```
Data Structures for modeling sc for scattering simulation and sp chromophore data
sc.optical_properties = 1;
sc.sm size = ; %smoothing parameter
sc.wv = []; %sequence of excitation wavelengths
sc.range = []; %sequence of spatial frequencies evaluated
sc.model = 'mc'; % 'mc': Monte Carlo or 'diff': Diffusion
sc.mc lut = [lookup table dir 'LUT.mat']; %
sc.p_{\overline{2}_p} = 1;
sc.bin size = 1;
sc.mua g = ; %100x less than musp 1/mm
sc.musp_g = ; %100x greater than mua 1/mm
sc.A_g = ; %Amplitude guess inverse mm
sc.b_g = ; %Scattering Power guess
sc.max A = ; %Max Amplitude inverse mm
sc.max b = ; %Max Power inverse mm
sc.fix A = ; %0 fixed, 1 initial guess, 2 average amplitude
sc.fix b = 0; %0 fixed, 1 initial guess, 2 average size
%Pharmacokinetic parameter validation from pathology
sc.vessel = 0.10; %average vessel diameter, in microns
save([sc_dir sc.model '_' sc.method '_sc.mat'],'calc');
sp.filename = ''; %tissue chromophore data
sp.fit = [ ]; %Select chromophores 1=on, 0=off
sp.names = {'Hb02' 'Hb' 'Edema' 'Hypoxia' 'FAD' 'NADH' 'PSI-chlorin' 'PSII-Ruth'};
sp.mult = [ ]; % factor
sp.units = { }; % units
save([sp_dir 'sp_.mat'], 'sp_');
% Data sphere - can we eliminate correlates within the data...
% Inputs:
% train Ref vec - Region of Interest vector
% train Tar vec - Pathology-verified classification vector
%Outputs
% new Ref - New patterns
% train Tar - New targets
% Aw - Filtered matrix
% means - Means vector
function [new Ref, new Tar, Aw, means] = sphere F(train Ref vec,
train Tar vec)
[r,c] = \overline{size}(train p);
means = nanmean(train p')';
stds = nanstd(train_p')';
new_p = train_p - means*ones(1,c); % subtract mean
cov mat = nancov(new p',1);
Aw = inv(sqrtm(cov mat)); % normalize to the inverse square of parameter covariance
if(isreal(Aw))
new p = (Aw*new p);
else
new_p = new_p./repmat(stds,[1 size(new_p,2)]);
end
end
% Randomly divide data for cross validation
% INPUT:
% new ref - data [#parameters #pixels]
% new tar - diagnostic labels [#pixels]
% OUTPUT:
% C1,C2,C3 = three groups of vectors
% D1,D2,D3 = corresponding labels for each group
***
```

```
function [C1, C2, C3, D1, D2, D3] = genCV(new p, new tar)
%Classes
Dx = unique(new tar);
% Randomized pixels from all samples (so patient data interspersed)
npix = size(new p, 2);
ipix = randperm(npix);
rtargets = new tar(ipix);
rpatterns = new_p(:,ipix);
%Divide patterns into three groups
C1 = []; C2 = []; C3 = [];
D1 = []; D2 = []; D3 = [];
for iDx = 1:length(Dx)
%Indexing the size of the three groups
inds = find(rtargets==Dx(iDx));
count = ceil(length(inds)/3);
if (mod (length (inds), 3) == 1)
I1 = count; I2 = count*2; I3 = count*3-2;
elseif (mod(length(inds),3) == 2)
I1 = count; I2 = count*2-1; I3 = count*3-1;
else
I1 = count; I2 = count*2; I3 = count*3;
end
%Pixels
Dx d = rpatterns(:,inds);
C1 = [C1 Dx d(:,1:I1)];
C2 = [C2 Dx d(:,I1+1:I2)];
C3 = [C3 Dx_d(:,I2+1:I3)];
%Labels
D1 = [D1 \text{ ones}(1, length(1:I1))*Dx(iDx)];
D2 = [D2 \text{ ones}(1, length(I1+1:I2))*Dx(iDx)];
D3 = [D3 \text{ ones}(1, length(I2+1:I3))*Dx(iDx)];
end
k = ; %Choose any between 1:15
%Select test & training sets
test = C1;
dx testing = D1;
train = [C2 C3];
dx training = [D2 D3];
%% Outlier removal
% INPUT:
% training - training dataset
% dx training - selected labels
% para - data structure containing parameter name and units as strings
% OUTPUT
% training - data with outliers removed
% training dx - labeled data
% Outs - # of outliers removed
function [IQtrain, dx_training, Outs] = rmIQR(train,dx_training,para,pnm)
[Q] = prctile(training,[],2); %Place range within brackets
Q1 = Q(:,1);

Q3 = Q(:,2);
IQR = Q3-Q1;
lbound = Q1-1.5*IQR;
ubound = Q3+1.5*IQR;
pind = find(strcmp(param.name,pnm));
outlier idx1 = find(training(pind,:) < lbound(pind));</pre>
outlier_idx2 = find(training(pind,:)>ubound(pind));
IQtrain = training;
IQtrain(:,[outlier idx1 outlier idx2]) = [];
```

```
dx_training([outlier_idx1 outlier_idx2]) = [];
outs = length([outlier_idx1 outlier_idx2]);
end

NNout = knnsearch(IQtrain', test', 'K',k);
```