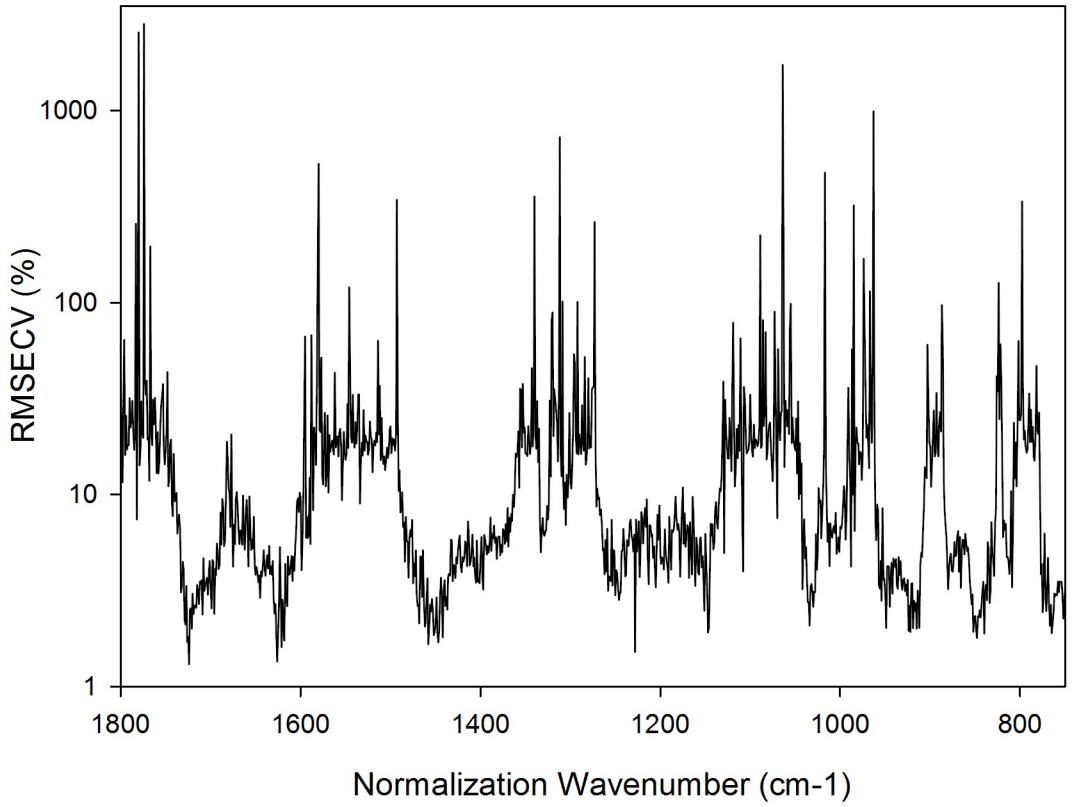

Figure S5. RMSECV as a function of normalization wavenumber for model based on t-BMA.

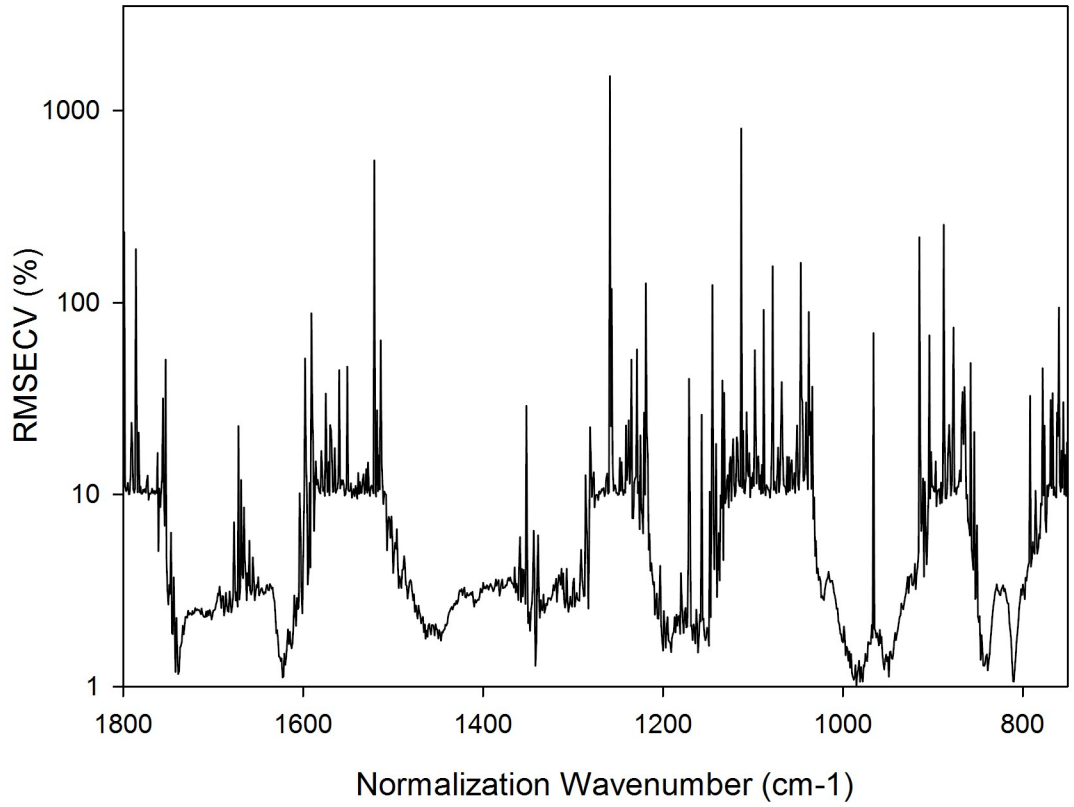

Figure S6. RMSECV as a function of normalization wavenumber for MMA_2 model based on the combined MMA set.

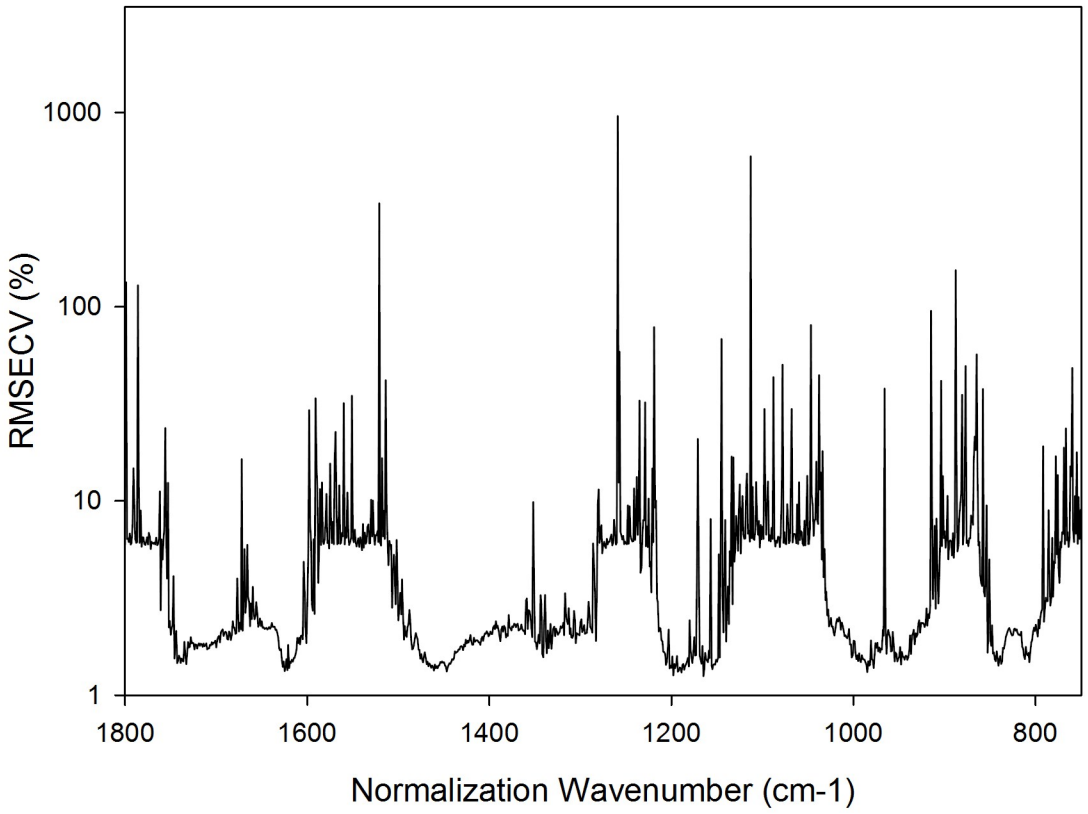

Figure S7. RMSECV as a function of normalization wavenumber for MMA_3 model based on the combined MMA set.

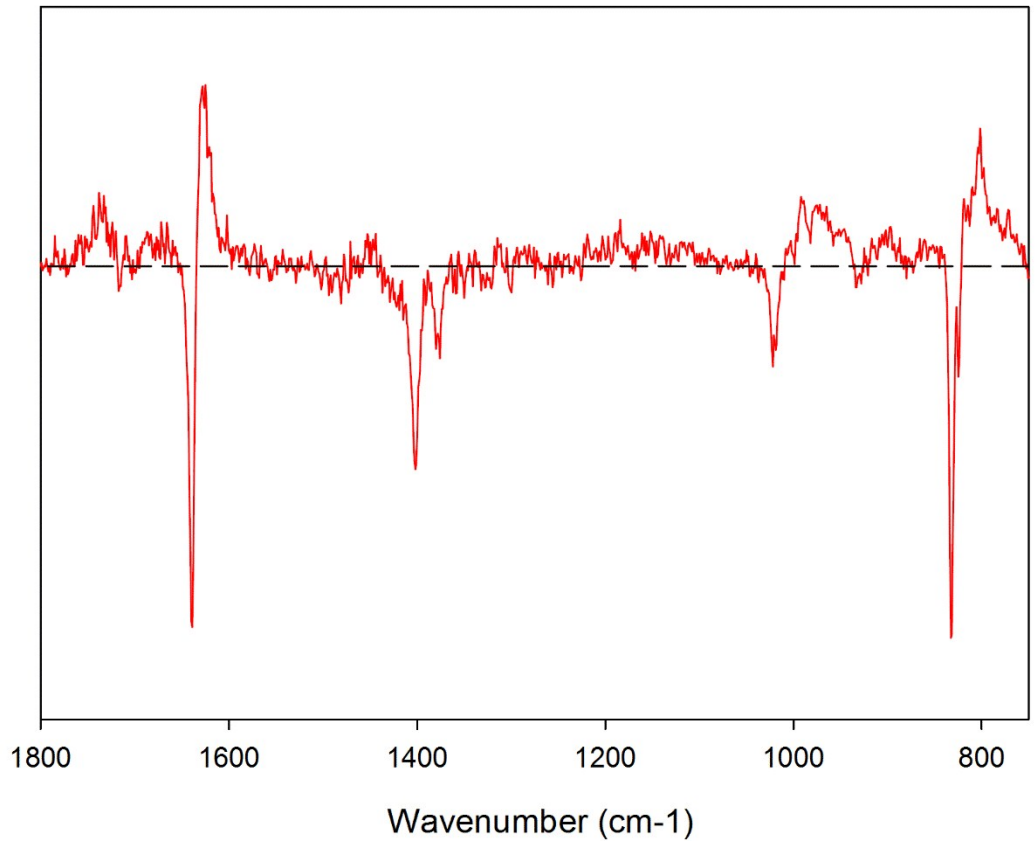

Figure S8. Loadings plot for model based on MMA-A.

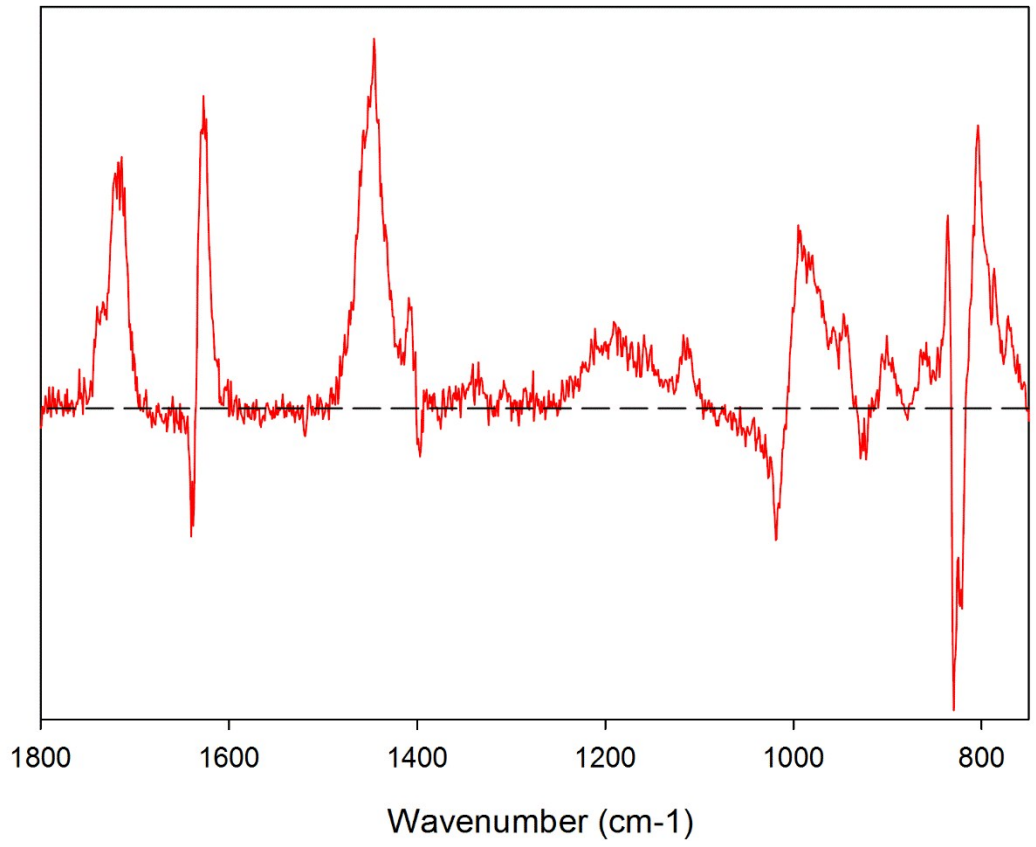

Figure S9. Loadings plot for model based on MMA-B.

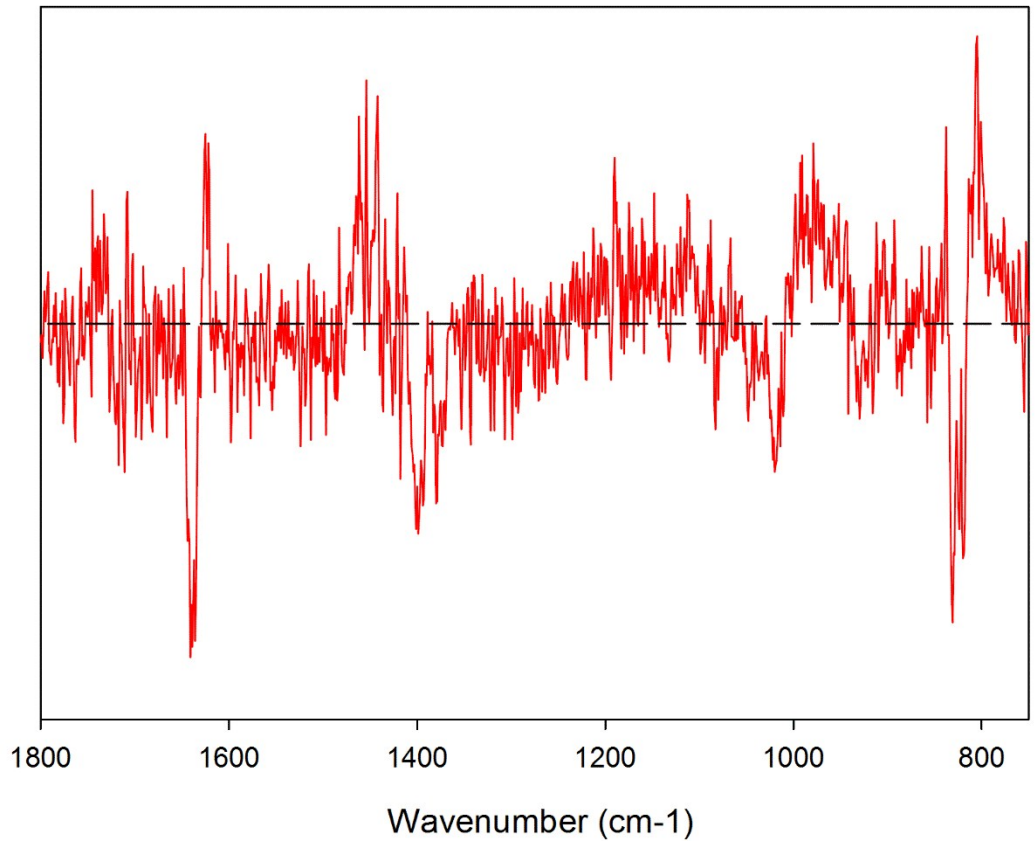

Figure S10. Loadings plot of model based for MMA-C.

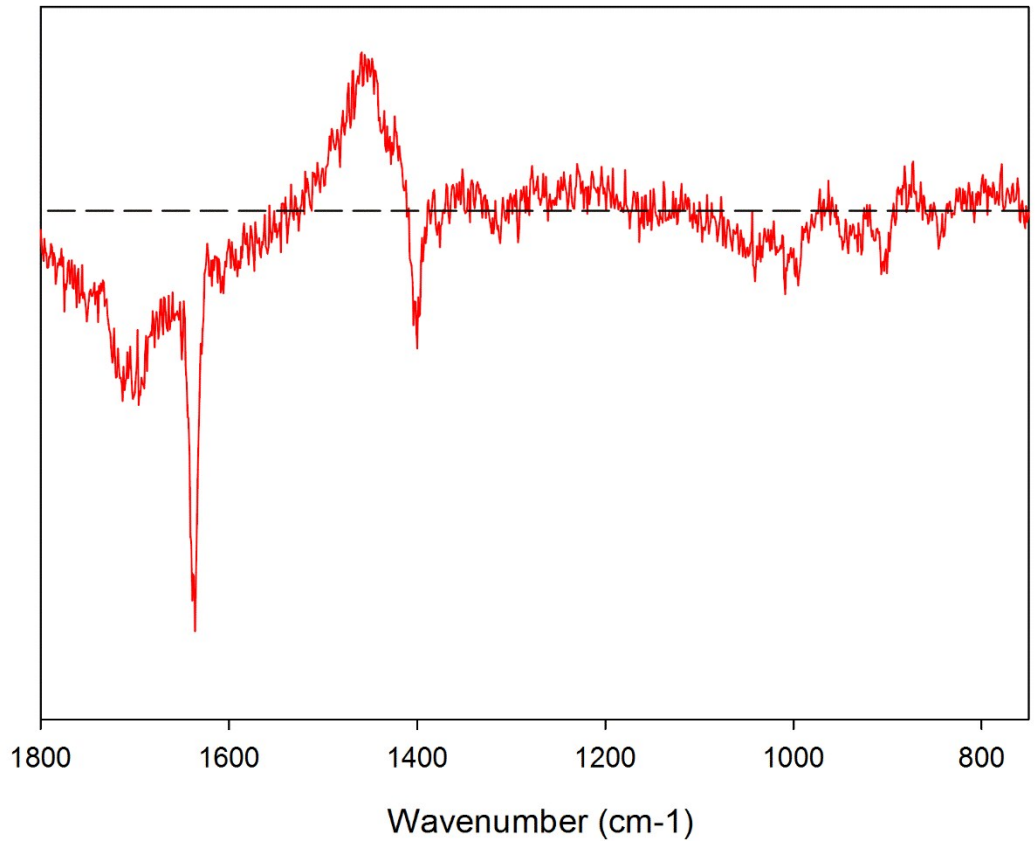

Figure S11. Loadings plot for model based on HEMA.

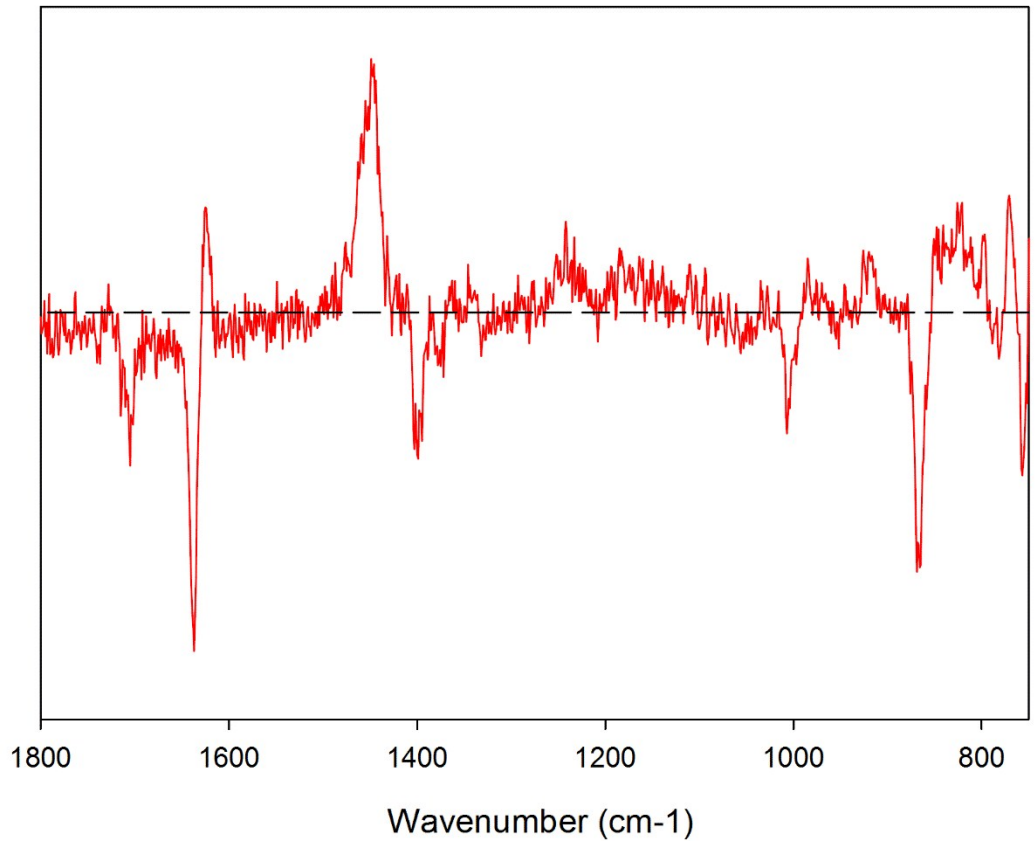

Figure S12. Loadings plot for model based on t-BMA.

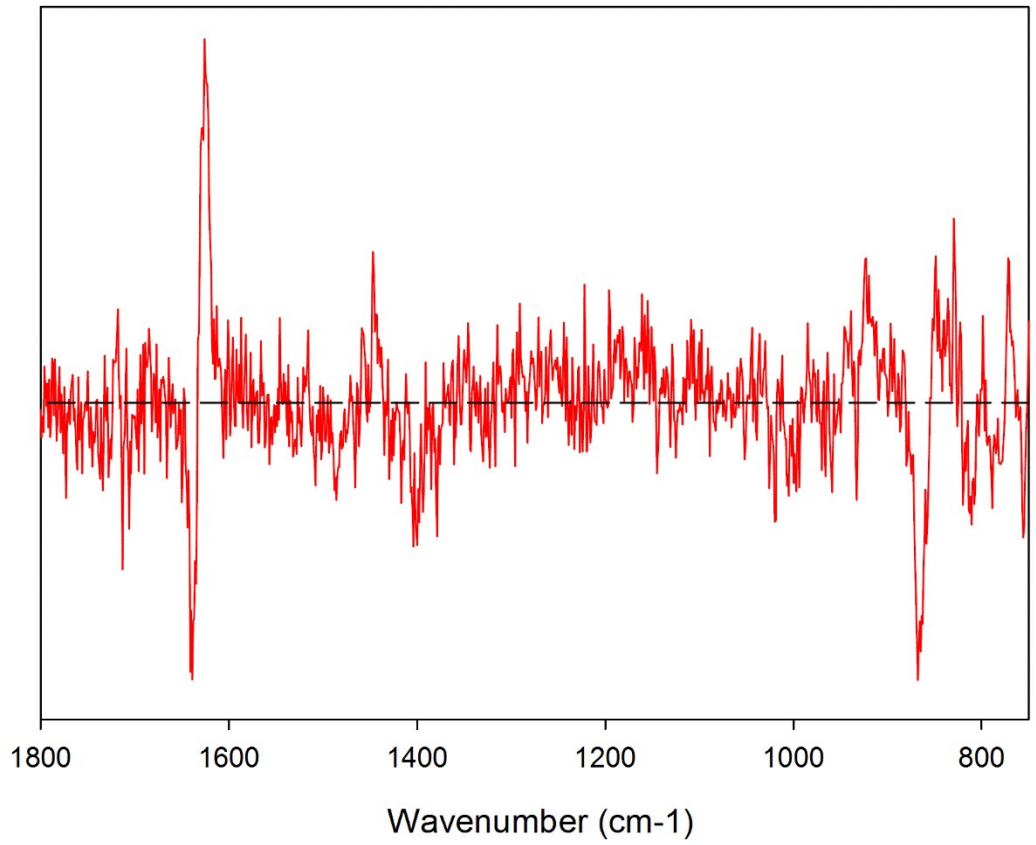

Figure S13. Loadings plot for model based on all monomers.

Matlab file for chemometrics calculations

%%Separate text into .m files before each function

%%Model generation is called using the PLSRCalibrate_FN_MC function, where entries is how many different folders of data are used.

%%spectra files should be .csv with the first column being x-values and second column y-values. Several files can be selected for each entry.

%%measurement files should be single column csv files with the same amount of entries and spectra loaded.

%%Other useful functions are calcRsq and calcRMSEList, to give a model R^2 value and a RMSE for each sample respectively.

function [RMSEList] = calcRMSEList(squaredErrorList, countSpectraTests)

%%Given an 2D list of squared errors for various sample, with several errors associated with each sample, and a list of how many errors per samples, this script calculates RMSE for each sample. If less than the maximum amount of errors is given, extra slots should be filled with zeros.

%determine number of samples and allocate empty matrices

[~,m] = size(squaredErrorList);

MSEList = zeros(m,1);

%determine RMSE for each sample from SE

for i = 1:m

  %add the errors for each value

  MSEList(i) = sum(squaredErrorList(:,i));

  %divide the summed errors by the count for a mean

```
    MSEList(i) = MSEList(i)/countSpectraTests(i);
end
%find the root of the mean squared error, list as one command
RMSEList = MSEList.^0.5;

function [rSq] = calcRSq(predictions, measurements)
%%Given a 1D list of predictions, and a 1D list of measurements, returns and R^2 value
SSResid = sum((measurements(:)-predictions(:)).^2);
SSTotal = sum((measurements(:)-mean(measurements)).^2);
rSq = 1 - SSResid/SSTotal;

function measurements = loadMeasurements()
%%Loads measurement file, singular, .csv
uiwait(msgbox('Select measurement file'));
[FileName,PathName,~] = uigetfile('*');
measurements = csvread(char(strcat(PathName, FileName)), 0, 0);

function [cutMatrix, Raman] = loadSpectra()
%%Loads spectra, one or many.
%collects filenames and path
uiwait(msgbox('Select spectra fileset'));
[FileName,PathName,~] = uigetfile('*', 'MultiSelect','on');
%creates some black matrices
cutMatrix = [];
%checks for one or multiple files
if ischar(FileName)
    matrix = csvread(char(strcat(PathName, FileName)), 0, 1);
```

```
    cutMatrix = matrix(matrix~=0); %%first two lines give the cutMatrix variable all the non-zero y values values

    [n,m] = size(matrix);

    allRaman = csvread(char(strcat(PathName, FileName)), 0, 0, [0, 0, (n-1), 0]);%Loads the x axis values into a variable

else %if multiple files, iterates through them adding them all to a 2D matrix, without the zeros

    for i = 1:length(FileName)

        mat = csvread(char(strcat(PathName, FileName(i))), 0, 1);

        matrix(:,i) = mat; %saved for cross-referencing x axis

        cutMatrix(:,i) = mat(mat~=0);

    end

    [n,m] = size(matrix);

    allRaman = csvread(char(strcat(PathName, FileName(1))), 0, 0, [0, 0, (n-1), 0]);

end

k = 1;

Raman = zeros(n,1);

for j = 1:n

    if matrix(j,1) ~= 0

        Raman(k) = allRaman(j);

        k=k+1;

    end

end

Raman = Raman(Raman~=0);


function [normMatrix] = normCbR(matrix, row)

%%Spectra is passed in as the variable matrix to normalise, and variable row

%%is the row to normalise off base on the first column
```

```
normMatrix = matrix;


%Repeating for all the columns
for i = 2:size(matrix, 2)
   %Calculating factor to multiply columns by
   normFactor = matrix(row, 1)/matrix(row, i);
   %Mutiplication of each row in column i by the factor
   for j = 1:size(matrix, 1)
      normMatrix(j, i) = matrix(j, i) * normFactor;
   end
end


function [loadingsVector, predictions, percentVariance, bestComp, eqFit,
squaredErrorByValue, countSpectraTests, predictionsByValue, cumRMSEList, lowRMSE,
xNorm, compRMSEList] = PLSRCalibrate_FN_MC(entries)
%%Calibrate the Partial Least Squares Regression
%Load the calibrated samples
%%Change the numerical values as required based on spectra
cumRMSEList = ones(1,1800); %%this should remain as "1,highest value"
[spectra, xValues] = loadSpectra();
xAxis = [750:1800];  %This range should be whatever the x axis of spectra is, in increments
for model (in this case increment is 1).
spectra = interp1(xValues, spectra, xAxis');
if entries > 1
   for a = 1:(entries-1);
      [tempSpectra, xValues] = loadSpectra();
      tempSpectra = interp1(xValues, tempSpectra, xAxis');
      spectra = [spectra tempSpectra];
```

```
    end
end
xValues = xAxis';
measurements = loadMeasurements();
parfor normValue = xAxis
    %calculate closest value to normalise on
    [~, indexNorm] = min(abs(xValues-normValue));
    %normalise the spectra base on value
    nSpectra = normCbR(spectra, indexNorm);

    [~, ~, ~, ~, ~, ~, RMSE, ~, ~, ~] = testPLSComponentsMC(nSpectra, measurements);
    cumRMSEList(normValue) = RMSE;
end

%collecting RMSE from the trials now becomes obsolete, collecting the
%lowRMSE from another model creaction based on the optimised settings

[~, xNorm] = min(cumRMSEList);

[~, indexNorm] = min(abs(xValues-xNorm));
nSpectra = normCbR(spectra, indexNorm);
[loadingsVector, predictions, percentVariance, bestComp, eqFit, compRMSEList, lowRMSE,
squaredErrorByValue, countSpectraTests, predictionsByValue] =
testPLSComponentsMC(nSpectra, measurements);

function [loadingsVector, percentVariance, predictions, RMSEModel, squaredErrorByValue,
countSpectraTests, predictionsByValue] = testCalibrationMC(spectra, measurements,
components, trials)
%%Tests the calibration leaving out 1/3 of the samples, selected on random
```

```
%%x times. Monte carlo method.


[~,numSpectra] = size(floor(spectra));

MCValidationSize = round(numSpectra/3); %change from 3 if you want to leave a smaller or
bigger fraction out


predictions = zeros(MCValidationSize,trials);

squaredError = zeros(MCValidationSize,trials);

error = zeros(MCValidationSize,trials);


squaredErrorByValue = zeros(trials,numSpectra);

errorByValue = zeros(trials,numSpectra);

predictionsByValue = zeros(trials,numSpectra);

countSpectraTests = zeros(numSpectra, 1);

for a = 1:trials;
   %%Generate how many spectra to remove and which ones will be removed,
   %%no repeats
   MCValidation = randi(numSpectra, MCValidationSize,1);
   [tempSize1,~] = size(MCValidation);
   [tempSize2,~] = size(unique(MCValidation));
   while tempSize1 > tempSize2
      MCValidation = randi(numSpectra, MCValidationSize,1);
      [tempSize1,~] = size(MCValidation);
      [tempSize2,~] = size(unique(MCValidation));
   end


   testSpectra = spectra(:,MCValidation);
   calibrateSpectra = spectra;
```

```
    calibrateSpectra(:,MCValidation) = [];

    testMeasurements = measurements(MCValidation);

    calibrateMeasurements = measurements;

    calibrateMeasurements(MCValidation) = [];


    [~,~,~,~,loadingsVector,percentVariance,~] = plsregress(calibrateSpectra',
calibrateMeasurements, components);

    predictions(:,a) = [ones(MCValidationSize,1) testSpectra']*loadingsVector;


    squaredError(:,a) = (predictions(:,a)-testMeasurements).^2;

    error(:,a) = predictions(a)-testMeasurements;




    countSpectraTests(MCValidation) = countSpectraTests(MCValidation)+1;



squaredErrorByValue(sub2ind(size(squaredErrorByValue),countSpectraTests(MCValidation)
,MCValidation)) = squaredError(:,a);


errorByValue(sub2ind(size(errorByValue),countSpectraTests(MCValidation),MCValidation))
= error(:,a);


predictionsByValue(sub2ind(size(predictionsByValue),countSpectraTests(MCValidation),M
CValidation)) = predictions(:,a);

end

RMSEModel = sqrt((sum(squaredErrorByValue(:)))/sum(countSpectraTests));

[~,~,~,~,loadingsVector,~,~] = plsregress(spectra', measurements, components);


function [loadingsVector, avPred, percentVariance, bestComp, eqFit, cumRMSEList, RMSE,
squaredErrorByValue, countSpectraTests, predictionsByValue] =
testPLSComponentsMC(nSpectra, measurements)
```

```
[~,numSpectra] = size(round(nSpectra));


maxComp = floor(((3-1)*numSpectra/3)-1); %change from 3 here if 1/3 is not used as
validation, "4" would be used if 1/4 of samples were used for validation

cumRMSEList = zeros(1,maxComp);


bestComp = maxComp;


parfor components = 1:(maxComp)

   [~, ~, ~, RMSEModel, ~, ~, ~] = testCalibrationMC(nSpectra, measurements, components,
(numSpectra*3));

   cumRMSEList(components)= RMSEModel;


end


%r = 0.9 type model starts here


for components = 1:(maxComp-1)

   rValue= (cumRMSEList(components+1))/(cumRMSEList(components));

   if rValue > 0.95 %This value can be tweaked to select more/less constricted models

      bestComp = components;

      break

   end

end

%model selection ends here

[loadingsVector, percentVariance, ~, RMSE, squaredErrorByValue, countSpectraTests,
predictionsByValue] = testCalibrationMC(nSpectra, measurements, bestComp,
```

```
(numSpectra*3)); % can change 3 to a higher or lower number to increase the interations of montecarlo validation.

sumPred = sum(predictionsByValue)';

avPred = sumPred./countSpectraTests;

[eqFit,~,~] = fit(measurements, avPred, 'Poly1');
```