***Sparta Lab - UP-TO-DATE WORKFLOW: CeA CRF EPHYS DATA [10-05-18]**

Updated:	4/24/2019 6:37 PM
Author:	James M. Irving

- **Please note there is an addendum at the bottom of this note containing a final step (creating intervals for the DID session)**
  that must be done before the scripts below can be run. [04/23/19]

*Sparta Lab - UP-TO-DATE WORKFLOW: CeA CRF EPHYS DATA [10-05-18]

Author: James M. Irving

UP TO DATE WORKFLOW: [edit 10/05/18]

This workflow starts after all of the cleaning the .pl2 files in OfflineSorter, opening them in NeuroExplorer (Nex), and after identifying the good intervals for each DID session (DIDSessInts variable), and then saved as .nex5 file.

Note: you will want/need a log of which files were recorded on what drinking day, this will be added to the DATA.fileinfo structure, and will then be included in Excel printouts at the end.

LEGEND
[!] = An choice/script/workflow that may break the chain required for remaining scripts OR critical point in which you may want to edit the code.
[i]= Important points and details.
bold, highlighted = name of a custom script or program that is used
normal, highlighted = text that must manually be typed into the Matlab command window.

START BY PUTTING ALL RAW .NEX FILES IN ONE FOLDER

- **Must have "DIDSessInts" interval variable**
  Very beginning is putting all .nex5 files into one folder

- **#1) In Matlab, run NexCombinedDATAstruct.m [i]**
  - communicates with Nex to extact all variables and info
  - Requires:
    - **_MakeDIDEventNamesFixed.nsc (Nex)**
    - nexDATA_splitSpikeRate (ml)
    - nexOptionsCriteria (ml)
    - **calcWFcorrelationMLTrigd (ml)**

- **#2) Classify light responses and lick responses:**
  - **Run nexDATAclassificationFixWIP2 [temp name]**
    - Requires:
      - **nexTestUnitResponsesFixWIP_Fin**
        - Calculates the Wilcoxon results comparing timebins.
        - Which requires:
          - **JMI_spkMatFun.m**
            - Creates PEH matrix to analyze # spikes per bin
      - **renameDATAfileinfoDrinkTypeDay**
        - Renames default fieldnames in DATA.fileinfo

- **#2B) Fill in drink type and drink day for each file's recording session (DATA.fileinfo)**
    - **Run nexDATA_fillDrinkDay** and type in each answer when prompted
        - **OR** manually fill in DATA.fileinfo.drinkDay and DATA.fileinfo.drinkType for every file.
    - **[!]** NOTE: Optional (sort of)
        - Excluding this step will not become an issue until **step #8,corrSPIKESandLICKSv2**
        - Can run this script right before step #8 as well.

- **#3A) Analyze binned firing rates to find & replace outliers.** saves index of outliers to use with future binned data
    - **Run findOutliers_DATA_RawSpikes_SaveBins_CutLicks [i]**
      (or just nexDATAcountUnitsFinal if don't want to remove outliers) **[!]**
    - Uses the raw timestamps to calculate average firing rates across session in 5 minute bins
        - **Finds outlier** timebins by using a sliding window of 30 mins
        - **Criteria**: bin is more than 3 scaled deviations from the median away from median of data in sliding window.
        - **Replaced** with linear interpolation to fill removed outliers
    - Requires:
        - DATA structure
        - **nexDATAcountUnitsFinal script**
            - Creates COUNTS structure: # of each type, and [Q,u] index for light, lick, and light-lick responses.

    - **[!] [i] NOTE: The data saved in DATA(Q).units(u).spikeRate.ratesClean is critical, is used for all norm firing analysis [i] [!]**
        - It is used later on by nexPrepAll_forNormFiringFigsSpliByDeltaRate.m

        - **TO CHANGE which version of the firing rate data is used for normalized firing rates:**
            - Change what matrix is stored as "cleanedData" in OUTLIERSfull.(currLightType).(currLickType).cleanedData;
            - in the middle of nexPrepAll_forNormFiringFigsSpliByDeltaRate

- **[!] #3B) SAVE DATA .MAT FILE NOW.** THIS DATA FILE WILL BE COMBINED WITH THE BURST DATA FILE AFTER BURST ANALYSIS] **[!]**
    - In Matlab: save('DATA-OutliersRemoved_NoBURSTS.mat' , '-v7.3')

#4) BURST ANALYSIS - USING NEUROEXPLORER & MATLAB

- **#4) PART 1 - EXPORTING UNITS TO NEUROEXPLORER FOR BURST ANALYSIS AND IMPORTING RESULTS BACK TO MATLAB**

    - In Matlab, run **makeUnitTimestampVarsForAll**.m
        - It will save matlab file('FUnitsToNexForBursts.mat')
        - Close Matlab, open NeuroExplorer

    - **In NeuroExplorer (Nex), you will first connect to Matlab**
        - Method One:
            - **Select Menu > Matlab > Get Data from Matlab > Open matlab in as engine;**
                - (Will open new clean matlab window)
                - IF NEX DOES NOT OPEN A FULL MATLAB WINDOW, YOU MUST USE METHOD TWO
        - Method Two**:**
            - Open NeuroExplorer. select Menu>New File.

        - In variable list, select either variable (StartStop or AllFile), then Menu > Matlab > Send selected variables to Matlab
        - Now, a Matlab window will open and contain the variable that you selected in Nex.
        - Delete the variable in Matlab
            - either type "clear" in Matlab
            - right click variable in window and select Delete

    - **In the NEW Matlab window :**
        - Navigate to folder with your saved file
        - load('FUnitsToNexForBursts.mat')

    - **In Nex:**
        - select **Menu > Matlab > Get Data from Matlab >Timestamp Variables > List of Neurons**
        - Shift-click to select all of the FUnit variables in the list

    - **Run the BurstIntervals for All Units in Burst NeuroExplorer script: (2 ways to do so)**

        - **Method 1) In Nex, open the Scripts window, right click on "BurstIntervals for All Units in Burst.nsc"**
            - select either "Run Script..." options
        - **Method 2) In the Matlab command window, paste these 2 lines of code:**

```
    nex = actxserver('NeuroExplorer.Application');
    res = nex.RunNexScript('BurstIntervals for All Units in Burst.nsc')
```

        - **NeuroExplorer will run the burst analysis and then send the resulting variables back to Matlab**
            - Should now have more variables in Matlab, with new variables ending in BurstSpikes,etc.

        - **In Matlab:** save('FUnits-postNexBurstAnalysis.mat','-v7.3')
            - Can either run burst analysis part 2 now, but may also load this file before running the first script below.

- **#4) PART 2 - ANALYZING BURST RESULTS FOR FURTHER ANALYSIS IN MATLAB**

    - In Matlab, open FUnits-postNexBurstAnalysis.mat from part 1
    - **Run BURSTunits_Analysis.m**
        - Creates & saves
            - BURSTunits structure
            - Saved as "BURSTunits_DataStructure.mat"

- **#4) PART 3 - IS PERFORMED BELOW AFTER COMBINING WITH PRIOR DATA:**
    - see steps #5-6

[i] BURST NOTE: Now that you have BURSTunits_DataStructure.mat you DO NOT NEED TO REPEAT the burst analysis. Just jump to step #5 AND #6 after updating your data.

- **#5) - COMBINE BURST DATA WITH SPIKE DATA & SAVE**
  - **Load** primary data file: DATA-OutliersRemoved_NoBURSTS.mat
  - then **load** BURSTunits_DataStructure.mat
  - **Save the combined data as a (large) .mat file**
    - save('DATA+BURSTS-OutliersRemoved.mat' , '-v7.3')

- **#6) CALCULATE %SpikesInBursts BY BINS & REMOVE PRIOR OUTLIER BINS**
  - Run **nexBurstByHourUsingCalcPercBursts**
    - **Requires:**
      - Variables**:**
        - DATA, BURSTunits,
      - Custom Matlab Functions:
        - **cutBinTimesGoodInts** (timestamps,binsize,goodInts,timebins)
          - Will cut out bad time bins based on intervals and timebins input arguments
        - **replaceBinnedOutliers** (binnedDataToClean, outliersStruct, fillMethod1, fillMethod2)
          - Will use OUTLIERSfull structure to replace the previously identified outlier bins for other datasets (like % spikes in bursts).
    -

- **#7) CALCULATE NORMALIZED FIRING RATES, COLORPLOT, AND GROUP AVERAGE DATA FOR GRAPHPAD**

  - **#7A) To calculate normalized firing data for all units: (required)**
    - **nexPrepAll_forNormFiringFigsSplitByDeltaRate [i]**
      - Performs all matrix-creating prep, analysis, and plotting for the normalized firing rate data, and %SiB data. **Needed to produce output data for to paste into Graphpad templates for bar and line graph files**. (done at end using "copyPaste..." scripts)
      - Creates:
        - COUNTStoPlot data structures
        - RATEScut data structures
        - SORTtracker data structures
          - SORTtracker/SORTtracker_SplitByRateChange = .light.lick... preSorted and postSorted data for all units
          - SORTbyLight/SORTbyLightOUT - same as above but only CRF vs NonCRF
      - Needed for:
        - Copy and pasting ground average data into Prism tables using the "copyPaste.." scripts save as Excel files
      - CRITICAL:
        - **[!] [i]** The firing rate data used is: DATA(Q).units(u).spikeRate.ratesClean,
        - **To change which data is used, see NOTE in Step #3) findOutliers_DATA_RawSpikes_SaveBins_CutLicks**

  - **#7B) To make normalized firing colorplot for JUST CRF units :**
    - **First, must manually enter this code:**
      - SORTtracker=SORTtracker.CRF;
      - SORTtrackerCRF=SORTtracker_SplitByRateChange.CRF

    - **plotNormFiringFromSORTtracker_V2WIP or plotNormFiringFromSORTtracker_V3WIP**
      - For JUST CRF colorplot

- **#8) Calculate spike and lick rate / cumulative lick correlation analysis**
  - Run **corrSPIKESandLICKSv2_updateMatchV3**.m
    - **Will analyze correlations of time-binned data for:**
      - **Lick Rates vs %SiB**
      - **Lick Rates vs Firing Rate (HZ???**
      - **Cumulative Licks vs % SiB**
    - Requires:
      - LICKS, SPIKES and BURSTunits (post-step #7) nexBurstByHourUsingCalcPercBursts)
    - Creates:
      - CORRlicks structure
  - **[!] NOTE: Make sure you have performed step #2B BEFORE running this script. (running nexDATA_fillDrinkDay)**

- **#8B) [!] THIS IS A PERFECT SPOT TO SAVE YOUR MASTER, VIRTUALLY COMPLETE DATA FILE**

- **#9) EXPORT EXCEL FILES REQUIRED FOR PRISM AND EXCEL ANALYSIS**
  - **#9A) Details for each unit for filtering:**
    - **Run printUnitNamesDetailsOutRem.m**
    - Saved master unit data spreadsheet with every unit's properties that we use for most graphs.
    - NOTE: right now this script assumes have
      - DATA(Q).fileinfo.drinkType
      - DATA(Q).fileinfo.drinkDay
    - [!] NOTE: Make sure you have performed step #2B BEFORE running this script. (**nexDATA_fillDrinkDay**)
    -
  - **#9B)Prism-ready tables for group averages and normalized firing rates**
    - **copyPasteSORTdata**
    - **copyPasteSORTdata_EarlyVsLate -**
    - **copyPasteSORTdata_EarlyVsLateSPLITS**
      - **Creates excels with data tables pre-configured to be pasted into Graphpad Prism temples.**

CRF Paper Ephys Workflow - Addendum - Save final data file (.pl2) [04/24/19]

Author: James M. Irving, Ph.D.

Fast Method - If Just need DIDSessionInts and do not have bad time periods to remove from analysis: [04/23/19]

- **Open Final .pl2 file in NeuroExplorer.**
- **Identify the time range of phototagging (EVT25)**
  - Select TimeStamps Tab at the bottom of the Variable Window.

    - Scroll horizontally to the Light Pulse event (name = EVT25 for OmniPlex Recordings)
        - Make note of the both the FIRST and LAST timestamps for EVT25

- **<u>Create an Interval Variable Called "DIDSessionInts" to exclude the phototagging time period</u>**
    - From Menu bar > Edit > Add Interval Variable. (the Create New Interval Variable window will open.)
    - **At top of Create New Interval Window:**
        - **enter the name as *exactly* "DIDSessionInts"** (must match exactly)
        - Take note of the Last Timestamp in File (directly below name)
    - **Fill in Row #1's Interval Start and Interval End based on when phototagging was done.**
        - If phototagging was in the beginning:
            - Interval Start = EVT25's LAST timestamp plus *at least* 1 sec. [if 1299.7554 enter 1301 secs]
            - Interval End = The Last Timestamp in File (rounded down) [if 15777.8584 enter 15777 secs]
        - If phototagging was at the end:
            - Interval Start = 0 seconds (beginning of recording)
            - Interval End = EVT25's FIRST timestamp minus 1 sec [if 14888.1245 enter 14997]
    - To Verify Interval Looks Correct (optional):
        - Select Intervals Tab on bottom.
        - Each interval has 2 columns.
        - Should see an "AllFile" start and end columns
        - Should see new DIDSessionInts columns.
- **Now, save file as .nex5 file. [** File > Save As Nex5 File.]
    - **Original folder is fine for now, but must move to same folder as all other .nex5 recordings for analysis)**

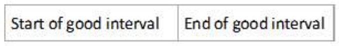
    - **Original Method: Saving session intervals for accurate session-wide firing rate data [creating "DIDSessionInts.csv"**

i. In same folder as final cleaned data file, save a new Excel workbook as a .csv file called "DIDSessionInts.csv"
    1. This .csv file will determine the good, non-phototagging, non-erroneous intervals that Nex uses for using data for firing rate histograms

ii. Use timeline view with EventPulse (Event25/MAP Data event?) turned on to find timestamps.
    1. Bottom of offlinesorter has a status bar that displays the voltage and time at the location of the mouse cursor.
    2. Identify timestamp of last light pulse, then use mouse to find timestamp shortly thereafter
        1. If theres a gap immediately following, use timestamp of first wave after gap.
        2. If no gap and no obvious change in spikes on timeline, then use ~100ms+ as timestamp

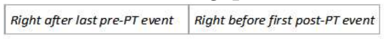
i. In DIDSessionInts.csv, each interval to include is a separate row [Need at least one interval]

1. Each interval row needs 2 columns to define intervals that are included in the DID session



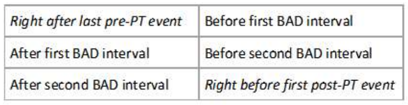
Start of good interval	End of good interval

2. If there's NO wide gaps or bad time periods to remove:



Right after last pre-PT event	*Right before first post-PT event*

3. If there's large gaps or bad data periods to exclude:



Right after last pre-PT event	Before first BAD interval
After first BAD interval	Before second BAD interval
After second BAD interval	*Right before first post-PT event*