```matlab
function out=PreDREEG(T,n,all_data)
%PREDREEG Dimension Reduction on n minute EEG on table T
%   Get T from PreRecordAnalysis
%
% Runs dimension reduction from table T with location field. This includes cleaning the↙
data and running a subset of features
% n denotes the amount of minutes the EEG is segmented into
% All_data (default 0) can be set to 1 to run a larger collection of features
%
%   Authors:
%       Michael Caiola (Michael.Caiola@fda.hhs.gov)
%       Meijun Ye (Meijun.Ye@fda.hhs.gov)
%
%   Disclaimer: This software and documentation (the "Software") were
%   developed at the Food and Drug Administration (FDA) by employees of
%   the Federal Government in the course of their official duties.
%   Pursuant to Title 17, Section 105 of the United States Code,
%   this work is not subject to copyright protection and is in the
%   public domain. Permission is hereby granted, free of charge, to any
%   person obtaining a copy of the Software, to deal in the Software
%   without restriction, including without limitation the rights to
%   use, copy, modify, merge, publish, distribute, sublicense, or sell
%   copies of the Software or derivatives, and to permit persons to
%   whom the Software is furnished to do so. FDA assumes no
%   responsibility whatsoever for use by other parties of the Software,
%   its source code, documentation or compiled executables, and makes
%   no guarantees, expressed or implied, about its quality,
%   reliability, or any other characteristic. Further, use of this code
%   in no way implies endorsement by the FDA or confers any advantage
%   in regulatory decisions. Although this software can be
%   redistributed and/or modified freely, we ask that any derivative
%   works bear some notice that they are derived from it, and any
%   modified versions bear some notice that they have been modified.
%   The Software is not intended to make clinical diagnoses or to be
%   used in any way to diagnose or treat subjects for whom the EEG is
%   taken.

if nargin < 3
    all_data = 0;
end
path = pwd;
cd ../../eeglab/ %replace with eeglab path
eeglab
close
cd(path)
g=[];F=[];name_list=[];cnt=0;
%h=waitbar(0,'Reticulating Splines');
WaitMessage = parfor_wait(height(T), 'Waitbar', true,'ReportInterval', 1);
for i=1:height(T)
    s=split(T.Location{i},'\');
    s=string(join(s(1:end-1),'\'));
    d=dir(fullfile(s,'*.edf'));
    names=string();
    for j=1:length(d)
        names(j)=string(fullfile(d(j).folder,d(j).name));
    end
    [rawEEG,join_flag,cnt1]=joinEEG(names);
    cnt = cnt + cnt1;
    disp("FIXED " + cnt1 + " RECORDS!")
    if isempty(rawEEG.data)
        %clear names
        warning(['Bad EDF: ',d(j).folder])
        continue
    end
    data=cutEEG(rawEEG,n);
    if isempty(data)
        %clear names
        warning(['Bad EDF: ',d(j).folder])
        continue
    end
    %clear names
    try
        clndata=filtEEG(rawEEG,data);
    catch
        warning(['Dirty EDF: ',d(j).folder])
        continue
    end
    newnames=[string(repmat(d(j).folder,[size(data,3),1])),join_flag+(1:size(data,3))'];
    name_list=[name_list;newnames];
    if ~all_data
        %calculate some quick features
        newF=Feats(clndata);
    else
        %calculate all Feats
        newF = BigFeats(clndata);
    end
    F=[F, newF];
    newg=repmat(i,[size(data,3),1]);
    newg(any(isnan(newF)))=NaN;
    g=[g;newg];
    %waitbar(i/height(T),h)
    WaitMessage.Send;
    pause(0.002);
end
%close(h)
WaitMessage.Destroy;
disp("Fixed " + cnt + " Total Files!")
F=F';
if ~all_data
    [Z,loss3]=tsne(F,"NumDimensions",3);
    g(isnan(g))=[]; %may need to be fixed
    c=lines(max(g));
    figure;
    scatter3(Z(:,1),Z(:,2),Z(:,3),15,c(g,:),'filled');
    title('tSNE')
    hold on
    scatter3(mean(Z(:,1)),mean(Z(:,2)),mean(Z(:,3)),50,'k','filled');
    %plotcube(2*std(Z),mean(Z)-std(Z),.1);
    %plotcube(4*std(Z),mean(Z)-2*std(Z),.1);
    % figure;
    % scatter3(X(:,1),X(:,2),X(:,3),15,c(g,:),'filled');
    % title('PCA')
end
out.F=F;
out.names=name_list;
out.min=n;
out.T=T;
end
function [rawEEG,out,cnt]=joinEEG(names)
out=0;
cnt = 0;
%if verLessThan('matlab','9.9')
try
    [rawEEG,cnt1] = FindCh(names{1},1);
    cnt = cnt+cnt1;
%     if cnt1
%         disp("Fixed " + cnt + " records!")
%     end
    param_flag = 0;
catch
    rawEEG.comments=[];
    rawEEG.data=[];
    param_flag=1;
end
%else
%end
for k=2:length(names)
    try
        [newEEG,cnt1] = FindCh(names{k},0);
        cnt = cnt+cnt1;
%         if cnt1
%             disp("Fixed " + cnt + " records!")
%         end
        if param_flag
            rawEEG=newEEG;
            out=1;
            param_flag=0;
        else
            rawEEG.comments=[rawEEG.comments;newEEG.comments];
            rawEEG.data=[rawEEG.data,newEEG.data];
        end
    catch
    end
```

```matlab
end
if param_flag
    rawEGG=[];
else
    rawEEG.data=rawEEG.data(:,1:250*60*(floor(length(rawEEG.data)/250/60)-1));
    rawEEG.xmax=length(rawEEG.data)/250;
    rawEEG.pnts=length(rawEEG.data);
    rawEEG.times=0:4:4*(rawEEG.pnts-1);
end
end
function data=cutEEG(EEG,n)
cuts=floor(EEG.xmax/60/n);
data=zeros(19,n*60*250,cuts);
for i=1:cuts
    data(:,:,i)=EEG.data(:,1+(i-1)*n*60*250:i*n*60*250);
end
end
function [clndata]=filtEEG(EEG,data)
for i=1:size(data,3)
    rawEEG=EEG;
    rawEEG.data=data(:,:,i);
    rawEEG.pnts=length(rawEEG.data);
    filtEEG = pop_eegfiltnew(rawEEG,1,100);
    EEG = pop_reref(filtEEG,[]);
    [~,W] = fastica(EEG.data,'verbose','off');
    EEG = pop_editset(EEG,'icaweights',W);
    EEG = iclabel(EEG);
    [~,ictype] = max(EEG.etc.ic_classification.ICLabel.classifications,[],2);
    icreject = find(ictype~=1);
    % EEG.reject.gcompreject(1,icreject') = 1;
    clean_EEG = pop_subcomp(EEG,icreject',0,0);

    %rawdata(:,:,i) = EEG.data;
    clndata(:,:,i) = clean_EEG.data;
end
end
function F=Feats(data)
srate=250;
tflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',4,'HalfPowerFrequency2',8, ...
    'SampleRate',srate);
aflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',8,'HalfPowerFrequency2',12, ...
    'SampleRate',srate);
gflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',25,'HalfPowerFrequency2',40, ...
    'SampleRate',srate);
F=zeros(380,size(data,3));
for j=1:size(data,3)
    stats = []; %mean, max, min, std 76 values
    spectent = []; % 19 Values
    PACag = []; % 19 Values
    PACtg = []; % 19 Values
    PACta = []; % 19 Values
    [abs_psd,rel_psd]=getPSD(data(:,:,j),srate);
    for i=1:19
        %% Spectral Entropy of each channel
        X = fft(data(i,:,j));
        S = abs(X).^2;
        P = S./sum(S);
        H = 0;
        for m = 1:length(P)
            H = H + P(m)*log2(P(m));
        end
        spectent = [spectent,-H];
        PACag = [PACag, getPAC(data(i,:,j),aflt,gflt)];
        PACtg = [PACtg, getPAC(data(i,:,j),tflt,gflt)];
        PACta = [PACta, getPAC(data(i,:,j),tflt,aflt)];
    end
    stats=[mean(data(:,:,j),2)',max(data(:,:,j),[],2)',min(data(:,:,j),[],2)',std(data↙
(:,:,j),[],2)'];
    F(:,j)=[stats,reshape(abs_psd,[1,19*6]),reshape(rel_psd,[1,19*6]),spectent,PACag,↙
PACtg,PACta];
end
end

function F=BigFeats(data)
srate=250;
F = Feats(data);
tflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',4,'HalfPowerFrequency2',8, ...
    'SampleRate',srate);
aflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',8,'HalfPowerFrequency2',12, ...
    'SampleRate',srate);
gflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',25,'HalfPowerFrequency2',40, ...
    'SampleRate',srate);
dflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',1,'HalfPowerFrequency2',4, ...
    'SampleRate',srate);
mflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',12,'HalfPowerFrequency2',16, ...
    'SampleRate',srate);
bflt = designfilt('bandpassiir','FilterOrder',6, ...
    'HalfPowerFrequency1',16,'HalfPowerFrequency2',20, ...
    'SampleRate',srate);
RCH=zeros(size(data,3),171*6);
for j=1:size(data,3)
    tdata = zeros(size(data,1),size(data,2));
    adata = zeros(size(data,1),size(data,2));
    mdata = zeros(size(data,1),size(data,2));
    bdata = zeros(size(data,1),size(data,2));
    gdata = zeros(size(data,1),size(data,2));
    ddata = zeros(size(data,1),size(data,2));

    for ch = 1:19
        tdata(ch,:) = filter(tflt,data(ch,:,j));
        adata(ch,:) = filter(aflt,data(ch,:,j));
        mdata(ch,:) = filter(mflt,data(ch,:,j));
        bdata(ch,:) = filter(bflt,data(ch,:,j));
        gdata(ch,:) = filter(gflt,data(ch,:,j));
        ddata(ch,:) = filter(dflt,data(ch,:,j));
    end

    tCH = getCOH(tdata,srate);
    aCH = getCOH(adata,srate);
    mCH = getCOH(mdata,srate);
    bCH = getCOH(bdata,srate);
    gCH = getCOH(gdata,srate);
    dCH = getCOH(ddata,srate);

    RCH(j,:) = [tCH,aCH,mCH,bCH,gCH,dCH];
end
F = [F; RCH'];
end

function C = getCOH(data,srate)
    ind = [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,...
        0,  0,  0,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,...
        1,  1,  1,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,...
        2,  2,  2,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,...
        3,  3,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,...
        5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  6,  6,  6,...
        6,  6,  6,  6,  6,  6,  6,  6,  6,  7,  7,  7,  7,  7,  7,  7,...
        7,  7,  7,  7,  8,  8,  8,  8,  8,  8,  8,  8,  8,  8,  9,  9,...
        9,  9,  9,  9,  9,  9,  9, 10, 10, 10, 10, 10, 10, 10, 10, 11,...
       11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13,...
       13, 14, 14, 14, 14, 15, 15, 15, 16, 16, 17; 1,  2,  3,  4,  5,...
        6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,...
       17, 18,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,...
       16, 17, 18,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,...
       16, 17, 18,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,...
       17, 18,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18,...
        6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18,  7,  8,  9,...
       10, 11, 12, 13, 14, 15, 16, 17, 18,  8,  9, 10, 11, 12, 13, 14,...
       15, 16, 17, 18,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 10, 11,...
       12, 13, 14, 15, 16, 17, 18, 11, 12, 13, 14, 15, 16, 17, 18, 12,...
       13, 14, 15, 16, 17, 18, 13, 14, 15, 16, 17, 18, 14, 15, 16, 17,...
       18, 15, 16, 17, 18, 16, 17, 18, 17, 18, 18]+1;
   C = zeros(1,length(ind));

    for c = 1:length(ind)
```

```
        y = mscohere(data(ind(1,c),:),data(ind(2,c),:),...
                          srate*30,0,1:0.1:100,srate);
        C(c) = mean(y);
    end

end

function [abs_psd,rel_psd]=getPSD(data,srate)
% data: sample X channel
data=data';
fband=[1 4; 4 8; 8 12; 12 16; 16 20; 25 40];
tot=bandpower(data,srate,[1 srate/2]);

n=size(fband,1);

for i=1:n
    abs_psd(:,i)=bandpower(data,srate,fband(i,:));
end

ch=max(size(abs_psd,1));

for j=1:ch
    rel_psd(j,:)=abs_psd(j,:)./tot(j);
end
end
function PAC = getPAC(data,pflt,aflt)

fp = filter(pflt,data);
fp = hilbert(fp);
phi = angle(fp).*(180/pi);

fa = filter(aflt,data);
fa = hilbert(fa);
A = abs(fa);

edges = -180:20:180;
bin_phi = discretize(phi,edges);
A_mean = zeros(1,18);
for b = 1:18
    A_mean(b) = mean(A(bin_phi==b));
end


Pj = zeros(1,18);
for j = 1:18
    Pj(j) = A_mean(j)/sum(A_mean);
end
Dkl = zeros(1,18);
for k = 1:18
    Dkl(k) = log(Pj(k)/(1/18))*Pj(k);
end
```

```
PAC = sum(Dkl)/log(18);
end
%%
function [rawEEG,cnt] = FindCh(file,first)
cnt = 0;
load('chlocs2.mat','channel_locations');
ch_locs = struct2table(channel_locations);
ch_locs = string(ch_locs.labels);
if first
    rawEEG = pop_biosig(file,'blockrange',[60↙
Inf],'importevent','off','importannot','off');
else
    rawEEG = pop_biosig(file,'importevent','off','importannot','off');
end
chs = struct2table(rawEEG.chanlocs);
chs = string(chs.labels);
chs = erase(chs,"-REF"|"-LE");
in = [];
for i = 1:length(ch_locs)
    in1 = find(strcmpi(chs,ch_locs(i)));
    in = [in in1];
    if and(isempty(in1),ismember(ch_locs(i),["T3";"T4";"T5";"T6"]))
        switch ch_locs(i)
            case "T3"
                in2 = find(strcmpi(chs,"T7"));
            case "T4"
                in2 = find(strcmpi(chs,"T8"));
            case "T5"
                in2 = find(strcmpi(chs,"P7"));
            case "T6"
                in2 = find(strcmpi(chs,"P8"));
        end
        in = [in in2];
    end
end
if length(in)~=19
    error("Could not locate all channels!");
end
rawEEG.data = rawEEG.data(in,:);
rawEEG.chanlocs = channel_locations;
rawEEG.nbchan = 19;
if ~isequal(in,[1:16,19:21])
    cnt = 1;
end
end
```