4/3/2018
James Antony

Analysis / real-time info

Overview

- Analysis flow chart for all major scripts
- Figure plotting
- Pre-processing
- Physiology scripts
- Aligning events
- Real-time code setup

Useful info

Before any of the scripts will work, go into analysis->load_root.m and change…
- Folder directory
- EEGLAB directory (you must have this installed somewhere to run some operations)
- directory for R script outputs

Subject numbers
Experiment 1: 02-22
Experiment 2: 101-125
Experiment 3: 181-205. Note that I use version=5 to run these analyses in the paper.
Control (no sounds experiment): 1-12 (unpublished data) and 302-321 (Antony et al., 2012, Nat Neuro, no sounds control).
Note also there was a version=4 that is not part of the paper. It was an earlier real-time experiment that proved helpful in designing the one that would eventually be Exp 3 (version=5). This dataset is omitted for simplicity, but can be shared upon request.

Analysis flow chart

- Scripts in boxes require more explanation - see scripts themselves

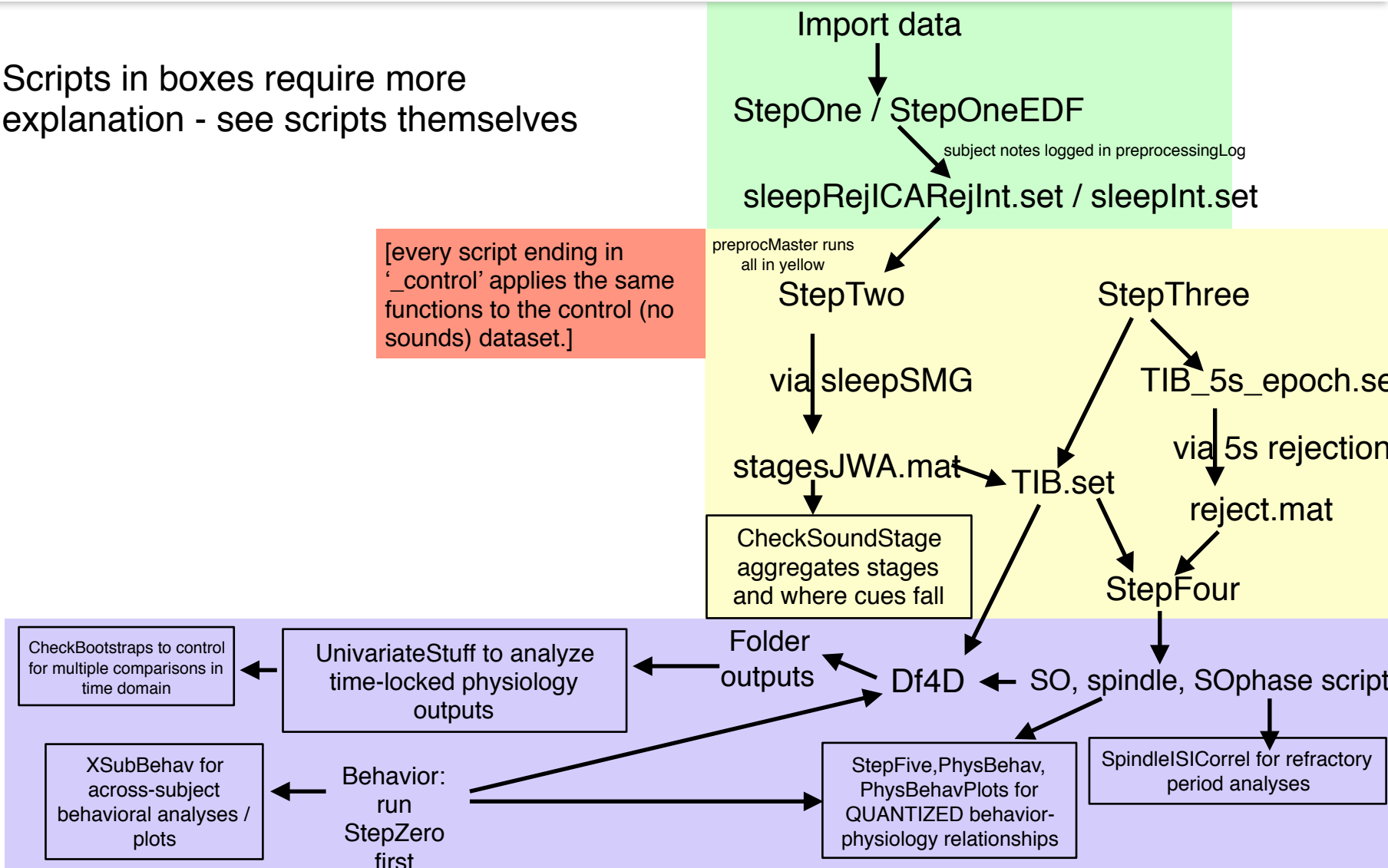


Figure plotting

Summary of the .m files where you can find code for each plot:
Fig 1B: XSubBehav, set version=1
Fig 1C: RMSexplanation
Fig 1D: StepFive, set version=1
Fig 1E: PhysBehav, PhysBehavPlot, set version=1
Fig 1F: XSubBehav, set version=2
Fig 1G: PhysBehav, PhysBehavPlot, set version=2
Fig 2A: StepFive, see bottom of script where we create version=3 to combine version 1/2
Fig 2B: UnivariateStuff, set version=3, memversion=3 (median split), and bslc (baseline correction) to 0
Fig 3A-D,F: SpindleISICorrel, set version=3
Fig 4B: StepFive, set version=5
Fig 4C: XSubBehav, set version=5
Fig 4D: StepFive, set version=5
Fig 4E: StepFive, set version=5
Fig 4F: UnivariateStuff, set version=5, memversion=4 (early vs late), and bslc (baseline correction) to 0
Fig 4G: SpindleISICorrel, see bottom of script for comparison between version=3 and version=5
Figure S1A-B: CompLinRegClassy2, uncomment lines at bottom
Figure S2A-C: SpindleISICorrel_control, set version=101. Note this will combine both control (no sound) datasets mentioned in the Methods section.
Figure S3: RMSexplanation (it will load a sample subject and time interval)
Table S1: CheckSoundStage, set version=10 to just grab data from all three experiments
Table S2: SpindleISICorrel, set version=3, see reverberation analysis and ‘PeakTimes’ variable

Importing data

- Use StepOne / StepOneEDF
  - Load files, downsample, select relevant channels
  - Note: two differences for Biosemi (from Neuroscan)
    - Electrode location files
    - Usually larger – need to load in steps and combine
    - See ‘StepOneBiosemi’
  - Best practice: save notes about each subject at the bottom of StepOne or somewhere
- StepOneEDF - does the same thing, but loads EDF files, which are what we get after converting from the real-time software, OpenViBE

EEGLAB structure

- Event structure - note each of the variables
- 'EEG.history' → scripting
  - Not perfect, and doesn't log everything, but logs many things to run in batch scripts
  - Best practices: script everything you can and only go to the GUI when necessary. This saves you time and also ensures any errors are (at least) not due to negligence on individual subjects.

Step 1.5 - Interpolation

- I do two things:
  - Graph spectrogram for every channel
    - figure;pop_spectopo(EEG,1, [0 EEG.pnts], 'EEG','freqrange',[2 64],'electrodes','off');
  - Skip through the recording every 500s to see if any channels go in an out for a decent length of time
- If you run ICA, you must do this first WITHOUT the channels you need to interpolate.
  - For instance, if you need to interpolate channel 57, use ‘channelstoinclude=[1:56 58:68]; EEG = pop_runica(EEG,’icatype’,’runica’,'dataset',1,'chanind',channelstoinclude,'extended',1);
- Then reject eye blink components
- Then use pop_interp (example: electrodes 8 & 10)
  - EEG = pop_interp(EEG, [8 10], 'spherical');

Master controller script for remaining steps

- FYI - I use the script, preprocMaster.m, to do interpolation and generally just run the other ‘Step’ scripts between the other tasks. It controls everything in the yellow box on the flow chart.

Filtering and separating files

- Use StepTwo
  - For CNTs:
    - EEG & EOG files filtered 0.4-50 Hz
    - EMG filtered 10-59 Hz (Delphine low-passed at 70)
  - Then data are re-referenced to L-MSTD and separated into 8 channels for the montage in ‘sleepSMG’ (‘SM.set’)

Sleep staging

- Use sleepSMG
  - Go to ‘SM.set’ directory
  - Stores stages in .mat file, variable stageData.stages
  - Additional things you can do:
    - Back-project spindle / SO analyses to test how well the automatic scripts are doing

Epoching & artifact rejection

- Here we reject artifacts in 5s segments
  - My philosophy: reject anything that isn't brain activity
- Use your judgment
  - If a single channel shoots up by 10,000 microvolts, probably should reject it
  - If it does this repeatedly, should probably go back and interpolate this channel and run through steps 2 & 3 again
  - There is constantly this tug and pull of the way to have optimal data. At some point you could interpolate small segments and do this forever and it would take forever, but you'd technically have better data. I see it best as being relatively conservative and time-saving with things, as I think the signal lost in not obsessing over data is relatively small.

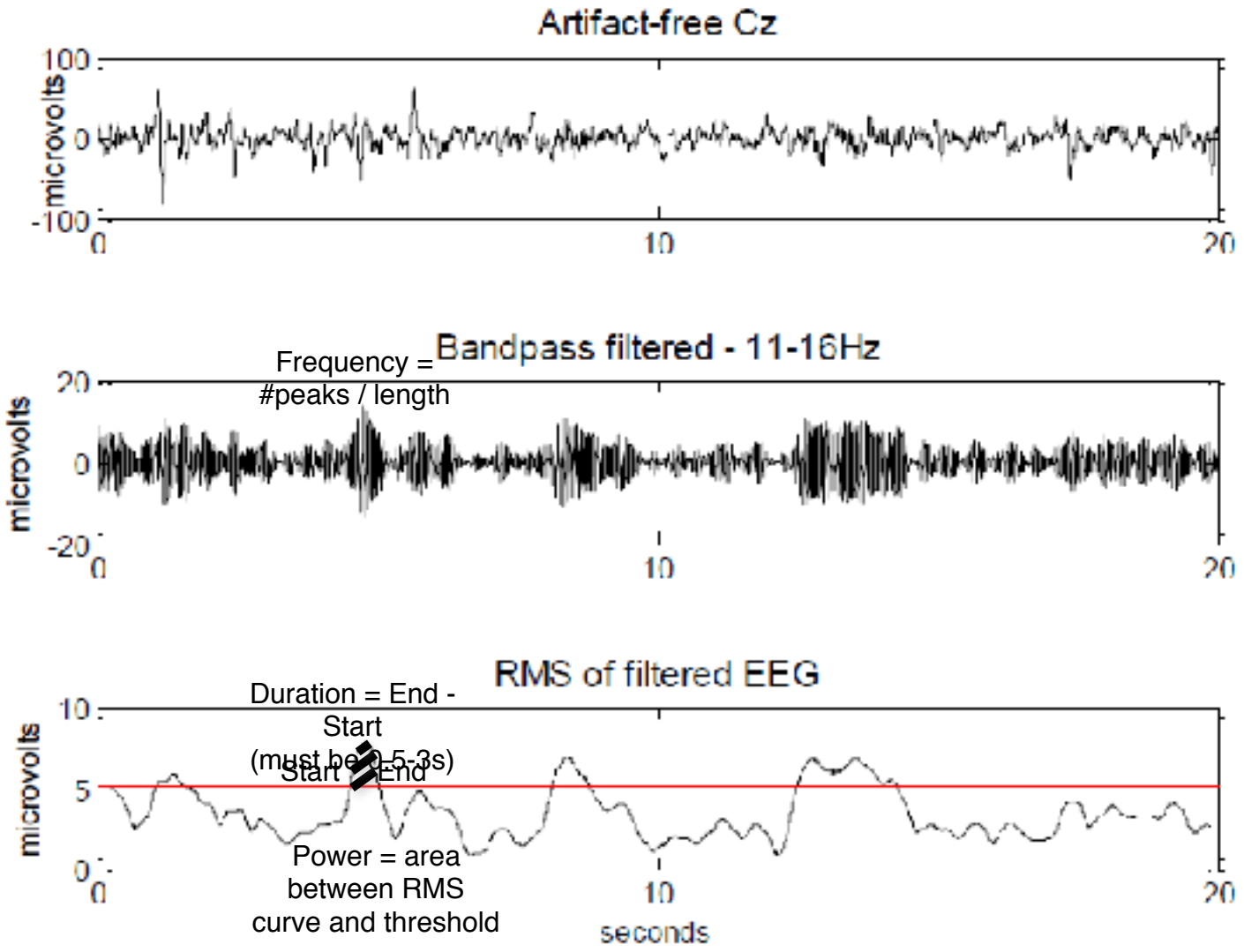
Epoching & artifact rejection

- StepThree splits up data into 5s “epochs”
  - Why 5s chunks? It goes quickly, though if you are overly concerned about losing extra data you can split it up how you want in StepThree and beyond steps
- Open ‘TIB_Filt_Epoch_5s.set’ for the subject
  - EEGLAB → Tools → Reject Data Epochs → Reject By Inspection
  - Keep default checks (do NOT select the Reject button), mark epochs you want to reject and DO NOT FORGET TO CLICK ‘UPDATE MARKS’ WHEN YOU’RE DONE
  - Type reject = EEG.reject.rejmanual; save reject reject;
  - Basically we are MARKING epochs for rejection and saving a .mat file of a 0 or 1
    - Why do we do this? In case you have to go back and change something in pre-processing, we have a file externally saved so we won’t have to do it again with new .set file!

Physiology scripts

- These files will read stages and artifact rejected epochs for each subject
- Can run on all stages separately, I run on just NREM altogether
- Saves each in individual folders
- Spindle / SO: saves each individual spindle characteristics (frq, dur, pow, timing) as well as spitting out density / total #s
- Spectral: formatted to give averaged power spectral density for each stage, though you can tweak it to give you more fine-grained
- SOphase - gives SO phase and power for a full continuous recording

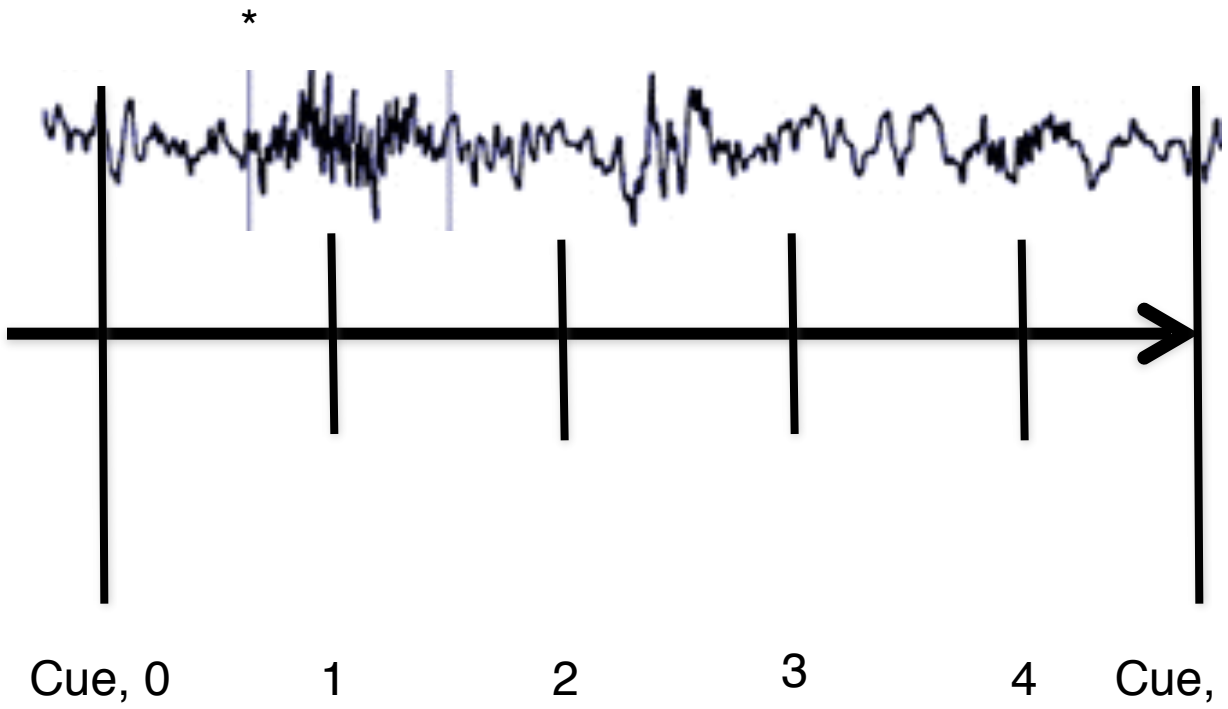
Spindle algorithm



Spindle algorithm

- SO script is similar
- Spectral a bit different
- Here you should try to walk through the script in bits and see how the algorithm works

Aligning physiology w/ cues



* Counted as spindle started between 0-1s (but you could cut it however you want).

Basic alignment approach

- You'll find this in StepFive.m
- Loop through all relevant cues
  - Restrict to sleep cues, possibly only in one condition
- Loop through all spindles in an electrode (or cluster of electrodes one by one)
- If spindles occur within a certain time of cue, mark the characteristics for later

Possible applications

- Do cues boost SOs / spindles in temporal proximity to cues?
- Does SO / spindle incidence differ based on condition / future memory?
- Inserting physiological starting points as “events” and time-locking to them (as in Ngo et al., 2013)
- Anything else you can think of related to these events

Analysis issues

- Using both ICA and sleep
  - Huge issue: need to reject continuous data to run ICA, but need to score sleep data in 30s epochs
  - Lots of time remapping needed to re-align the events
  - Do this only if you need to
  - Another option is to look for ICA components during sleep. This is something I've explored, but never published. For instance, there are nice spindles / SO components that come out.

Real-time setup

I used OpenViBE for this experiment. If you're interested in using this setup, please go here (http://openvibe.inria.fr/downloads/) and read further. All of the files related to the real-time setup are in **realtime** folder in the main directory. Here I will just give a high-level overview. As always, please e-mail me if you're using this to actually set up an experiment and would like some assistance or can improve this (james.ward.antony@gmail.com).

Things you'll need to do to directly replicate our setup:
- install OpenViBE. I recommend doing a few tutorials to at least get the basics of what it can do.
- have a 32-bit MATLAB version installed. This *does* mean an older version of MATLAB. We used 2011a.
- change settings in OpenVIBE acquisition server like I have in Settings1 and Settings2. Check channel names to make sure they're correct and change if not.
- within 'spindle_filter_Initialize.m', change subject name. I have included one sample subject '177' for reference if you want to read in offline. This function gets called by the OpenViBE designer and you will need it to point to the correct directory of your files.
- within the same file, change the ipA address to the acquisition (Biosemi) computer and ipB to the presentation computer (where you will send the udp packets)
- on OpenVIBE acquisition server, click Connect and then Play
- open OpenViBE designer and open the 'sample-sleep-readin' (offline) file. It should look like the screen shot taken in 'DesignerOffline'. Click on some (or all) of the boxes and see what settings we used to get an idea of the way you can use the program. Also, click on the boxes where we read in MATLAB files and change to MATLAB directories and the correct directories for the .m files.
- If this runs, there are things you can do to see the algorithm work. When we detect stage-2 or SWS, we press 'a' to turn on the spindle detection algorithm. Then when we're ready to deliver cues, we press 'z'. For both of these, you need to have the OpenViBE box selected that has information about the cues.
- If you want to try real-time open the 'sample-sleep' file. Click on the boxes where we read in MATLAB files and change to MATLAB directories and the correct directories for the .m files.

Real-time practicalities

More practical things about running real-time setup for real. Ignore if not interested in specifics. **This is the actual order of things we do when running!**

Before subject is done with learning:

On Biosemi computer

1. Open MATLAB 2011 (must be 2011 b/c 32-bit!)
    1. Open documents -> Spindle -> signal_filter_Initialize.m, enter subject ID
    2. Change 'iter' to '1' initially (and '2' later if you have to restart file and point to the correct restart file).

On both computers

1. Open up google drive in rtSpindle folder

When finished with Phase 3

1. Set up for sleep

On stimulus computer

1. un DRDF_Motor>CreateSleepCues_2 and enter the correct 'subid' to create sleep cues .txt file.
2. Send cues to self on Google Drive!
3. Change overall computer sound level to '10' and play jantony -> spindle -> brown_noise2
4. Run 'testclient.m' line 'setuptc_DRDF.m'. (within jantony → spindle). This will open up the port for accepting UDP packets. Don't run 'closetc' until after the nap.

On Biosemi computer

2. Put cues in Documents -> Spindle -> Data (leave outside subject folder)
3. Close Biosemi. This must be done or OpenViBE won't run.
4. Open OpenViBe -> openvibe-acquisition-server, click Connect then Play
5. Open openvibe-designer and switch the tab to sample-sleep_DRDF.xml
6. Click play on the openvibe-designer and everything starts
7. Enlarge signal windows and bring up trigger window
    1. Make sure 'system load' is visible on main OpenViBE designer
    2. When sub enters / leaves NREM, press 'a'
    3. When sub enters / leaves SWS, press 'z'
    4. ***MUST CLICK ON TRIGGER WINDOW FOR IT TO REGISTER TRIGGER!
    5. NOTE - if the spindle threshold (default=4.5) appears too high or too low (is collecting way too many or too few spindles), press 'e' to increase it (+0.5) or 'r' to decrease it (-0.5), but otherwise leave it.
    6. NOTE - time of recording is at top of screen next to play button!
8. To FINISH - press STOP on openvibe

Real-time troubleshooting

More practical things about running real-time setup for real. **This is the actual order of things we do when running!**

Troubleshooting

If the input isn't coming in …

1. I recommend just closing openvibe designer and acquisition server all the way out, then loading it back up.

If you need to stop during the experiment ...

1. The experiment will print a file and 3 others one folder back from the data file. The folders will have the correct date in the format 'month_dat_hour_minutes_seconds'. You should move them to 'spindle → Data → [subid folder]' when finished.
2. Move the data file (which is a .ov file created with the current year/month/date in the spindle → Data folder) into the subject folder ('spindle → Data → [subid folder]')
3. To update which sounds have been played!!!
    1. Go to signal_filter_Initialize.m within the 'spindle' directory
    2. Change the 'iter' variable from '1' to '2'
    3. Change 'prevsoundfile' to the sounds file you just moved into the data folder in step 1!
    4. Change 'randSoundfile' to the randSounds.txt you just moved into the data folder in step 1!

Testing the scripts offline

1. Use sample-sleep-readin.xml instead of sample-sleep.xml
    1. If you can't do this, then do the following:
        1. Use sample-sleep.xml but change the name to something else
        2. Delete connection from Acquisition client to decimation and delete connections from decimation factor to channel selectors.
        3. Switch to generic file reader, change name to intended directory
        4. Right click on output and change it from EBML stream to signal
        5. Connect to both channel selectors directly
        6. Change signal back to EBML stream (it's weird and buggy but you gotta do it this way)
        7. Press play!
2. Load 177's .ov file in Generic Stream Reader and change subid in signal_filter_initialize.m.
3. Turn on stimulation using 'a' and 'z' and observe

Converting to EDF (from Margaret):

1. Within the OpenVibe Designer, find the .xml file that is named converter-CSV or converter. change the input file to the .ov file you wanna convert, and the output file to the name of the .csv file you want to output in the correct directory ('[subid]_S.csv'). be sure to add the extension .csv to it. it will take usually 1.5x as long as the actual file itself (i.e., for a 90 min nap, it will take about 135 min, or 2 hr 15 min, to convert. so you can just leave it up running). NOTE: it will not tell you it's finished … but if you wait long enough it will be done so you can just stop it.
2. After the .ov file is converted to .csv, you can then use EDFbrowser to convert the .csv file to .edf (it's under Tools --> ASCII to EDF converter). When the popup box comes up, you should click **Load** and then load 'ascii_to_edf.template' from the main 'spindle' directory.
3. once you load it in, then you can convert the .csv file. fill in the random blanks with subject number or whatever; doesn't really matter. it'll take a while (~5-10 min), and then a popup will come up asking you to put the .EDF file in the appropriate directory. after you do that, it'll take another ~2-3 min and then it should be finished.
4. Drop EDF files to dropbox, as well as sounds.txt, spindles.txt, and triggers.txt files from 'spindle' directory.

Questions?

- Email james.ward.antony@gmail.com