% This script is the first step - extract epochs around the relevant events from the raw EEG data

% After going through all steps, check whether you get topographies and

% waveforms matching the ones I got, saved as pngs in this shared FullERPanalysis folder

% In this example dataset, subjects continuously monitored a dot stimulus for 2 min at a time, which contained

% 32 'target' periods of coherent motion that were either strong (80% coherence) or weak (40%), left or right, and had

% to click a button corresponding to motion direction (left hand for leftward, right hand for rightward) within 1.9 sec

% of target onset in order to get points. Dots were incoherent in between targets.

% Each subject performed 16 such blocks, 4 of each of 4 types that biased the number of one of the coherences OR of one of

% the directions, e.g. 'LD' blocks had 75% leftward direction, 25% rightward, 'LC' had 75% low coherence, 25% high coherence, and so on. For the purposes of

% this demonstration we are just going to compare high and low coherence trials, regardless of block type.

% Note: During the coherent-motion targets, coherence didn't step immediately to 40/80% but rather linearly ramped

% up over 480 ms - hence response times (RT) might be a bit slower than usual. Also, coherence during targets

% stepped up to 40/80% and down to 0% 18.75 times/second to evoke a steady-state EEG response at 18.75 Hz. Other than these biases in the numbers of

% targets and the ramping and oscillating of coherence, the paradigm is very similar to the one in Kelly & O'Connell (2013)

% This is an example dataset showcasing many of the ways you can get terrible data from not applying the

% electrodes and gel carefully , checking the channels before recording and in between blocks, and talking to

% the subject about keeping eye strictly on fixation, minimising blinks (or getting them to time them around a

% second after responding), etc.

% If you're using these scripts as a basis for your analysis of another dataset, remember to write your own

% comments and do not leave these here - it's terrible practice to leave in stale comments that refer to

% different datasets or analyses!

% The EEG data (bdf files) are in a google drive folder called 'Rose2019' - if you don't yet have access, let me know

% All scripts last revised by Simon on 26/11/21

clear

% list the subjects:

allsubj = {'P01', 'P02', 'P03', 'P04', 'P05', 'P06', 'P07', 'P08', 'P09', 'P10', 'P11', 'P12', 'P13', 'P14', 'P15', 'P16', 'P17', 'P18', 'P19', 'P20', 'P21', 'P22', 'P23', 'RL01'};

% What I tend to do is make this allsubj list once and keep it the same for every other script, so that the order of subjects is always the same

% If I want to come back here and re-run individual subjects, I change 'for s=1:length(allsubj)' in the main loop below to, e.g. 'for s==[4 17]'

% without changing allsubj or 'blocks' up here. If you want to do it another way, just be careful!

blockConds = {'HC' 'LC' 'LD' 'RD'}; % blocked conditions - many experiments will have more than one type of block that will have been run, and probably the EEG files would have been named accordingly

blocks = {[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]

[1:4] [1:4] [1:4] [1:4]};%... % list the block numbers for each block type of each subject. In this cell array, make one row per subject

% The EEG trigger codes we used for the different events were as follows:

% 1 - subject pressed to start after instructions

% 4 - fixation appeared

% 5 - start of each incoherent-motion inter-target interval (ITI)

% Target Triggers (coherent motion onset):

% 40% leftward target = 101;

% 40% rightward target =102;

% 80% leftward target = 103;

% 80% rightward target= 104;

targtrig = [101:104]; % list them in a vector so we can find them all below

% Since those four target triggers refer to a 2 x 2 crossing of the separate factors of direction and coherence, and we probably won't want

% to keep coming back to this comment to remember which is which, we will convert these target triggers into two separate vectors

% specifying motion and coherence respectively. Below, we'll do that by subtracting 100 from the target triggers 101-104 and then use

% the following 4-element vectors to translate them into motion direction and coherence:

trig2md = [1 2 1 2]; % Here we define 1=left, 2=right

trig2coh = [1 1 2 2]; % similarly, translating trigger code to coherence, where 1=low (40%), 2=high (80%)

% Response Triggers:

% 51 - participant has answered by pushing the left putton

% 53 - participant has answered by pushing the right putton

resptrig = [51 53];

% So list all the relevant triggers that we'll be concerned with (there might be others, like onset of fixation point at beginning of trial etc):

relevant\_triggers = [1 4 5 51 53 101:104];

anapar = struct(); % keep all analysis parameters in a structure that can be saved with the epoched data

anapar.minRT = 0.2; % in sec; what is the minimum RT we would regard as a genuine reactino to the target? In this task with contunuous dots and slowly ramping targets 0.2 sec is a conservative minimum. Other tasks would be different - e.g. in fast discrete tasks we might accept anticipatory responses before evidence onset as legitimate responses - think about what's appropriate for YOUR task

datafolder = 'H:\Backup\_UCDlabEEG\Rose/'; % in what directory are the subject folders containing the raw EEG (.bdf files)? (/ syntax may vary with OS)

bigmatfolder = 'bigmats/'; % Where to put the results? For each subject all single trial epochs are saved in a matfile, which can get big, so you might want to put this folder on an external harddrive

% EEG parameters:

fs = 1024; % sampling frequency

nchan=128; % number of EEG channels

next=8; % number of external channels (EOG etc)

% load in a structure 'chanlocs' containing the x,y,z locations of each of the 128 scalp channels in the cap

load chanlocsBioSemi128; % note this was made by calling>> readlocs('cap128.loc') , and, since the locations for the 8 externals (129:136) are all (1,0,0), getting rid of those by calling chanlocs = chanlocs(1:128)

anapar.LPF = 1; % 1 = low-pass filter the data, 0=don't.

% The low-pass filter designed below is a windowed sinc filter (see Semmlow 2004 signal proc book).

% It has a 3dB freq a bit lower than fc and it has specially strong attenuation at exactly 50Hz (mains in Ireland)

% Note that the below settings for fc and L were figured out by trial and error.

% For EEG with fs=512 Hz sample rate, use fc=38 and L=77

% For fs = 1024 Hz, use fc=37, L=137.

fc=37; % Low Pass Filter cutoff in Hz

% Get FIR filter weights 'LPK' for Hamming-windowed sinc filter (see Semmlow 2004 signal proc book):

wc = fc/fs\*2\*pi;

LPK=[]; % for low pass kernel

L=137;

for i=1:L

n = i-ceil(L/2);

if n==0, LPK(i)=wc/pi;

else, LPK(i) = sin(wc\*n)/(pi\*n); end

end

LPK = LPK.\*hamming(L)';

% figure;

% freqz(LPK,1,10000,fs); % run this to look at the frequency response of the filter

anapar.detrend\_data=1; % detrend the data (1) or not (0)?

anapar.HPF = 0; % high-pass filter the data? Usually we don't do this unless the drift in the data is bad and not linear (so detrending alone won't get rid of it)

anapar.LoCutOff = 0.05; % Cutoff frequency for the HIGHPASS filter (which is a 'low cut off' of the overall banpass we're effectively doing). Typically 0.05 Hz and very rarely above 0.1 Hz because it can result in distortions of the ERPs - see Steve Luck stuff online

% define the start and end times defining the epoch to extract - this should be wide enough to allow for any ERP or

% spectral analysis you might want to do. In this case we will analyse the spectrum in the half-second before the targets start

% and will do short-time fourier analysis which requires sliding windows of typically 400-500 ms duration.

% The end time should be late enough to go beyond the very longest RT you regard to be legitimate. In our

% behavioural analysis we saw that RT distributions extended out to 2200-2300 ms, so let's go longer than that

epoch\_limits\_ms = [-533 2500]; % note the use of integer number of cycles of SSVEP (18.75Hz for these data)

ts = round(epoch\_limits\_ms(1)/1000\*fs):round(epoch\_limits\_ms(2)/1000\*fs); % hence get the list of sample points relative to a given event that we need to extract from the continuous EEG

t = ts\*1000/fs; % hence get the timebase for the epoch (which we can plot average ERPs against) in milliseconds

%define the baseline window:

BLwin = [-2\*1000/18.75 0]; % exactly two cycles of the SSVEP flicker rate (18.75Hz).

% Now loop through the subjects and extract the single trial ERPs and the other important information about every trial:

for s=1:length(allsubj)

disp(['Extracting epochs for subject ' allsubj{s} '...'])

% make empty vectors and matrices for each thing we want to save on every single trial:

modir = []; % motion direction, 1=left, 2=right

coh = []; % coherence. 1=low, 2=high I assume

cond=[]; % block condition from list above

respLR=[]; % which button was actually pressed on the trial, 1=left, 2=right. So a trial n is correct when respLR(n)=modir(n)

RT=[]; % RT for each trial, in seconds

blocknum = [];% also keep track of block number so we can interpolate on per-block basis (cos bad electrodes usually fixed between blocks)

erp = []; % the EEG epochs. Whereas all of the above are 1-D row vectors, this will have dimensions 136 channels x epoch timepoints x trial - so the 3rd dimension should match the length of all preceding vectors

% now list the eeg files to be used for each subject:

filenames = []; filecond=[]; f=0; % initialize as empty a list of filenames and corresponding conditions

for c=1:length(blockConds)

for b=blocks{s,c}

f=f+1;

filenames{f} = [allsubj{s} '\_' blockConds{c} num2str(b) '.bdf'];

filecond(f) = c;

end

end

% Now go through these files, read them in and get their data:

for f=1:length(filenames)

% read data in:

% (we use eeg\_read\_bdf (by Gleb Tcheslavski, I did not change it) because EEGLAB's function has changed too often across versions!)

[EEG.data,numChan,labels,txt,EEG.srate,gain] = eeg\_read\_bdf([datafolder allsubj{s} '/' filenames{f}],'all','n');

% Extract the sequence of triggers and the sample points at which they happen:

% (The function eeg\_read\_bdf outputs the triggers as another EEG channel tagged on as the last row of EEG.data with pulses in it)

trgsig = EEG.data(end,:); % This has pulses wherever a trigger happens. This is usually channel 137 because there are 128+8(external) electrodes, but just in case there are more/less channels for some reason, we'll use 'end'

EEG.data(end,:)=[]; % then remove trigger channel from EEG.data

df = diff(trgsig); % take the first derivative of the trigger signal to detect the pulse transitions

trigs = df(find(df>0 & df<256)); % Find the pulses as the positive transitions and record their amplitudem which are the trigger codes)

% note trigger codes are always between 1 and 255 (usually a TTl pulse from parallel port)

stimes = find(df>0 & df<256)+1; % sample points in continuous data at which the triggers were sent

EEG.data = EEG.data\*gain; % a quirk of this bdf reading function is that you need to multiply by gain to get the microvolts on the scalp

EEG.data = single(EEG.data); % optional - if the double precision results in your files sizes being too big, you can convert to single precision, which is what EEGLAB does

% Shave off beginning and end bits based on triggers, because sometimes experimenter is delayed starting or

% stopping the recording, and the EEG might contain effects of the subject stretching out, yawning, dancing a jig

goodstuff\_start = max(stimes(find(ismember(trigs,relevant\_triggers),1)) - fs, 0); % go back 1 second before first trigger that matters

goodstuff\_end = min(stimes(find(ismember(trigs,relevant\_triggers),1,'last')) + 2\*fs, size(EEG.data,2)); % go 2 sec beyond the last relevant trigger

EEG.data(:,goodstuff\_end+1:end)=[];

EEG.data(:,1:goodstuff\_start)=[]; stimes = stimes-goodstuff\_start; % stimes is adjusted here also of course, so its sample points correspond to the now truncated continuous EEG

% Note that all the above lines of code in this loop are designed to work for ANY dataset, regardless of the triggers used (still it's worth thinking about whether shaving off up to 1 sec before 1st trigger and beyond 2 sec after last trigger is appropriate for your data)

% detrend?

if anapar.detrend\_data

% Detrending removes linear trends which often contaminate the continuous signal across a whole block,

% but it sometimes can make the signal WORSE, e.g. when there is a massive spike in the data near the beginning or end,

% then detrending will make an otherwise horizontal channel dramatically tilted.

% Therefore, sometimes as you do Bad Channel check you'll note blocks that should not have been detrended, and come back

% here and re-run for that subject with exceptions:

filesNotToDetrend = {'P11\_RD2.bdf','P11\_RD3.bdf','P11\_RD4.bdf','P15\_HC1.bdf','P15\_HC2.bdf','P15\_LD3.bdf','P21\_HC3.bdf','P21\_RD4.bdf'};

if any(strcmp(filenames{f},filesNotToDetrend)) % On first run of your data make this 'if 1' to detrend all blocks, and come back and rerun with exceptions if yo find blocks that shouldn't be detrended

disp(['Did not detrend block ' filenames{f}]);

else

EEG.data = detrend(EEG.data')';

end

end

if anapar.HPF % High pass filter?

[B,A] = butter(3,anapar.LoCutOff/(fs/2),'high'); % butter inputs are order, cutoff freq (in normalised scale where 1=half sample rate, and optional third argument telling it to be e.g. 'high'-pass)

EEG.data = filtfilt(B,A,double(EEG.data)')'; % filtfilt performs zero-phase digital filtering by filtering in both the forward and reverse directions

end

% LP Filter

if anapar.LPF, for q=1:nchan, EEG.data(q,:)=conv(EEG.data(q,:),LPK,'same'); end; end % it's an FIR filter so we can implement directly by convolution

% find indices of triggers corresponding to the events we want to time-lock to (targets):

targs = find(ismember(trigs,targtrig));

% Now loop through the single trials and grow the trial-descriptor vectors and 'erp' matrix

for n=1:length(targs)

if stimes(targs(n))+ts(end) > size(EEG.data,2), continue; end % first, check whether the epoch relative to the current event is even contained within the bounds of the EEG data (maybe the recording was stopped mid-trial?) - if not, skip the trial ('continue')

blocknum = [blocknum f];

coh = [coh trig2coh(trigs(targs(n))-100)]; % cute trick: subtract 100 and then use the 'trig2coh' above to translate into 1's and 2's

modir = [modir trig2md(trigs(targs(n))-100)]; % similar

cond = [cond filecond(f)];

nextrespind = find(ismember(trigs,resptrig) & stimes>stimes(targs(n))+anapar.minRT\*fs,1); % find the index of the next response in trigs/stimes

% Note that if a response on the trial was made faster than minRT defined above, the RT will be recorded as 'nan' below even

% though they did respond - that makes sense for this experiment because targets come up unpredictably and

% gradually emerge (so RT<200 ms is most probably a false alarm) but does it make sense for your task? Think about it

% (and/or ask Simon/a postdoc)

if ~isempty(nextrespind)

RT = [RT (stimes(nextrespind)-stimes(targs(n)))/fs]; % to get RT in sec, we take the difference in sample points between target and following response, and divide by the sample rate

respLR = [respLR find(resptrig==trigs(nextrespind))]; % resptrig above lists the buttons that can be pressed, usually left and right. to re-code the button pressed as 1=left, 2=right, fish out the index of resptrig corresponding to the current response

else % if there WAS no next response, set the response parameters for this trial as 'not a number'

RT = [RT nan];

respLR = [respLR nan];

end

% Now extract the epoch

ep = EEG.data(:,stimes(targs(n))+ts);

%Baseline correction:

BLamp = mean(ep(:,find(t>BLwin(1) & t<=BLwin(2))),2); % First compute the baseline amplitude by averaging across the baseline window

ep = ep - repmat(BLamp,[1,length(t)]); % Then subtract the baseline amplitude from the whole epoch (done for each individual channel, in one go)

% Now add the current epoch onto the growing 'erp' matrix by concatenation along the 3rd dimension:

erp = cat(3,erp,ep);

end

end

% Now that we have all trials from all blocks, save everything for this subject:

save([bigmatfolder allsubj{s} '\_raw'],'erp','t','modir','coh','cond','respLR','RT','blocknum','filenames','anapar');

end