**Electrophysiology Analysis Optimization using Intan Technologies**

Documentation and Protocol presented

by

Baihan Lin

to

Olavarria Lab

Department of Psychology

in partial fulfillment of the data analysis task

in the subject of

Computational Neuroscience

University of Washington

Seattle, Washington, USA

Apr 2016

**Mentors:**

Prof. Jaime Olavarria

Dr. Adrian Andelin

**Author:**

Baihan Lin

**Abstract:**

The thesis is given birth due to the need to analyze a set of electrophysiology recording data and the curiosity of finding the best way to generate and present the result, and to simplify the entire electrophysiology analysis in a simple automated protocol. Comparing to the traditional method, which spends 17640 minutes (~300 hours) to analyze the 441 cases of data files in our current stage, my method only takes less than 10 minutes, which is 1764 times faster and saves 300 hours. I hope this method described in this documentation can offer a more systematic and convenient way in the anlaysis of electrophysiology recording data. I look forward to further improvements if any. After tens of hours of my working each week to develop this automated protocol, I hope lab members in the future can simply run the program in 10 minutes to save their time and lives.

**Comment:**

This documentation is mainly for the purpose of recording my thoughts and attempts in optimizing the data analysis task, therefore not in a standard of submission in any kind.

**Keyword:**

MATLAB, protocol, electrophysiology experiments, Mac terminal, Intan Technologies

**Contents:**

**[Abstract …………………………………………………………………………………. 2](#Abstract)**

**[Keywords ……………………………………………………………………………….. 2](#Abstract)**

1. [**Traditional Solution:** (Challenges Faced)……………. ………………………….…**5**](#TraditionalSolutions)
   1. Darking/Contrasting Process
   2. Utilization of Gaussian Blurring via ImageJ or Photoshop
   3. Arbitrary Drawing of line in the V1

1. [**Task 1: Indicating Patch Index** (Java, ImageJ, MATLAB, Patch Analyst) ……….. ...**6**](#IndicatingPI)
   1. Solution 1: Implementing the traditional (via Java or MATLAB)
   2. Solution 2: Patch analysis via Patch Analyst (New possible alternative)
   3. Solution 3: Particle analysis via ImageJ (New possible alternative)
   4. Solution 4: width of line (Improvement on traditional)
   5. Solution 5: Cluster Index (New possible alternative)
   6. Solution 6: Machine Learning (New possible alternative)
2. [**Task 2: Optimizing the data:** (ImageJ, Excel, Java, MATLAB)…………………...... **9**](#Optimizingdata)
   1. Standardization
   2. Smoothing Method (Moving Average):
      1. Excel (Exploration)
      2. Java (Implementation)
      3. MATLAB (Outlook)
3. [**Task 3: Systematization:** (systematize the protocol more efficiently) …………….**13**](#Systematization)
   1. Automatic file reading and processing
   2. Visual user-friendly version
   3. Automatic software operation
   4. Patch Processor (a Software designed by Baihan Lin)
4. [**Method Validation** ………………………………………………………………... **22**](#MethodValidation)
5. [**Future Improvements** ……………………………………………………………. **23**](#FutureImprovements)
6. [**Data Summary** …........................................…………………………………….... **24**](#DataSummary)
   1. Data Charts
   2. Excel Smoothing Plot (as an indication of the validity of SF = 10/364)
   3. ImageJ Plots & Lines (line & line with width of 100)
7. [**Code Summary** (20 Java programs)…….………..………………...…………….. **33**](#ImportantCodes)
   1. **First Attempt** (using TreeMap)
      1. [MapPatchIndexMain.java](#a1) (main method)
      2. [MapPatchIndexSolver.java](#a2) (subprogram)
   2. **Better Solution** (using ArrayList)
      1. [PatchIndexMain.java](#b1) (main method)
      2. [PatchIndexSolver.java](#b2) (subprogram)
   3. **Smooth Factor Finder** (test different smooth factor to find the most proper)
      1. [SmoothPatchIndexMain.java](#c1) (main method)
      2. [PatchIndexSolver.java (subprogram)](#c2)
   4. **Automatic System** (read-all files in a folder)
      1. [AutoPatchIndexMain.java (main method)](#d1)
      2. [PatchIndexSolver.java](#d2) (subprogram)
   5. **Attempt to visualize** (output the system console, failed)
      1. [AutoPatchIndexMainBackUp.java](#e1) (main method)
      2. [PatchIndexSolver.java](#e2) (subprogram)
      3. [ConsoleTextArea.java](#e3) (subprogram)
      4. [LoopedStreams.java](#e4) (subprogram)
   6. **Second Attempt to visualize** (user-friendly, using acm.jar and JComponent)
      1. [VisualPatchIndexMain.java](#f1) (main method)
      2. [VisualPatchIndexSolver.java](#f2) (subprogram)
   7. **Standard Java File** (standardize, exportable to jar, then to exe)
      1. [VisualPatchProcessorMain.java](#g1) (main method) with jar/exe
      2. [VisualPatchIndexSolver.java](#g2) (subprogram)
   8. **Version 2.0** (with excel export, smooth factor and imageJ functions, browse)
      1. [VEPatchProcessorMain.java](#h1) (main method) with jar/exe
      2. [ExcelFileDealer.java](#h2) (subclass)
      3. [PISolver.java](#h3) (subclass)
      4. [ImageJ.java](#h4) (revised subclass)

[**Bibliography ………………………………………………………………………..... 130**](#Biblography)

[**Special Acknowledgement …...…………………………………………………….... 131**](#SpecialAcknowledgemetn)

**Work Progress:**

**(What, when and how I approached the project)**

I used MATLAB and GitHub to perform my project, because it seems previously I programmed a lot but never kept record for it. Working at night and other irregular hours, I found it sad that these attempts and efforts are sometimes untracked. Thus, I introduced this to ensure my attempts are tracked for my own record.

During the project, I reported weekly and communicated with Dr. Adrian Andelin in emails times (). From GitHub, I made at least commits (major adjustments of codes), created versions of analysis code for differnet purposes.

From the GitHub daily coding graph (Figure ), I worked continuously for this new analysis project every week for tens of hours from April 13th 2016 when I was introduced to the traditional method.

From the GitHub coding punch card (Figure ), it seems I coded for this project spread out the days, mostly at night and early morning. This might be odd but to me, coding at night is the most productive since I found it serene and focusing.

In summary, I put in considerable amount of effort and time into this project, and I really value the trust and responsibilities Prof. Olavarria and Dr. Adrian gave me. I sincerely hope my endeavor and effort do help facilitate the analysis in our lab!

Using my method, running the whole MATLAB analysis for all 441 cases can be accomplished within 10 minutes (normally 5 minutes is sufficient), with only one click and a wait for 10 minutes before viewing our products in specific folders. Comparing to the traditional method costing 17640 minutes, my method is 1764 times faster and saves almost 300 hours of work!

Here is the GitHub repository of all the codes and my progress (Figure ): <https://github.com/doerlbh/OLab_IntanEphys/>

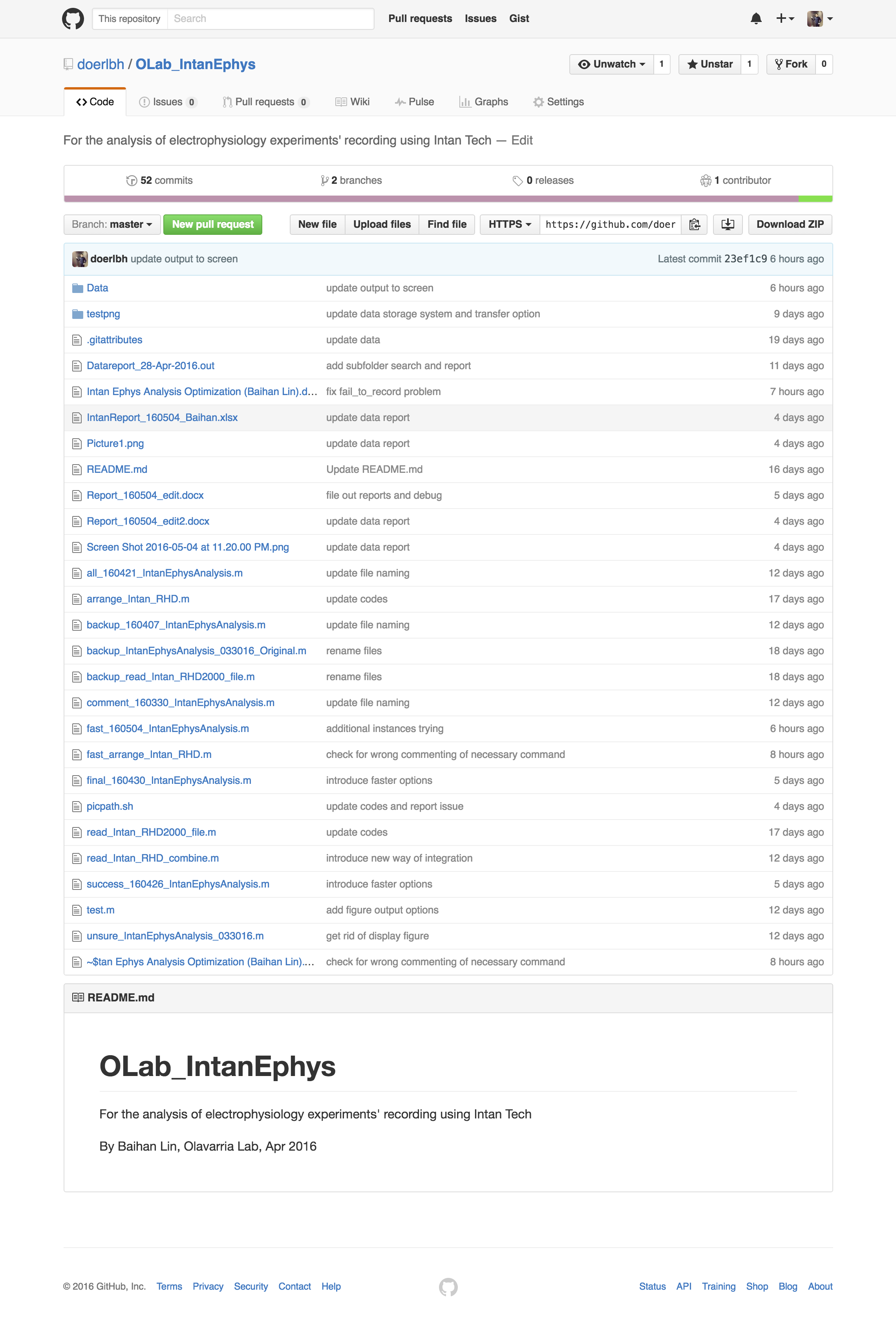
****

Figure . Screenshot of GitHub repository on May 8th, 2016

**Problem Identification:**

**(What to do)**

To study the effect of sensory deprivation on brain development, Using a combination of transneuronal tracing, in situ hybridization for the immediate early gene Zif268 and electrophysiological recordings, our lab recently showed that the primary visual cortex (V1) in pigmented rats has ODCs, and these ODCs correlate with callosal inputs from the opposite hemispheres. Using similar methods, my project aims to understand the effect of monocular deprivation (MD) on the newly discovered system of ODCs in rat visual cortex. However, introducing a new system of electrophysiology, RHD2000 Amplifier Evaluation System by Intan Technologies, we need a new way of analyzing its corresponding data format.

The RHD2000 Amplifier Evaluation System is a modular family of open-source hardware and software that allows users to record biopotential signals from up to 256 low-noise amplifier channels using RHD2000 digital electrophysiology chips from Intan Technologies. As shown in Figure , A USB interface board connects to a host computer via a standard USB cable. Small amplifier boards connect to the interface board via thin, flexible all-digital cables that may be daisy-chained to form robust connections up to 10 meters in length.



Figure , RHD2000 Amplifier Evaluation System

As shown in Figure , Open-source, multi-platform GUI software controls the operation of the amplifiers and streams data to the screen and to disk in real time at user-selected sampling rates from 1 kS/s to 30 kS/s.

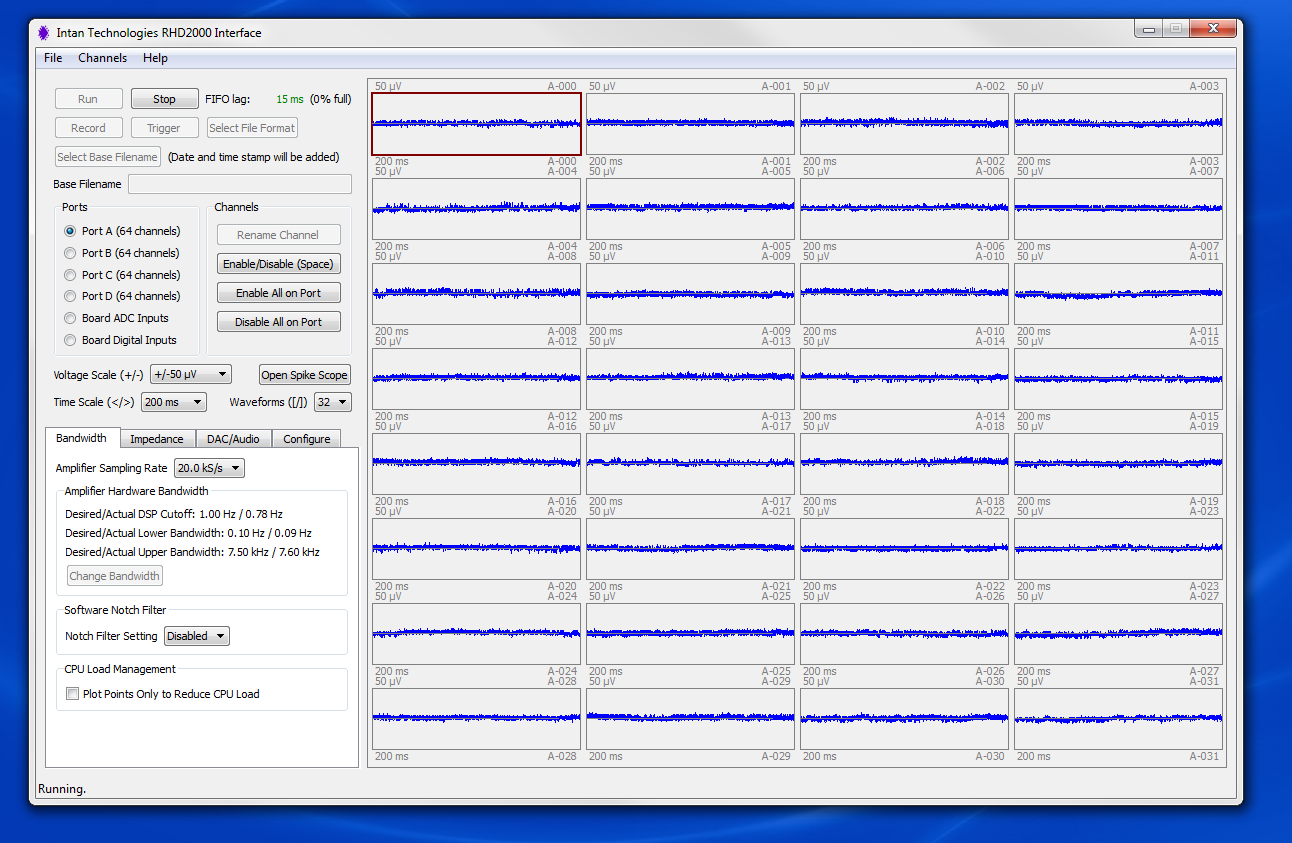


Figure , RHD2000 GUI interface

The RHD2000 Evaluation System allows users to perform the following functions:

* Monitor and record live signals from 16 to 256 low-noise amplifier channels using RHD2000 biopotential amplifier chips.
* Reconfigure amplifier bandwidths and sampling rates from software.
* Measure in situ electrode impedances (both magnitude and phase) at arbitrary frequencies with the click of a button.
* Use eight on-board digital-to-analog converters (DACs) to reconstruct analog waveforms from selected amplifier channels with <1 ms latency.
* Monitor audio of any two amplifier signals using a stereo "line out" jack.
* Record up to eight auxiliary analog inputs and 16 digital inputs synchronized with amplifier data.

These generated binary datasets which cannot be interpreted easily. And we need to use MATLAB in order to decipher the information and perform our customized analysis.

**Traditional Solution:**

**(Work but still room)**

The traditional solution developed by Intan Technologies and Dr. Adrian Andelin consists of two parts: reading the binary datasets and plotting with analysis.

**Step 1: generate Intan “.rhd” files**

From the RHD2000 Evaluation System, our electrophysiology data was recorded in segments of 60s, each includes input of signals from different channels.

**Step 2: read Intan “.rhd” files**

Intan files consists of binary or hex information like following (Figure ):

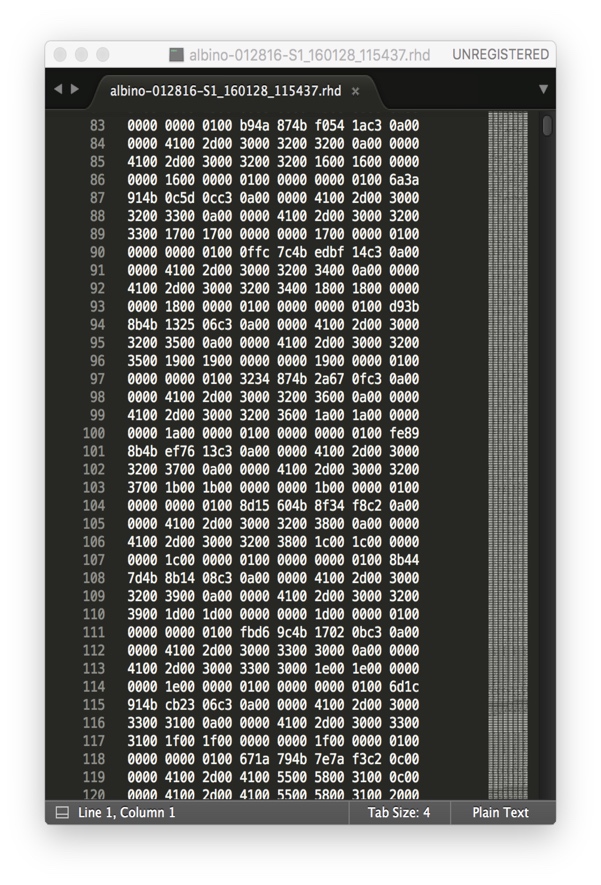
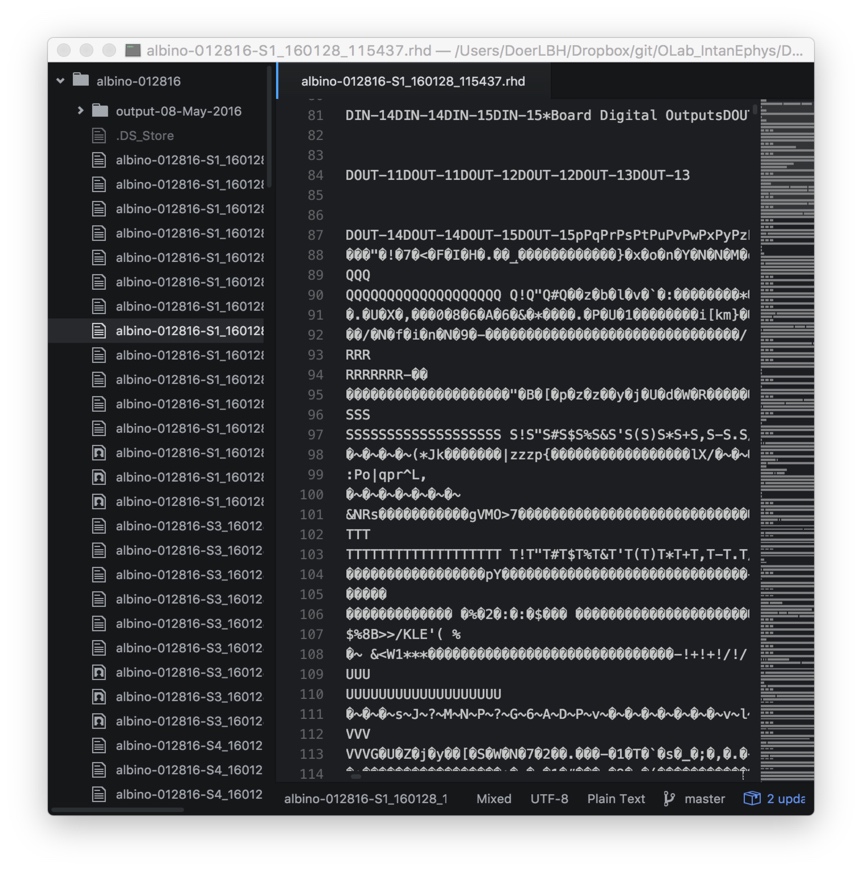


Figure . “.rhd” files are shown in binary (left) or hex (right), either unreadable

As shown, these “.rhd” files are not readable and has to be interpreted by official read\_Intan\_RHD2000\_file.m and it generated a series of variables (Figure ):

>> whos

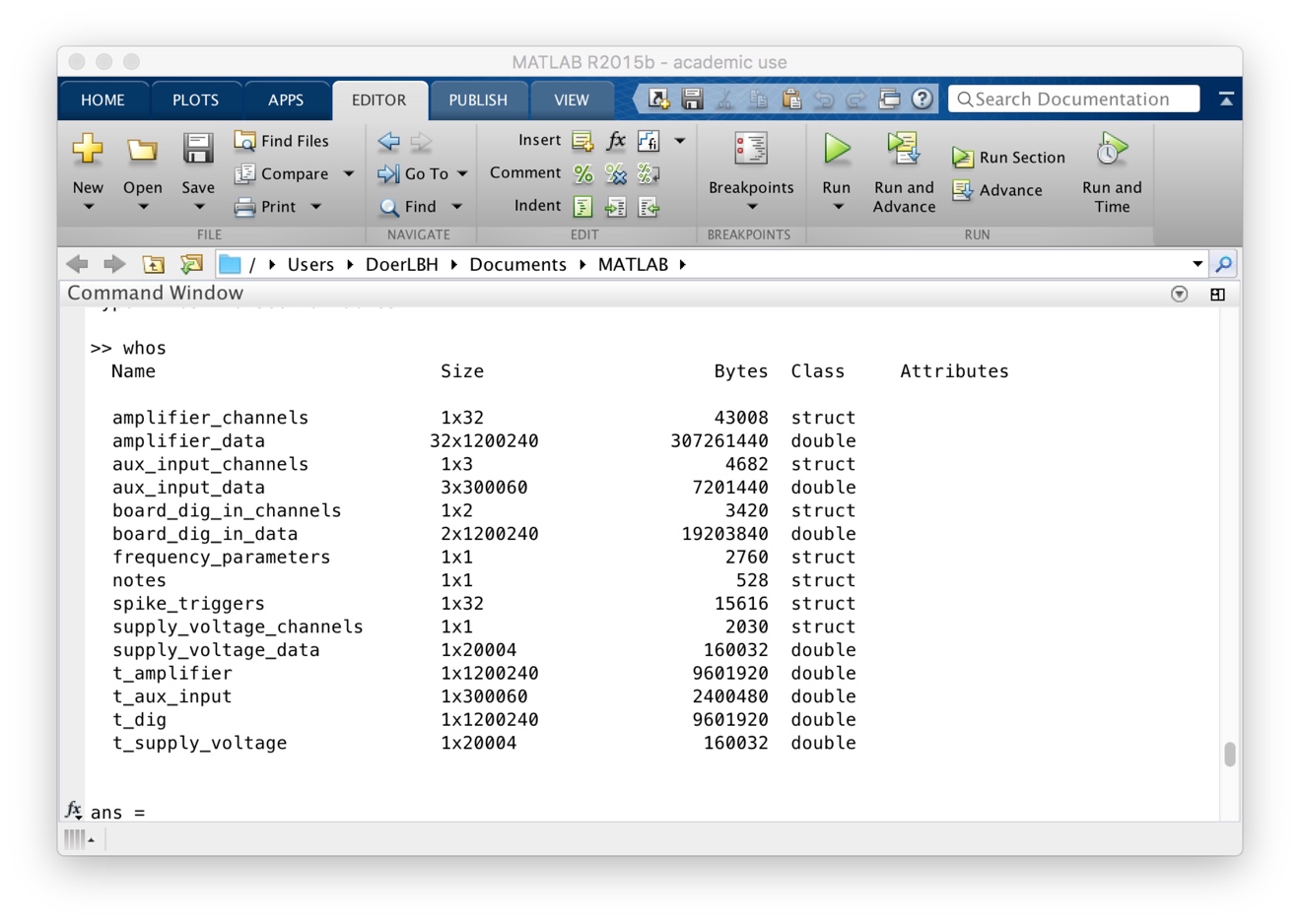


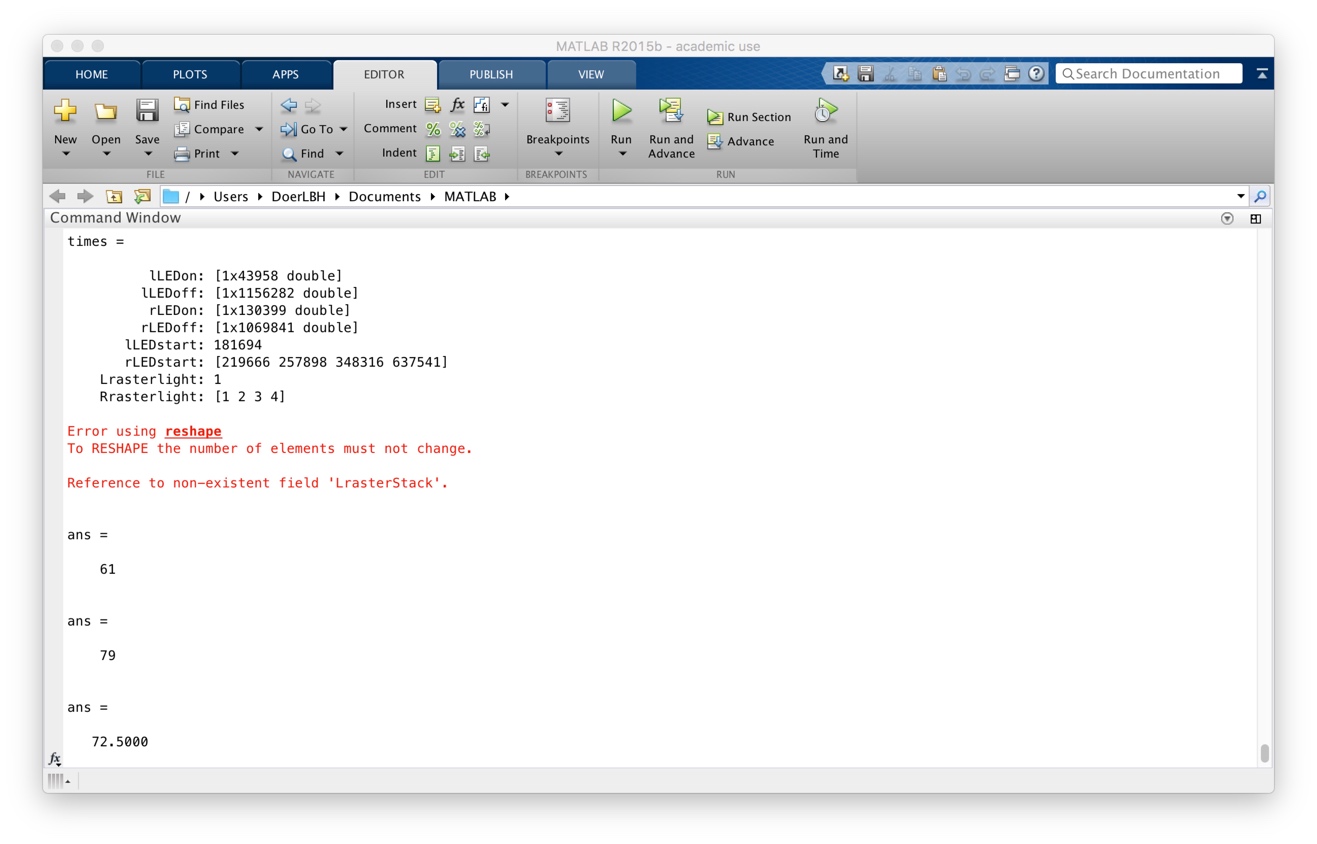
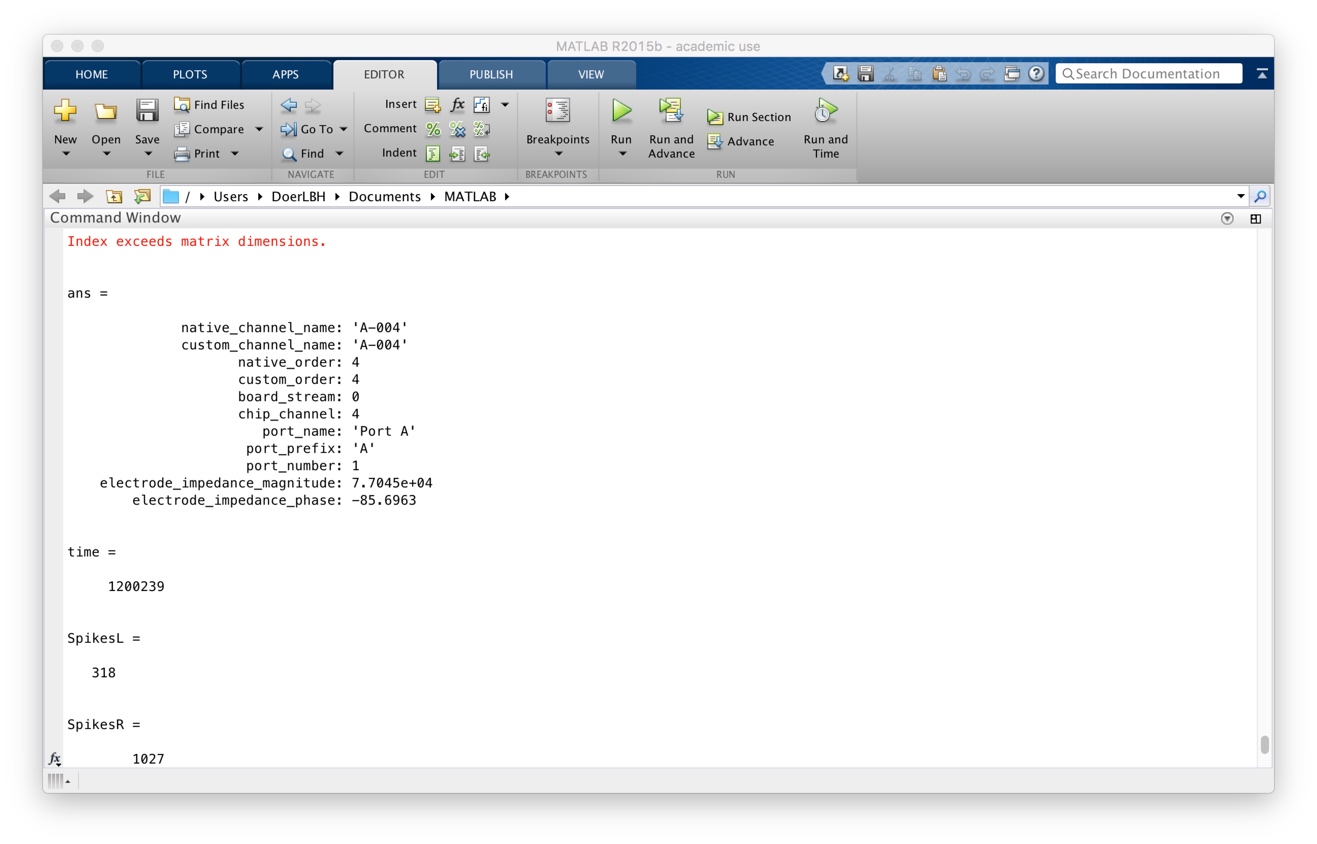
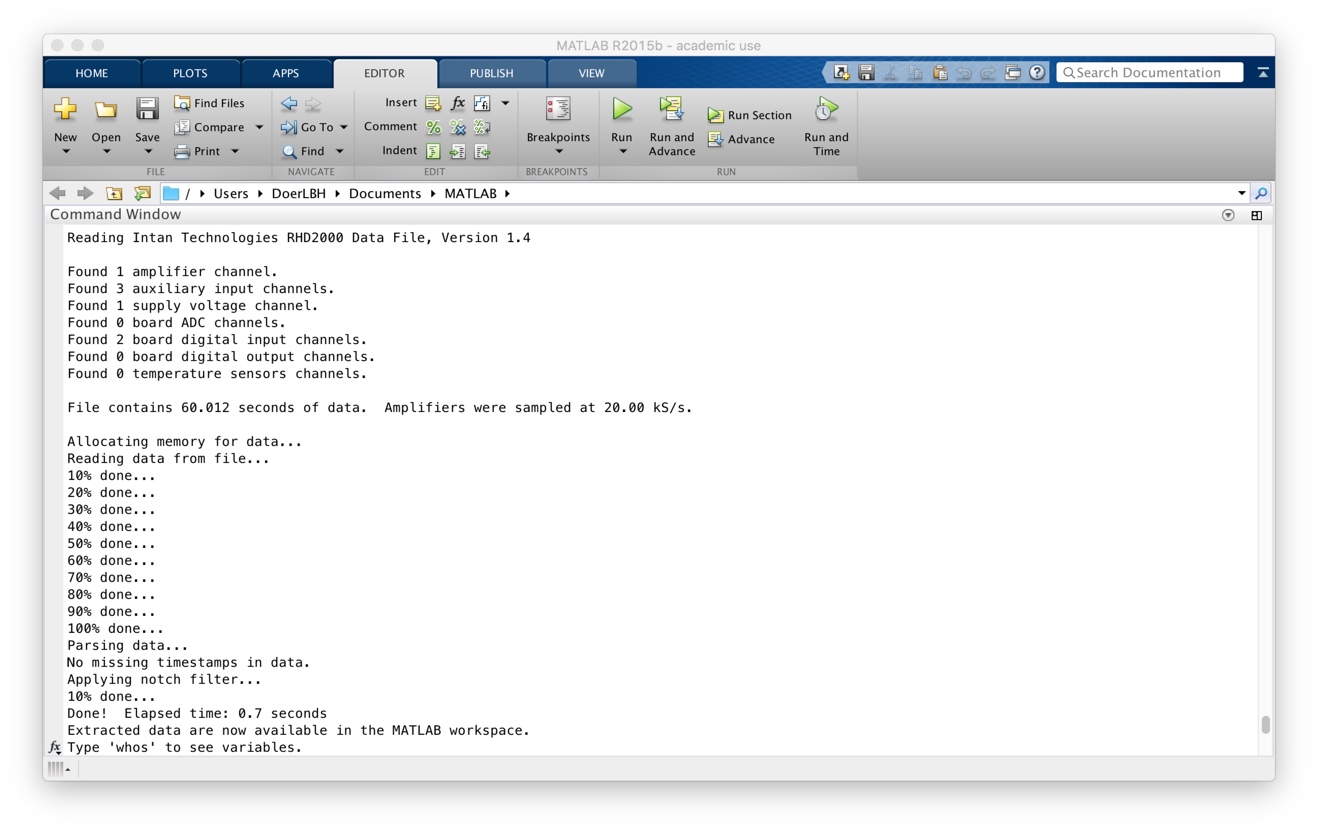
Figure . variables generated by read\_Intan\_RHD2000\_file.m

**Step 3: analyze variables extracted from “.rhd” files**

Running backup\_IntanEphysAnalysis\_033016\_Original.m, I encountered multiple issues, mostly due to MATLAB version discrepancy. And after debugging them, finally I made it run in backup\_160407\_IntanEphysAnalysis.m, whose console window is shown below in Figure .

This version of fixed analysis method, as implied in Figure of this instance, still encounters various problems analyzing different cases of “.rhd” files, including but not limited by “Index exceeds matrix dimensions”, “Reference to non-existent field 'Lrastercell'” etc..

This is due to the implicit nature of different cases of electrophysiology recording reflected by “.rhd”. And to ensure success, one needs to run separate steps and modify various locations of parameters in order to make it run smoothly by getting rid of the errors.



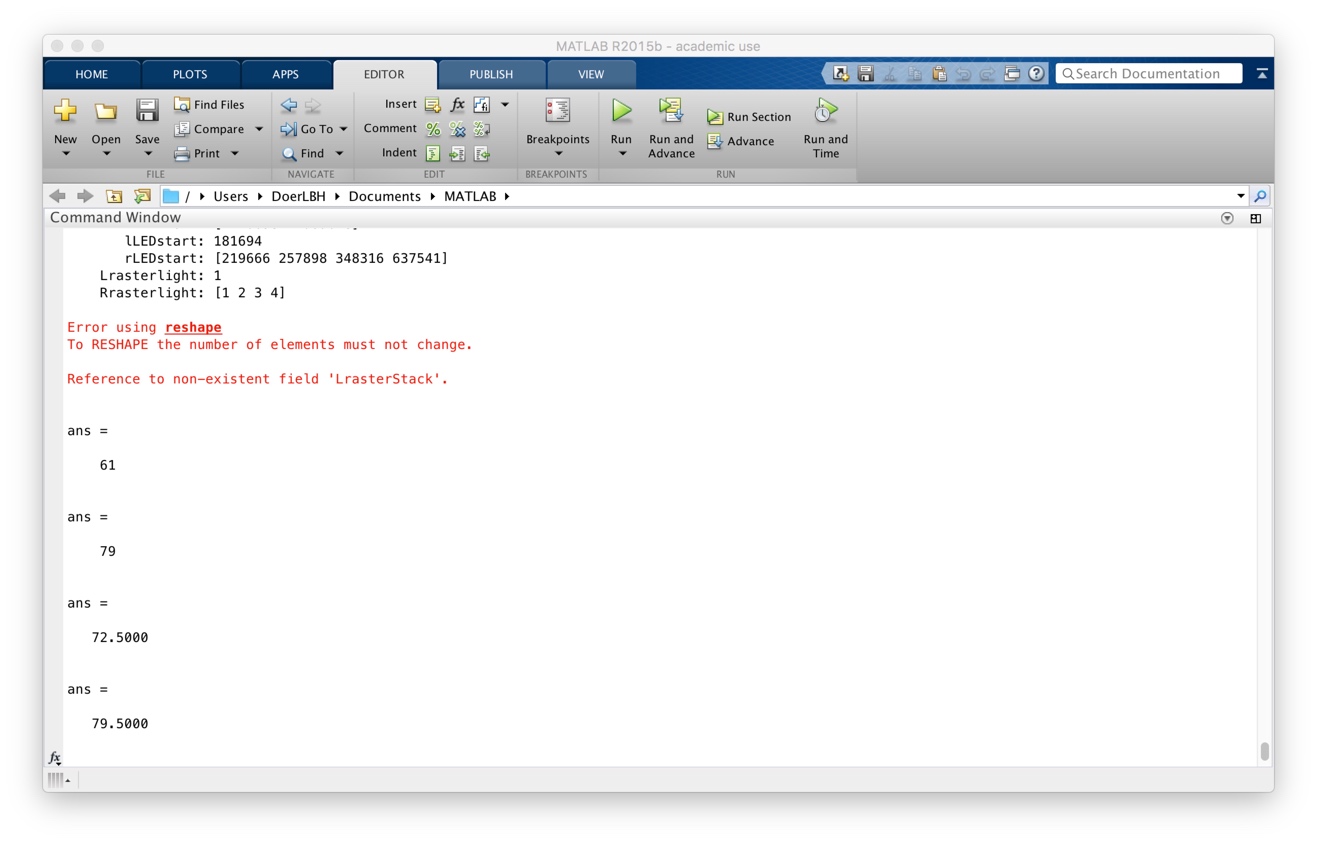
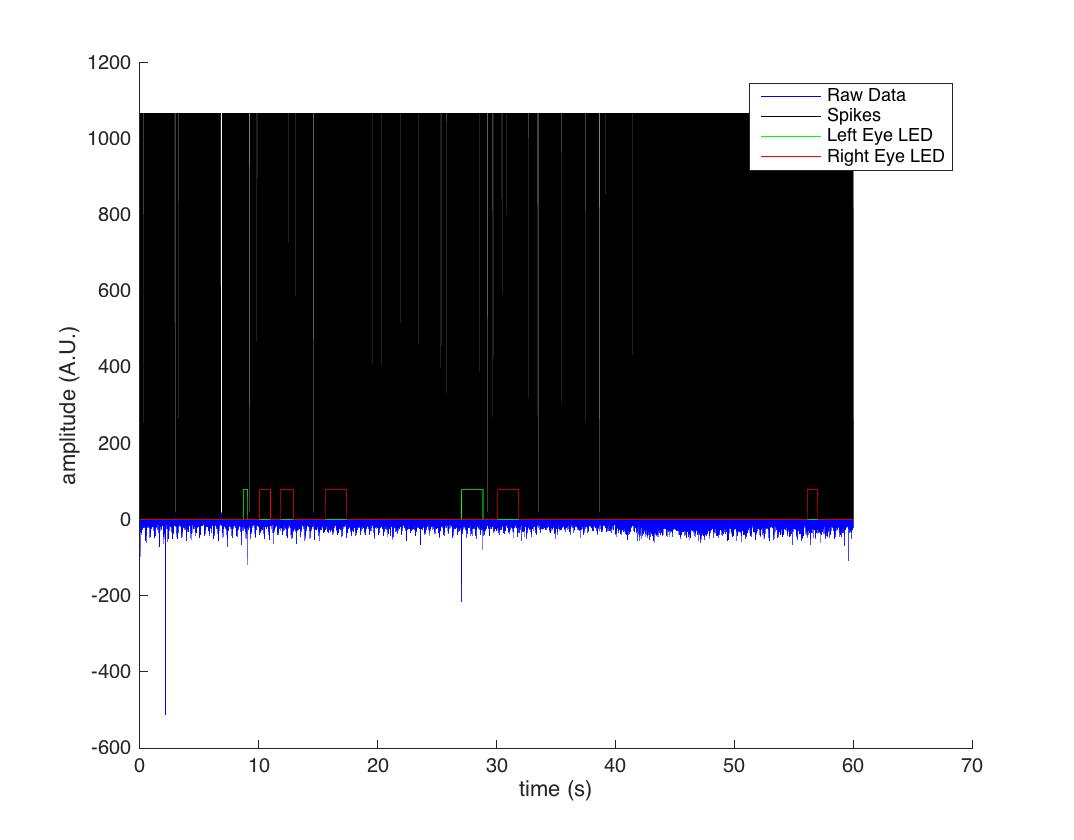


Figure . Console window of the traditional method on a test “\*.rhd” file

**Step 4: export raster plots and recording spikes plot**

As shown below in Figure , there is the spikes plot and raster plots for left eye and right eye after multiple modifications in parameters in each cases. However, to be noted, this is only



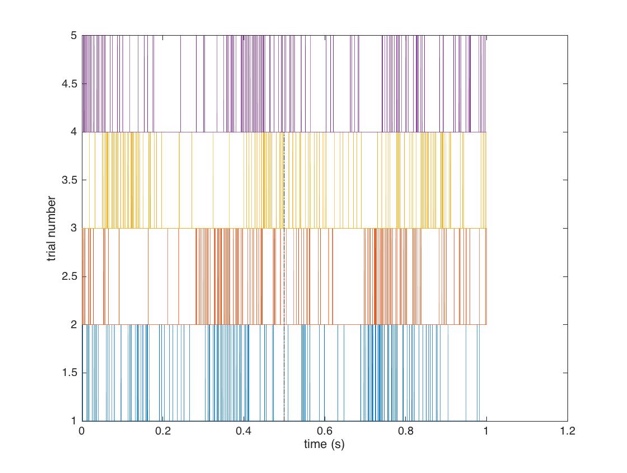
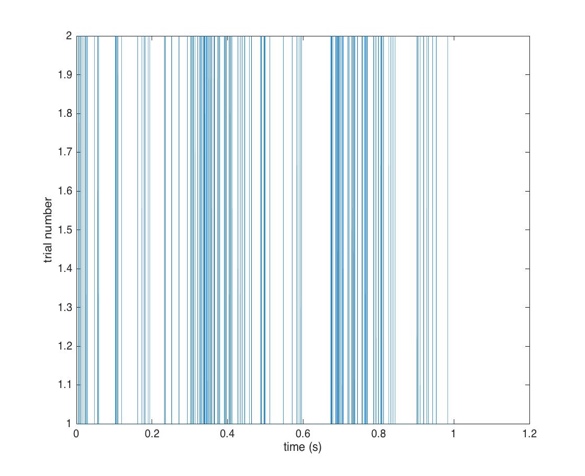


Figure . spikes plot and raster plots for left eye (left) and right eye (right)

**Issues to tackle:**

**(Efficiency & Validity)**

1. **Efficiency**

The traditional solution developed by Intan Technologies and Dr. Adrian Andelin consists of two parts: reading the binary datasets and plotting with analysis.

Take current data collected from lab in the past half year as an example:

011216 (ignored due to failed testing of Intan equipment)

albino-012816 129 cases

albino-020116 139 cases

MD-012516 62 cases

Normal-012616 111 cases

**In total 129 + 139 + 62 + 111 = 441 cases**

If one use this method, taking an average analyzing time of one instances for 40 minutes (for running the program, changing variables based on different properties of each case to make it work, exporting the images), he or she would be spending 441 \* 40 minutes = 17,640 minutes = 294 hours = 12.25 days (for 24 hours) = 14.7 weeks (for 20 hours per week) = 147 days (2 hours per day). This is terrifying for a lab to analysis only four rats.

1. **Validity**

As shown in previous graphs and analysis

**Attempt 1: Mac Terminal Concatenation:**

**(failed)**

**Perhaps we can concatenate the “.rhd” files of the same recording instance into one single file.**

I tried below (Figure ), making the assumption that “.rhd” files are simply headless binary datasets that can be easily combined.

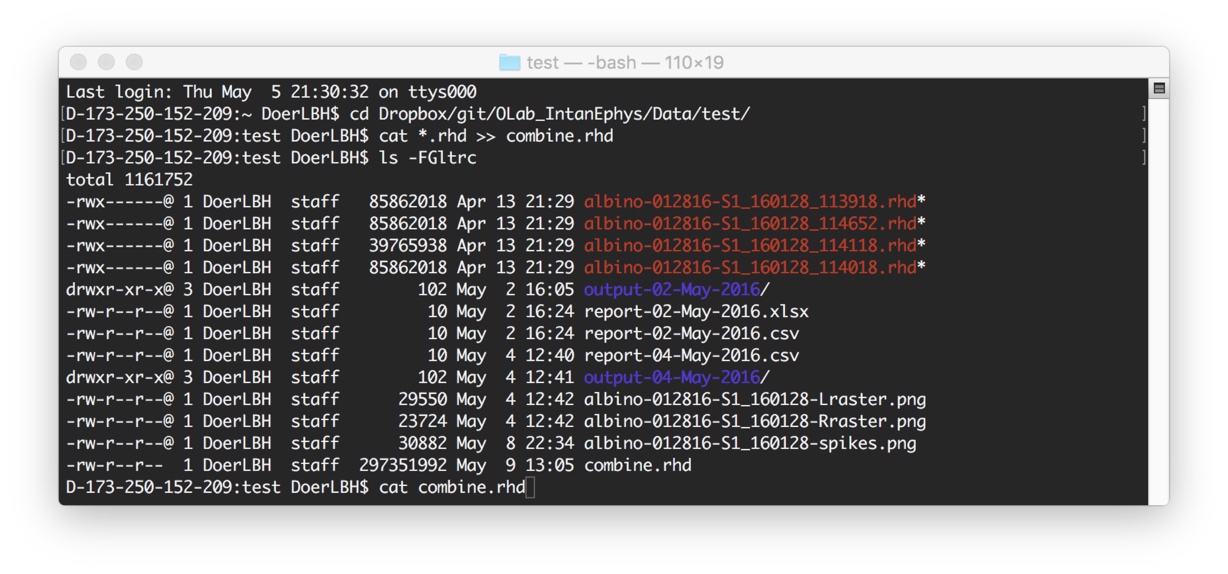
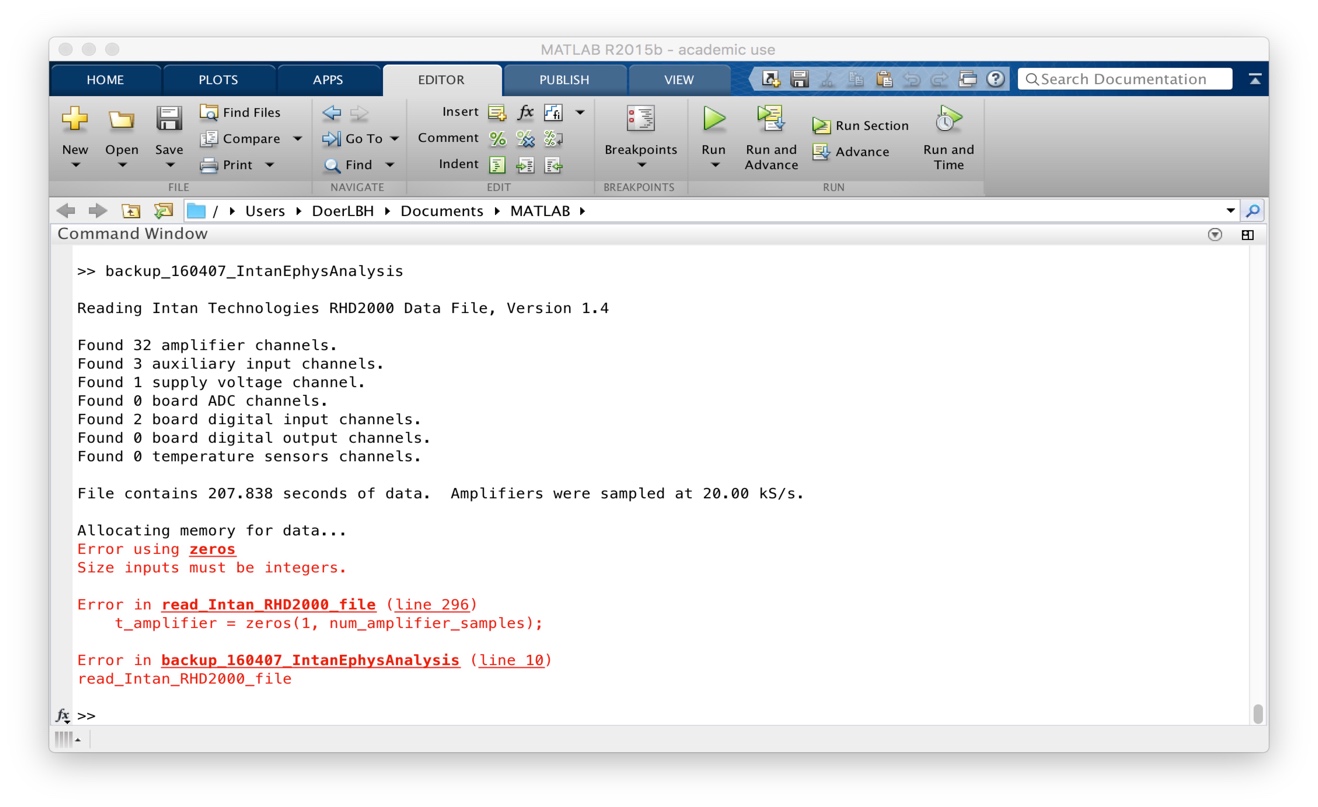
****

Figure . Mac terminal command to concatenate “.rhd” files

My assumption was wrong (Figure ), so we need to combine them later with MATLAB.

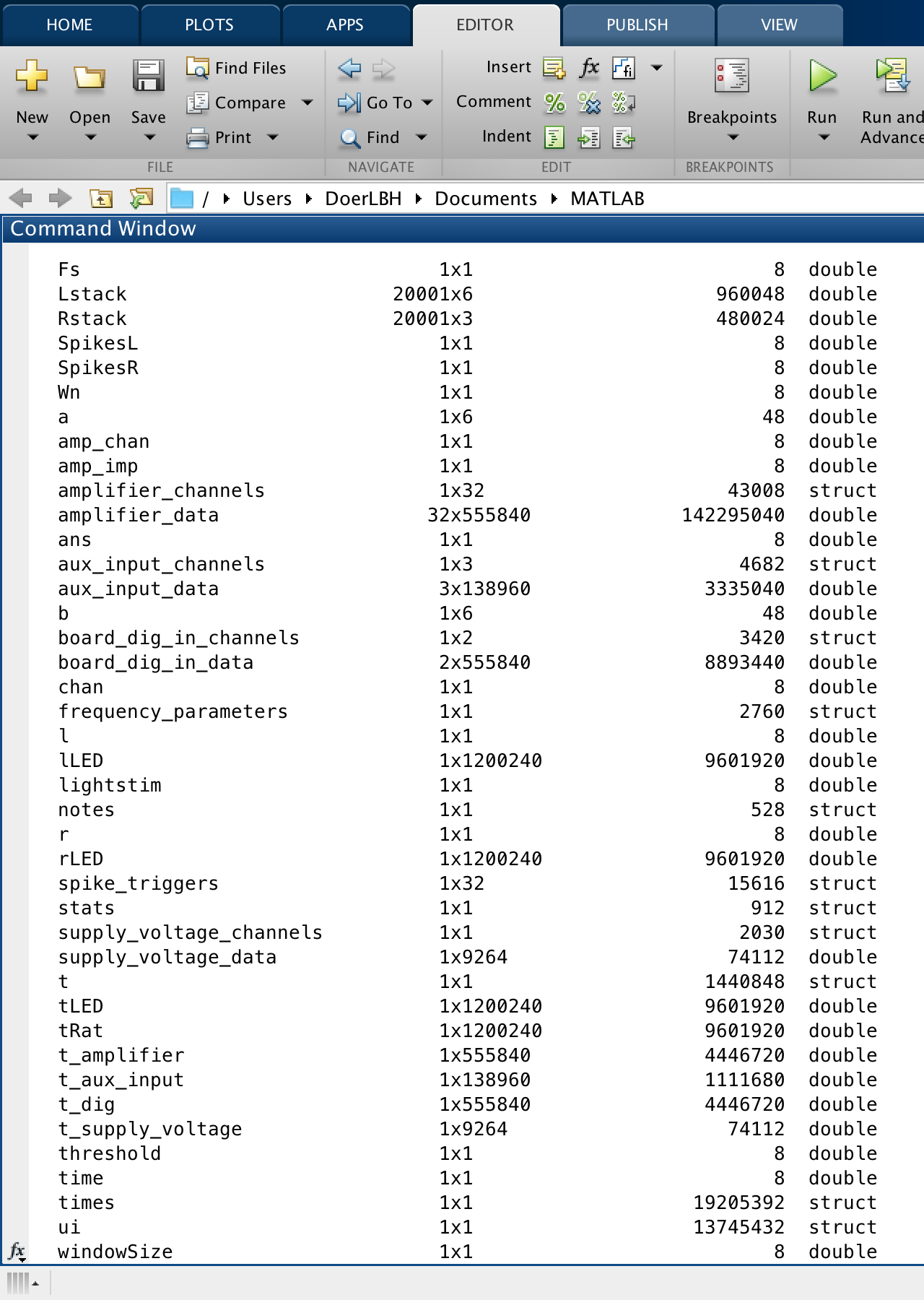
****

**Attempt 2: MATLAB Combination:**

**(success)**

**Perhaps we can concatenate the “.rhd” files of the same recording instance into one single file.**

I tried below (Figure ), making the assumption that “.rhd” files are simply headless



Fs

Lstack

Rstack

SpikesL

SpikesR

Wn

a

amp\_chan

amp\_imp

amplifier\_channels

amplifier\_data

ans

aux\_input\_channels

aux\_input\_data

b

board\_dig\_in\_channels

board\_dig\_in\_data

chan

frequency\_parameters

l

lLED

lightstim

notes

r

rLED

spike\_triggers

stats

supply\_voltage\_channels

supply\_voltage\_data

t

tLED

tRat

t\_amplifier

t\_aux\_input

t\_dig

t\_supply\_voltage

threshold

time

times

ui

windowSize

amplifier\_channels 1x32 43008 struct

amplifier\_data 32x555840 142295040 double

aux\_input\_channels 1x3 4682 struct

aux\_input\_data 3x138960 3335040 double

board\_dig\_in\_channels 1x2 3420 struct

board\_dig\_in\_data 2x555840 8893440 double

frequency\_parameters 1x1 2760 struct

notes 1x1 528 struct

spike\_triggers 1x32 15616 struct

supply\_voltage\_channels 1x1 2030 struct

supply\_voltage\_data 1x9264 74112 double

t\_amplifier 1x555840 4446720 double

t\_aux\_input 1x138960 1111680 double

t\_dig 1x555840 4446720 double

t\_supply\_voltage 1x9264 74112 double

amplifier\_data 32x555840 142295040 double

aux\_input\_data 3x138960 3335040 double

board\_dig\_in\_data 2x555840 8893440 double

supply\_voltage\_data 1x9264 74112 double

t\_amplifier 1x555840 4446720 double

t\_aux\_input 1x138960 1111680 double

t\_dig 1x555840 4446720 double

t\_supply\_voltage 1x9264 74112 double

/Users/DoerLBH/Dropbox/git/OLab\_IntanEphys/Data/test

**Attempt 3: Excel Report Integration:**

**(success)**

Now having accomplished the combination, analysis and plotting, I want it to be able to export to Excel worksheet

Using my method, running the whole MATLAB analysis for all 441 cases can be accomplished within 10 minutes (normally 5 minutes is sufficient), with only one click and a wait for 10 minutes before viewing our products in specific folders. Comparing to the traditional method costing 17640 minutes, my method is 1764 times faster and saves almost 300 hours of work!

In addition, the

**Data Summary**

1. **Excel report**
2. **Figures**
   1. albino-012816
   2. albino-020116
   3. MD-012516
   4. Normal-012616

**Frequently Asked Questions (FAQ)**

1. **Why do I don’t have enough plots generated?**

This can be due to different reasons.

A common mistake is to put the parent folder instead of the innermost subfolder as your input. If so, the program can mistakenly assign wrong attributes. Be sure to choose the innermost subfolder to start with.

Another common reason can be a lack of LED input in the first place. This happens when both or one of the LED input is missing. Other possibilities will need more debugging.

1. **Why are there blue horizontal lines across the spike plots?**

This is due to the concatenation and it is normal.

**Code Summary:**

**Bibliography:**

Intan Technologies, <http://intantech.com/RHD2000_evaluation_system.html>

Driscoll, T. A. (2009). *Learning MATLAB*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Feng, J. (2004). *Computational neuroscience: Comprehensive approach*. Boca Raton: Chapman & Hall/CRC.

**Special Acknowledgement:**

Thank Prof. Jaime Olavarria and Dr. Adrian Andelin for giving me the opportunity to analyze his data! In order to present her the most convincing data, I am lucky to review, utilize and summarize my ability of information searching, computational modeling and conception actualization. In the entire process of developing ideas into software, I learnt a lot from the questions and reflections at every step and had the fortune to enjoy the excitement of tackling the challenges one by one! Her tolerance on the deadline and encouragement on the difficulty give me the strongest fortitude to proceed! Her usual strict and meticulous style forces me to think deeper and striver closer to perfectionism!

Thank Prof. Jaime Olavarria for introducing me with insightful lectures into this dynamic field of biopsychology and neuroscience and letting me getting involved in the challenging projects of his warm lab! (It is my honor to be able to be peer TA in his class next quarter! Thank him to give me courage to challenge Psychology Honors Program next quarter!)

Thank my friends in the lab, Adrian and Melissa who support me with sunshine! Adrian never fails to encourage me whenever I make any mistake or face some challenge. He taught me how to flexibly tackle incidents in research but still gave me chance to shoulder experimental rats.

Thank University of Washington for giving us the platform to scientifically explore practical academic problems interdisciplinarily!

I will continue the voyage of exploring the infinite realm of neuroscience in my academic career fearlessly.

Baihan Lin

April 2016