```
%%%%%%%%%%
%1. select data that exclude NaNs to train SOM
%%%%%%%%%%
cd('\\fs1-per.nexus.csiro.au\{oa-waimos}\work\IMOS\TB\DAP\Test\MAT')
load Data_Temp_5m_extend_bottom

%%%%% Use a block of missing data
timen=datenum(2010,1,1):1:datenum(2023,5,1);

ts_md=datenum(2020,1,1);
ids=find(timen==ts_md);

te_md=ts_md+150;
ide=find(timen==te_md);

% % %%TR20
% % sss=39; % 25m sensor
% % dss=69; %175m sensor
%TR20_obs=T_new(ids:ide,sss:dss);

% % %%TR10
sss=18; % 25m sensor
dss=31; %90m sensor

% % sss=13; % SST
% % dss=33; %100m bottom

%TR10_obs=T_new(ids:ide,sss:dss);

% % %%ROT
% % sss=1; % SST
% % sss=6; % 25m sensor
% % dss=12; %55m sensor
% % ROT_obs=T_new(ids:ide,sss:dss);

load Fre_sl
timen=datenum(2010,1,1):1:datenum(2023,5,1);

 %=========================================================================================
%The map cannot understand  that day of year 365 is close to day of year 1 in space
% Thus we use a poor-man's quaterion transform and give the map the cos and sin of the doy
%=========================================================================================
doy=doy(timen);
cos_doy = cos((pi/180.0)*doy)';
sin_doy = sin((pi/180.0)*doy)';
DoY=cat(2,cos_doy,sin_doy);



DAT_T_ori=cat(2,T_new,DoY,fre);
%%%DAT_T_ori(n_train+1:end,:)=NaN; % 20% of input data are NaNs

%DAT_T_ori(ids:ide,sss:dss)=NaN; % from 1/1/2020 to 30/5/2020 (150 days~5 months~1 deployment,
```

```
DAT_T_ori(ids:ide,sss:dss)=NaN; % from 1/1/2020 to 30/5/2020 (150 days~5 months~1 deployment, c
%DAT_T_ori(ids:ide,sss:dss)=NaN; % from 1/1/2020 to 30/5/2020 (150 days~5 months~1 deployment,
DAT_orig=DAT_T_ori;
DAT_use=DAT_orig;


rows=any(isnan(DAT_use),2); % find any rows that had nan
DAT_use(rows,:)=[]; % remove any rows that had nan. Meaning that, we only use full data profile

[n_points,n_vars] = size(DAT_use);
%%%%%%%%%%%
% 2. Use SOM to train the map
%%%%%%%%%%%

% Fixed parameters for the training of the map
% We've tested numerous values here and the work well, but they
% can be modified without fear of breaking the script
radius_left = [6,4,2,1,0.5,0.1];
radius_right = [4,2,1,0.5,0.1,0.0001];
training_length = [60,100,200,200,140,60];

n_munits = 1000; %number of classes
%n_munits = 500; %number of classes
%=================================================================%

Sobs=som_data_struct(DAT_use(:,:));
Sobsn=som_normalize(Sobs,'var');                %Normalize the data

%replace NAN's in normalized data with 0
% % [id_nan] = find(isnan(Sobs.data));
% % Sobsn.data(id_nan) = 0;

%Make a map object
sMapobs=som_make(Sobsn,'munits',n_munits);

for i = 1:length(radius_left)

    %Iteratively train the SOM with progressively longer training lengths and smaller influence
    sMapobs=som_batchtrain(sMapobs,Sobsn,'radius',[radius_left(i) radius_right(i)],'trainlen',t

end %for i = 1:length(radius_left)

%Get the hits map and the best matching units
hits=som_hits(sMapobs,Sobsn);
bmus=som_bmus(sMapobs,Sobsn);

%Set the data matrix to NaN places where the best-matching units are empty
for i =1:size(Sobsn.data,1)
        Sobsn.data(i,isnan(Sobsn.data(i,:)))=sMapobs.codebook(bmus(i),isnan(Sobsn.data(i,:)));
end

        %=============================================================================
        %=============================================================================
        %LOCAL CORRELATIONS IN DATA SPACE
```

```
        % Here, we apply Eqn. 1 in C&C 2016 to the trained map.
        %DAT_COR_it = 99*ones([size(sMapobs.codebook),nit_vars]); %

        DAT_COR_it = 99*ones([length(sMapobs.codebook),n_vars]); %original code

        for i=1:length(sMapobs.codebook) %nc_it
            if ~isempty(Sobsn.data(bmus==i,:))
                %Calculate the local correlations in the data space
                corr_train=corr(Sobsn.data(bmus==i,:));
                %DAT_COR_it(i,:)=1+ sqrt(sum(corr_train(:,1:m_vel).^2,2)); % only use velocity
                DAT_COR_it(i,:)=1+ sqrt(sum(corr_train(:,1:n_vars).^2,2)); % use all variables

            end %if isempty
        end %for i=1:length(sMapobs.codebook)
%%%%%%%%%%%
 %3. Work with original data including NaNs, but keep the same SOM classes, use SIM to update B
 %Set up a new map, this time initialised with the test data, we the same topolog as the previo
 %%%%%%%%%%%
DAT_T_o=cat(2,T_new,DoY,fre);
 %DAT_fill=DAT_orig;
DAT_fill=DAT_T_o;
%DAT_fill(ids:ide,sss:dss)=NaN; %from 1/1/2020 to 30/5/2020 (150 days~5 months~1 deployment, da
DAT_fill(ids:ide,sss:dss)=NaN; %from 1/1/2020 to 30/5/2020 (150 days~5 months~1 deployment, dat
%DAT_fill(ids:ide,sss:dss)=NaN; %from 1/1/2020 to 30/5/2020 (150 days~5 months~1 deployment, da
[n_fill] = find(isnan(DAT_fill)); % n_fill include NaN from original obs and TR10

% % replace NaN's with 0
% %DAT_fill(n_fill) = 0;

Snew=som_data_struct(DAT_fill(:,:));
Snew_norm=som_normalize(Snew,sMapobs.comp_norm);
Snew_norm.data(n_fill)=NaN;

data_150=Snew_norm.data(ids:ide,:);


 hits=som_hits(sMapobs,Snew_norm);
 bmus=som_bmus(sMapobs,Snew_norm);


for i =1:length(Snew_norm.data)
    %i =1:size(Snew_norm.data,1)
    Snew_norm.data(i,isnan(Snew_norm.data(i,:)))=sMapobs.codebook(bmus(i),isnan(Snew_norm.data(
end %for i =1:size(Snew_norm.data,1)


for i=1:length(sMapobs.codebook)
        %Find the weighted distance in data space
        DDist(i,:)= sum(((repmat(sMapobs.codebook(i,:),length(Snew_norm.data),1)-Snew_norm.data

end %i=1:length(sMapobs.codebook)

%The new best matching unit is with the smallest weighted distance in data space for the inputs
for i=1:length(Snew_norm.data)
```

```
    wow=find(DDist(:,i)==min(DDist(:,i)));
    bmus_new(i)=wow(1);
end

bmus_150=bmus(ids:ide);
bmus_150_update=bmus_new(ids:ide)';

%%%%%%%%%%%
% 4. Fill data
 %sMap1 = som_denormalize(sMapobs,Sobsn);
%%%%%%%%%%%
 Snew_norm.data(n_fill)=NaN;
 for i =1:size(Snew_norm.data,1)
    Snew_norm.data(i,isnan(Snew_norm.data(i,:)))=sMapobs.codebook(bmus_new(i),isnan(Snew_norm.d
end %for i =1:size(Snew_norm.data,1)

 sMap1 = som_denormalize(sMapobs,Snew_norm);

 m_tmp=74;
 temp_pred = nan*Snew.data(:,1:m_tmp);
 for isub = 1:length(bmus_new)
    temp_pred(isub,[1:m_tmp])=sMap1.codebook(bmus_new(isub),[1:m_tmp]);
 end

 Data_obs=T_new(ids:ide,sss:dss);
 Data_pred=temp_pred(ids:ide,sss:dss);
 [ro,co]=size(Data_obs);


 D_pred1=reshape(Data_pred,[ro*co 1]);
 D_test1=reshape(Data_obs,[ro*co 1]);

 Tlinear_fit = fitlm(D_test1,D_pred1);
 R_square=Tlinear_fit.Rsquared.Ordinary
 T_RMSE=Tlinear_fit.RMSE
```