

UT SOUTHWESTERN MEDICAL CENTER



The Tutorial

Jui-Jui Lin

Principal Investigator:
Bradley LEGA M.D.

Last Revised: May 23, 2018

Contents

1	Introduction	1
2	BioHPC	1
2.1	Getting Access	1
2.2	Directory organization	1
2.3	Working on the cloud	2
2.4	Running parallel jobs	4
2.5	Access from off campus	4
3	Analysis of EEG data	4
3.1	Basic signal processing background	5
3.2	Organization of data	5
3.3	Raw EEG and event-related potential (ERP)	6
3.4	Oscillatory power	8
3.5	Connectivity Analysis	8
3.6	Statistical testing	8
4	Exploring Folders on your Apple Computer Using Terminal	9
4.1	Common Command Line Functions	9
5	Data Collection	10
5.1	Verbal Tasks	10
5.2	Spatial Tasks	10
5.3	Stim-Only Tasks	10
6	Data Processing	10
6.1	Memory Testing	11
6.2	Obtain EEG recording	11
6.3	Split EEG Recording	11
6.4	Making Event Structures	11
6.5	Align Events	12

1 Introduction

Welcome to the lab. As you will quickly find out, there are a ton of things you need to be familiar with in order to become a productive and contributing member. This is a tutorial designed to cover a little bit of everything that goes on in the lab, including getting set up on BioHPC, running analysis, patient testing, and more.

This is a programming heavy lab that uses mainly Matlab. Prior experience with it is a plus, but definitely not required; plenty of people have gone through the lab without having coded, but you are expected to learn quickly. The best thing to do to get into programming is to just do it. I've provided plenty of template analysis and plotting scripts on BioHPC that will guide you through the basics. That being said, the learning experiences in programming comes not from running code that works, but comes from fixing code that doesn't. You will get stuck a lot in the beginning, and that's ok. You will never get better if you don't struggle through debugging your own code. Luckily, Matlab is extremely user friendly and has a ton of documentation online and offline. The 'help' function is one of the biggest tools at your disposal when you run into a function you're not familiar with (e.g. type in 'help reshape' to see documentation on the 'reshape' function). Most of the time you can even type in questions in your mind verbatim into Google and find the answer (e.g. "How to average values in MATLAB?", "How to loop through files in a folder using MATLAB?", or "How to plot a line in MATLAB?"). Try spending a reasonable amount of time troubleshooting an error before asking for assistance, and any of the lab members will be happy to help. If you are a more experienced Matlab user, the templates are there merely as a guide, as you become more familiar with analysis, you can write scripts to run code and process them however you want.

If you notice any issues with this tutorial (unclear instructions, typos, etc.), please email jui-jui.lin@gmail.com. This tutorial is meant to be updated as much as necessary to develop a guide that will help new members of the lab develop the skills necessary to perform complex neural signal analyses. If you have a question, it is likely that someone in the future will too and it would be helpful to answer as many questions as possible in the context of this guide.

2 BioHPC

BioHPC is the high-performance computing group here at UT Southwestern. They provide computational resources, workshops, most importantly, cloud storage options for research labs on campus. This section is meant to be a very brief overview on how to get started on the cloud and will not teach you how to use every single resource BioHPC offers. There are great documentations and powerpoint slides on their website (<https://portal.biohpc.swmed.edu>) on how to use everything they have. If you have any questions, you can also contact their support team (biohpc-help@utsouthwestern.edu).

2.1 Getting Access

To obtain access, first register for an account on their website. Your account will be activated after attending the orientation workshop, which takes place on the first Wednesday of every month. Email support to sign up for the session.

2.2 Directory organization

Once you have your account activated, you will gain access to 3 directories: /home2, /work, and /project. /home2 will be your home directory and your own folder will be at /home2/[your_utswh_id]. I do not recommend using this directory besides as a back up for your code, since it only has a limit of 50 GB. You'll want to do all your analysis and save out results (plots, matrices, etc) at your work directory /work, which has a 5 TB limit.

Finally there's the project directory `/project`, where data for every lab is stored. Obviously, you'll only have access to our own lab's data, which is at: `/project/TIBIR/Lega_lab/shared/lega_ansir/`

The only folders that you should really be familiar with is `'shared_code'` and `'subjFiles'`. `'shared_code'` contains a master `eeg_toolbox` for the lab (for all your analyses), data processing functions (event-making), and lab tutorials for various tasks. `'subjFiles'` is where all our subject data is stored and its individual folders are organized as follows:

- **behavioral:** Behavioral data of all tasks the patient performed. Organized as event structures.
- **docs:** Miscellaneous text files with electrode labels (jacksheets, `depth_el.info`, etc)
- **eeg.noref:** Raw eeg files of all the recorded channels during testing session
- **eeg.reref:** Same as `noref`, but the values are recalculated using an average rereferencing scheme. These are the files the event structures are aligned to.
- **freesurfer:** CT and MRI files
- **raw:** EEG files of the testing sessions directly from the clinical recording systems (`.EEG`, `.21E`, `.ns2`).
- **tal:** Information on localization of electrodes. Includes a structure that gives Talairach coordinates of individual electrode contacts.

2.3 Working on the cloud

There are several ways to access data from the lab directory. The most efficient way is to work on the cloud directly by using a Web Visualization from the BioHPC website, in which you'll be able to use one of the nodes on the cluster (essentially having remote control of a more computer that's connected to the cloud server). To use this feature you'll first need to download TurboVNC (<https://www.turbovnc.org>), which is a program that sets up the GUI and when you start the Web Visualization from the webpage, it'll give you a node address and a password to enter into TurboVNC when it prompts for them. If everything worked correctly, you should see a Linux desktop pop up (Figure 1). The GUI will function like a regular computer with a plethora of software (Matlab, Python, R, etc) preinstalled. To start a program, open up a terminal window by going to **Applications -> System Tools** at the top left-hand corner of the screen. For this example, we will try to start an instance of Matlab. First we will look at what Matlab modules are available to us. In the terminal prompt, type in: `module avail matlab`, and all the available Matlab versions should populate. Decide on your favorite version (anything besides 2017, bug explained in later section), and type in: `module load matlab/[version]` and then `matlab` to start it. You can start any module in the same way, as long as you load them properly. For the complete list of modules available, type: `module avail`.

Another useful method of accessing data on BioHPC is mounting the cloud directories to your local device. To do this on a Mac, open up any finder window and click on **Go -> Connect to Server** on the toolbar on the top of your screen. Here you'll have the options to mount your `/home2`, `/project` or `/work` directories using the following server addresses, respectively:

```
smb://lamella.biohpc.swmed.edu/<username>
```

```
smb://lamella.biohpc.swmed.edu/project
```

```
smb://lamella.biohpc.swmed.edu/work
```

You'll then be prompted to enter your BioHPC account info to log in. This method is especially suitable when moving small amounts of files to and from the server, accessing your analysis outputs, and plotting offline. In fact, It is ill-advised to do the latter two tasks directly on the cloud, as you will often have hundreds if not thousands of output files and it's incredibly slow to navigate through them if you're trying to troubleshoot them on the server. Plotting with Matlab on Biohpc is also poorly optimized; you'll often run into unusual formatting issues that you would not otherwise experience offline as the nodes sometimes

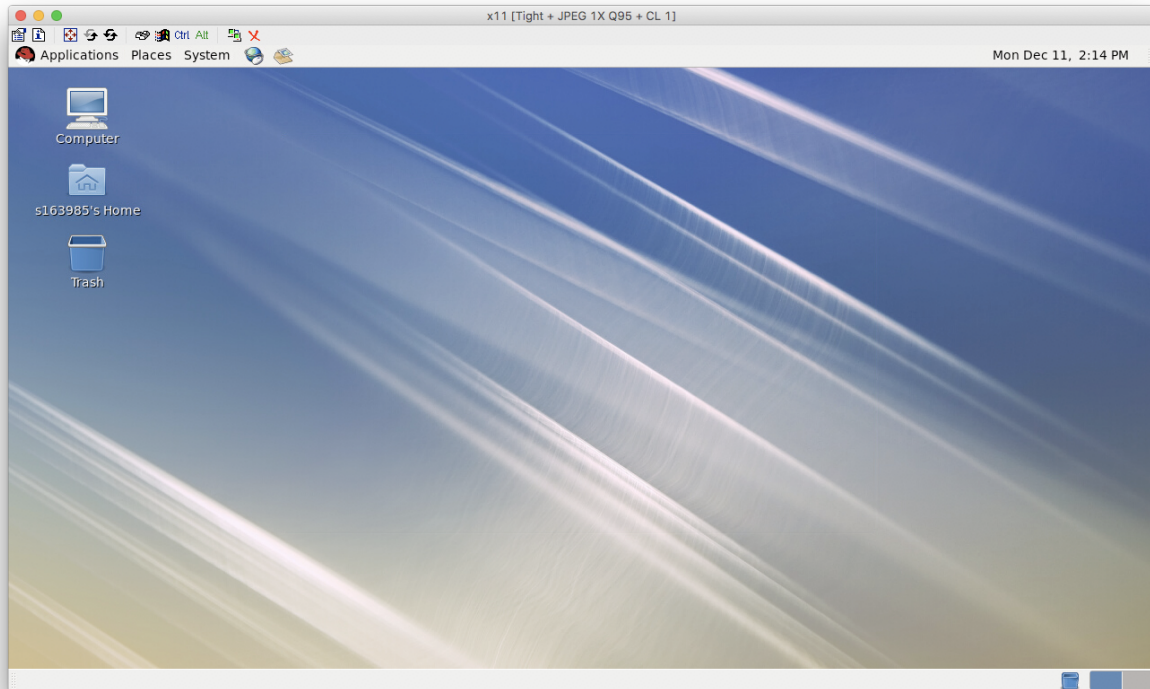


Figure 1: Linux Desktop.

lack proper graphics drivers. For mounting on Windows machines, see the BioHPC introductions page (<https://portal.biohpc.swmed.edu/content/guides/introduction-biohpc/>).

Matlab setup on the cloud We just went over how to start up Matlab on Linux, but without a few setup steps, every time you open up the program it will be as if it is freshly installed without any user preferences. The default initial folder Matlab recognizes is your home2 directory, and as mentioned earlier in this text, it is more efficient to work from and save your outputs to your work directory. To modify this setting, click on the **home** tab and go to **preferences**. On the left hand side, select **General**, and click on the custom option under **initial working folder** and navigate to your /work folder using the ... button on the right. Click **apply** and **ok** to confirm changes. Currently the 2017 versions of Matlab seems to be bugged and the initial working path can't be modified, so please pick an earlier version. The next thing to do set up paths for Matlab to recognize where your functions, scripts, and toolboxes are going to be. Make a new script from the home tab, name it **startup.m** and save it to your initial working directory. Inside the script, you want to enter two lines of code:

```
addpath(genpath('/work/TIBIR/Lega_lab/[your_id]'));  
addpath(genpath('/project/TIBIR/Lega_lab/shared/lega_ansir/shared_code'));
```

This script is what Matlab will run every time during startup and load code from your work directory and the lab master EEG toolbox without you having to do this manually. Lastly, restart Matlab and the terminal to confirm the change in initial working directory in the displayed current folder. You can check if your code did actually get added by using the **which** function (e.g. `which gete_ms.m`). If the output returns a directory, then the start script is working properly. It should be noted that these setup steps only apply to currently Matlab version you are using, i.e. if you went over these steps for 2016b, the changes would not be reflected when you try to start another version; therefore, you should pick and stick with one version.

For the purpose of executing any tutorial code, you should copy the `shared_code` folder to your local directory, add it to your local Matlab's path, and have the lab project directory mounted.

2.4 Running parallel jobs

***All the code and scripts you need to do this is in `/shared_code/tutorial/parallel_jobs*`**

One of the greatest utilities that BioHPC offers is the ability to perform computations in parallel. Think about how a for loop is processed in series with one iteration followed by another. Parallel computing treats different iterations of the same loop as multiple jobs and performs them simultaneously. To take advantage of this feature, you'll need to organize your script to a specific format:

- The script should center around one main for loop to be processed with the main body of the code embedded inside a try/catch statement. This will allow iterations of the loop to become independent of each other and any error that occurs in one will be recorded and not interfere with the progress of another.
- The inputs into the script (e.g. subject, electrode, regions, etc) should be organized into a Matlab structure (see `makeElecStruct.m` in `/tutorial/analysis`) for sequential processing in the main for loop.
- Iterations of the loop should generate a 'lock' file if it was attempted/performed, a 'done' file if completed without error, and an 'error' file if an error occurred. These files are just empty text files that give insights to the progress/result of the loop based on their names.

Refer to `lockfile_template.m` in the tutorial folder for more information on this setup.

To submit a job to the cloud, you'll need to use a couple of shell scripts: `srunScript.sh`, and `matlabWrapper.sh`. You'll want to copy both of these files into your work folder to use as your own. The first script includes the parameters to submit the job (refer to job submission tutorial on the BioHPC homepage for more info on this), and calls on the second script, which contains the actually command to start a job. The parts that need to be modified for your own use is labeled on the templates. After you make the changes, use the Nucleus Web Terminal on the homepage to submit the job by typing:

```
sbatch /work/[location_of_your_srunScript.sh]
```

2.5 Access from off campus

There is the option to use BioHPC off campus which will require you to get on the campus network. To do this, you'll need to download Pulse Secure (<http://www.utsouthwestern.net/intranet/administration/information-resources/network/vpn/>) and follow the instructions on the intranet page for the installation and configurations. Once setup is complete, start the program and add a connection to server `?utswra.swmed.edu?`. After you enter your UTSW password, it will prompt you for a secondary password, which you can get through the Duo Mobile app. When the program confirms connection to the campus network, you can then access BioHPC as you would normally.

3 Analysis of EEG data

In computational neuroscience, analyses often revolve around investigating signal characteristics between trials of various conditions. In the study of memory, the focus is on what our brain signals look like when we remember vs. when we don't remember something. This can be done on two time periods during a memory experiment: the encoding and retrieval period. An analysis on the encoding period seeks to find differences in brain states after a stimulus or item is presented, and an analysis on the retrieval period focuses on differences in brain states before an item is recalled. The conditions in the encoding period are categorized into Recalled and Non-recalled, which reflects the subject's later response to item. The analogs in the retrieval periods

are Correctly Recalled and Incorrectly Recalled (Intrusions). It's important to try to keep the terminology consistent so that methods of an analysis can be clearly explained and understood.

3.1 Basic signal processing background

One of the most important element of signal processing is a mathematical concept called Fourier Theorem. It states that any continuous time-varying signal can be expressed as the sum of a series of sinusoids of varying frequencies, amplitude, and phase shifts (Fourier Series). In time-frequency decomposition we run into the reverse problem: we have signal and want to break it down to its individual frequency components. How this is done is through a linear operation called Fourier Transform, or Fast Fourier Transform (FFT). In this process, multiple dot products are calculated between a signal of interest and sinusoids of a range of frequencies in order to visualize how much of that frequency component exist in the signal. Through a FFT, the signal is transformed from the time-domain into the frequency-domain with no time information preserved, because sine waves have infinite length. A solution for preserving temporal precision during a transform operation is to use a finite kernel in calculating the dot products, and one of the most common way is with a wavelet. Wavelets, specifically the Morlet wavelet is a sine wave modeled by a Gaussian (normal distribution), provides an excellent means of localizing frequency information in time in terms of the control the user has over the tradeoff between temporal and frequency precision, which is the predominate method we use in this lab. This is a very rough explanation of how a signal is processed, but the main takeaway is that we breakdown signals to its individual frequency components to understand how these patterns change over time during the encoding and retrieval periods of a memory task. Although understanding of the minute arithmetic details behind time-frequency decomposition is not necessary in performing analysis, it is prudent to at least have a good sense of what these algorithms are doing to your signals in order to have a better idea of the overall process. For more information on decomposition and wavelet theories, refer to chapter 11 of Cohen.

3.2 Organization of data

The EEG data we collect from our subjects are all organized and formatted in the same way: as event structures. These event structures are Matlab .mat files that contain a multitude of information about the task that was done, including patient performance, parameters used, and most importantly, the time offset values that align the EEG time to the Unix computer time of when the event occurred. Below is all the information included in the event structure from a Free-Recall (FR) task.:

- subject: this field contains the subject code that corresponds with the event in that row
- session: this field specifies the session number for the event in that row.
- list: this field specifies the list number within the corresponding session. If a subject participated in 3 sessions, there will be 3 list #1s. If the subject completed 1 full session and 2 partial sessions, there may only be 1 list #25. This field is -999 for events that are not a part of a list of words or the recalls corresponding to a list of words.
- serialpos: this field specifies the position of the word corresponding with that event in the corresponding list. The first word for list 1 will have a serial position of 1. The sixth word will have a serial position of 6. The eighth word of the eleventh list will have a serial position of 8 (and list 11).
- type: this field specifies what type of event the corresponding event was. The possible entries for this are below
 - B: the beginning of the EEG recording
 - SESS START: the beginning of the session on the testing computer
 - COUNTDOWN START: the beginning of the countdown from 10 to 0 that precedes each list
 - COUNTDOWN END: the end of the countdown that precedes each list

- PRACTICE WORD: a word that was presented during the first list of each session which is a practice list
 - PRACTICE DISTRACT START: the beginning of the distractor math problems for the practice list
 - PRACTICE DISTRACT END: the end of the distractor math problems for the practice list
 - PRACTICE REC START: the beginning of the 30 second recall period for the practice list
 - PRACTICE REC EN: the end of the distractor math problems for the practice list
 - TRIAL: this precedes any events for a new list
 - ORIENT: this corresponds to a fixation point on the screen (+) that precedes the list of words after the countdown.
 - WORD: the presentation of a word on a test list (these are the events to consider when doing an analysis on encoding).
 - REC WORD: a vocalized response of a word during the recall period. These entries are manually scored by research assistants in the lab.
 - SESS END: marks the end of the session
 - E: marks the end of the recordings
- item: this field specifies the specific word that was presented or recalled. This field contains an X for entries that do not correspond to a presented or recalled word.
 - itemno: this field specifies the number of the word presented or recalled from the wordpool. Analyses can be performed using the word number or the actual word entry in ?item?.
 - recalled: this field contains -999 for any entry that is not a word presentation. It has a value of 1 for words that were presented and remembered and a value of 0 for words that were presented and not remembered.
 - intrusion: was the item correctly recalled (intrusion == 0), a Prior List Intrusion (PLI, intrusion > 0), or an Extra List Intrusion (ELI, intrusion == -1).
 - eegfile: this field points to the EEG file that corresponds with that entry in the events structure. You will need to change this to point to the location of the EEG files where you saved the folder. The filename should be the same but the folder location will need to change.
 - set: this field specifies the number of milliseconds from the beginning of the EEG file where the event in that entry begins. This is used by functions in the eeg toolbox to determine what segment of the EEG data to analyze for a specific event.

3.3 Raw EEG and event-related potential (ERP)

An event-related potential is the change in voltage patterns during an 'event' of interest. These events can be anything from any stimulus presented to the subject to any responses we get from them during a memory task. The idea behind an ERP is that any voltage deflection that persists after averaging across many events is a time-locked response to the event being evaluated. For example, if every time a subject studies a word, there is a positive voltage deflection, averaging across all encoding events will reveal a positive voltage deflection. However if there is a positive deflection during some words and a negative deflection during others, the averaged voltages will be roughly 0.

Similarly, if there is an early positive deflection on some words and a late deflection for other words, the average voltage (ERP) may show a broad positive deflection with a very small magnitude or may not show any voltage deflection at all. An ERP will only show strong effects for voltage changes with the same directionality and timing across many trials.

It is important to note that a voltage deflection in the ERP does not necessarily reflect a true voltage deflection that occurs across every encoding event. If a few trials show very large positive voltage deflections but a majority of trials show a very small negative voltage deflection, the average may appear to show a positive deflection (though it will not be as large as it was for the few trials that had a true positive deflection).

ERPs are time-specific snippets extracted from the raw EEG signals. To do this, we use a custom Matlab function called `gete_ms`. This is a core function in the EEG toolbox that extracts voltages recorded from an electrode. The function uses the `eegfile` path in the `event.mat` structure to locate the `eegfile` that corresponds with an event and the `eegoffset` field in the `events.mat` file to know what time relative to the beginning of the recording to analyze. This function will only work if the `event.mat` structure points to a valid EEG file.

The inputs and outputs of the function is as such:

Inputs

- `channel`: the channel number you wish to analyze
- `events`: the events you want the voltages for. The output will give you a voltage for each time point for each event. The output matrix is described in more detail below.
- `DurationMS`: the length of time (in milliseconds) of EEG recording you would like to analyze
- `OffsetMS`: the time relative (in milliseconds) to the beginning of the event you would like to begin analyzing the EEG data
- `BufferMS`: the time (in milliseconds) before and after your desired time window to use as a buffer to prevent edge artifacts. The importance of this field is beyond the scope of this tutorial. You can use 500 ms for this.
- `filtfreq`: the frequencies you want to filter out or allow through. It is typical to use [58 62] to filter out 60 Hz line noise from electrical outlets in the U.S..
- `filttype`: the type of filter you want to use. 'stop' does not allow the specified frequencies through and is used for line-noise filtering. 'low' allows any frequencies below the 'filtfreq' through (low-pass). 'high' allows any frequencies above the specified frequency through (high-pass).
- `filtorder`: the significance of this input is beyond the scope of this tutorial. You can use an input of 1 for this field.
- `resampleFreq`: this entry is used to change the sampling rate of the EEG file. This is useful if trying to compare EEG recordings for patients with different recording rates. For instance, if a subject at the University of Pennsylvania had a 500 Hz recording and subjects at UT Southwestern have 1000 Hz recording, you can resample all of the data to 500 Hz.
- `RelativeMS`: this input allows you to subtract a baseline period from the voltage. This is useful to mitigate the effects of voltage drift across a session and to enable a more standardized comparison of activity from different electrodes and subjects.

Outputs

- `EEG`: this is a matrix of voltage values with dimensions (# of events by # of times).
- `resampleFreq`: this is an output of the sampling rate used.

Example

```
EEG = gete_ms(21,events,1800,-200,500,[58 62],'stop',1,200,[-200 0]);
```

This will return a matrix with as many rows as number of events in 'events' and 360 columns. The re-sample frequency is 200 Hz so each sample is 5 ms. The duration of the voltage computed is 1800 ms (360 samples). The voltage output begins 200 ms before the onset of the word and ends 1600 ms after the word onset. Therefore, samples 1 to 40 are before the word, and samples 41 to 360 are after the word. A baseline subtraction of the average voltage from -200 ms to 0 ms was performed.

3.4 Oscillatory power

As described earlier in the signal processing background section, time-frequency decomposition allows us to study in the changes in various frequency components of a signal over time and is one of the most basic analysis you can perform. This is done with another critical function in the toolbox called `getphasepow`. As the name suggests, the function returns the phase and power values of a signal across all the events trials in a time-frequency space using Wavelet Transform. Here's an example of using the function:

```
[phase,power] = getphasepow(21,events,1800, 0, 500,'filtfreq', [58 62], 'resampledtrate', 500,'freqs', freqs);
```

You'll notice these inputs are very similar to the ones use in `gete_ms`. This is because `gete_ms` is called within `getphasepow` to extract the EEG snippets for calculations of phase and power for the channel. However, `getphasepow` accepts a separate input 'freqs', which is a vector of the frequencies of interest we want to identify the phase and power values for, usually generated by the Matlab expression:

```
(2^(1/8)).^(8:n); (caret symbols are exponents in Matlab, doesn't show up well in Latex formatting...)
```

This will output a vector with logarithmically spaced values between values of power of two up to $2^{\hat{n}/8}$. A log spaced vector is used because we want more resolution in the lower frequencies as lower frequency bands have smaller ranges and high frequency bands have a higher ranges of values. As you get into more literature in the field, especially the rodent ones, you will start to see theories built upon changes in oscillatory power along with the trends and implications of increased or decreased activities in different frequency bands in different brain regions for different memory paradigms, and develop a better sense of what kind of results to expect in your analysis.

3.5 Connectivity Analysis

Placeholder

3.6 Statistical testing

Statistical testing is an integral part of any research analysis. Often times it is inadequate to present descriptive statistics, as it only describes the data you have without the ability for generalization. In such cases statistical comparisons allows us to determine the inferential statistic so that the result from our data set can be used to make an inference on the population level. In this section, you'll find some of the most common stats tests we use in this lab and its tutorial, along with resources to help address any questions you may have.

T-test: <https://www.youtube.com/watch?v=0Pd3dc1GcHc&t=141s>

Wilcoxin-Ranksum test: <https://www.youtube.com/watch?v=nRAAAp1Bgnw&t=72s>

Analysis of Variance (ANOVA): https://www.youtube.com/watch?v=-yQb_ZJnFXw&t=4s

One of the most important concepts to understand in statistics is the correct interpretation of a p-value from a statistical comparison. What the p-value tells you is that, assuming the null hypothesis is true, what is

the probability you would observe differences as extreme or more extreme between your two distributions. Make sure you nail down this concept. The following links also provide great additional resources:

<https://www.graphpad.com/guides/prism/7/statistics/>

<https://www.youtube.com/user/statisticsfun>

Cohen Textbook

4 Exploring Folders on your Apple Computer Using Terminal

It is often much more efficient to work with folders and files on your computer from the command line (Terminal on an Apple Computer). Depending on your operating system, the command line interface on your computer may be different but the functionality should be preserved across all systems.

Terminal is effectively the same as 'Finder' on an Apple Computer or 'My Computer' on a PC but using it requires becoming acquainted with some common commands. While it may seem to add unnecessary complexity at first, you will soon find that navigating through your computer from the command line can help you accomplish seemingly tedious tasks much more efficiently.

As you first begin working in Terminal, it may be helpful to have a file browser window (such as Finder on a Mac) open. As you use terminal commands to navigate from your computer, verify that the result of navigating through your computer with the file browser is the same as navigating through the command line.

4.1 Common Command Line Functions

- **pwd:** lists current directory
- **ls:** lists contents of current directory
- **ls -l:** lists the read-write-execute permissions for each file in the current directory, the owner and group specifications for a file, the size of the file, the date it was created, and the file name.
- **chgrp:**
- **chmod:**
- **cd [path]:** change directory to the path specified by [path]. For example:
 - **cd /Users/jdoe/Documents/** will move to the Documents folder for the user account 'jdoe'. Typing 'ls' after this command will show the contents of that folder.
 - **cd /Users/jdoe/Documents/exampleFolder** will move to the folder 'exampleFolder' if it exists. Again, typing ls will show the contents of this folder.
 - **cd exampleFolder** while already in Documents will yield the same result as typing the full path for exampleFolder. Typing the forward slash before the path allows you to change to a directory from anywhere on the computer (as long as there is only one folder with the name that follows the first forward slash). Excluding the forward slash before the path will change directories to files within the current folder.
 - **cd ../** will move out of the current folder into the folder where it is contained.
 - **cd ~** will move to the root or home directory for the current user.
- **cp [source path] [destination path]:** including the '-r' specification indicates a recursive copy which allows you to copy a folder and all of its contents rather than just a single file.
- **cp -r [source path] [destination path]:**

- **scp [source path] [destination path] and scp -r [source path] [destination path]:** These commands are the same as cp but they allow you to copy a file from a server. For example:
 - **scp -r jdoe@rhino2.psych.upenn.edu:/data/eeg/R1053T/behavioral/ /Users/jdoe/Documents/R1053T/behavioral**
will copy the folder on the rhino server at the specified path to the specified local path.
- **mv [source path] [destination path]:** moves the source file/folder to the destination.
- **mkdir [path]:** creates an empty folder in the specified path. Nested folders will be created if intermediate paths do not yet exist
- **rm [path] and rm -r [path]:** deletes the file or folder respectively located at the specified path.
- **ssh -XY [username]@[serverURL]:** prompts you to enter a password to connect to the specified server with the specified username.
 - **ssh -XY jdoe@rhino2.psych.upenn.edu** will prompt you to enter the password to connect to rhino2 under the username jdoe.

5 Data Collection

stim, nonstim, setup,

5.1 Verbal Tasks

FR1, pyfr_stim

5.2 Spatial Tasks

TH1, train

5.3 Stim-Only Tasks

CCEP, cortical mapping

6 Data Processing

Before patient data can be analyzed, its behavioral component and EEG component must be organized to the proper format. The processing pipeline is as followed:

Memory Testing -> Obtain EEG recording -> Split EEG Recording -> Make Event Structure -> Align Events

Below, I will go over the processing details of the most common memory task we analyze in this lab: FR1.

6.1 Memory Testing

When you're about to start a test with the patient, stop and start the clinical EEG, and do the same after it's over so the session recording will be isolated. The most important thing to keep in mind about memory testing is that if you don't see sync pulses on the clinical EEG while the test is going on, that session will be useless. These sync pulses are injected into channel DC09 by the testing laptop as a marker to align the session events from Unix timestamps to the EEG time. If there are no sync pulses recorded, the session will have to be thrown out.

After testing, the audios file should be annotated immediately by the person who administered the test using PennTotalRecall. The program will generate .ann files for every list which will be an integral part of event making.

6.2 Obtain EEG recording

Along with annotating the audio files, the EEG recording for that session should also be saved out from the clinical system immediately. For macro recordings the files of interest are the .21E and .EEG; for micro is .ns3 and .ns6. These file should be organized into the raw folder inside a subject's directory with the format: 'MM_DD_YYYY [task name]'

6.3 Split EEG Recording

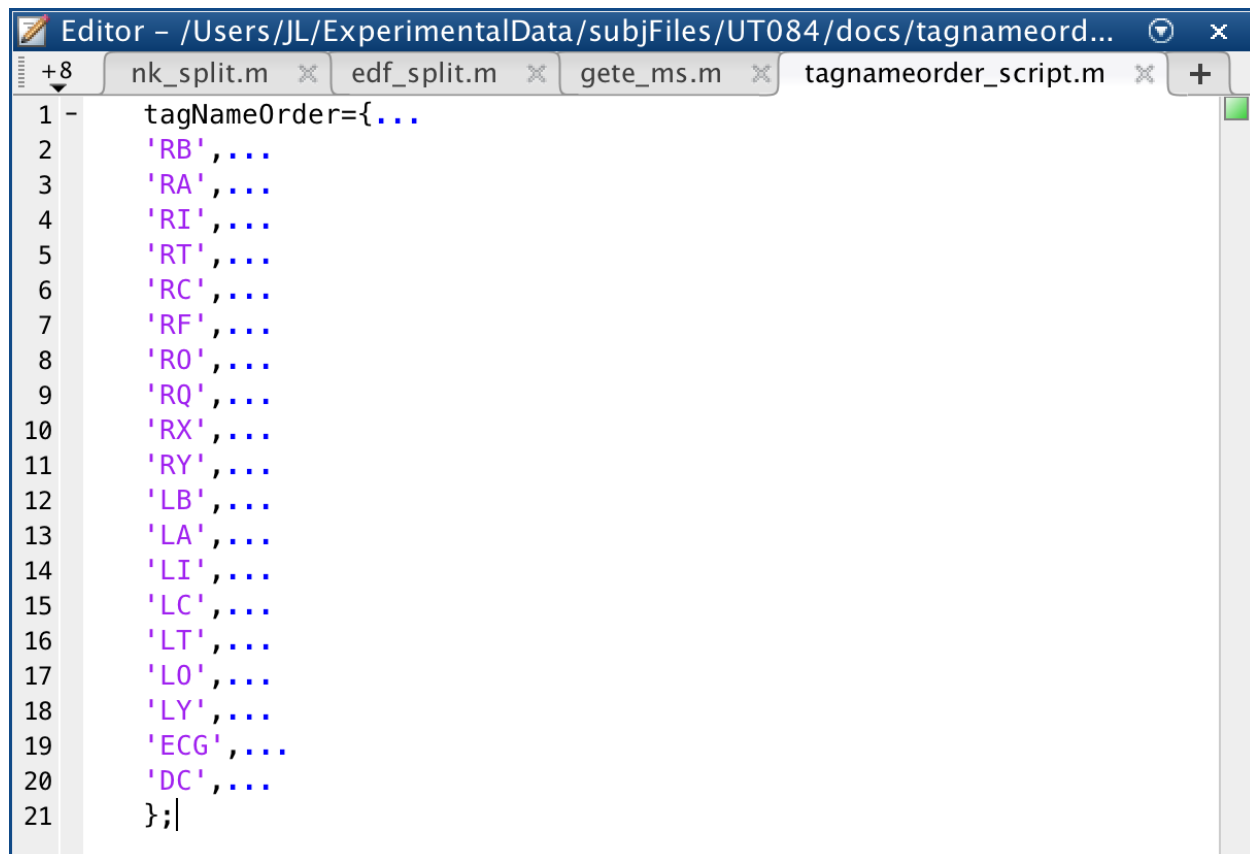
This step separates the raw EEG recording into its individual channel components and performs a re-referencing step. The process is done with a wrapper script call `nk_split_wrapper.m`. Before running the script, you'll need to create a new `tagnameorder_script.m` file inside of the docs folder in the subject directory (you can copy it from a previous subject). The script should follow the format as shown (Figure 2). The electrode label names will need to be updated to reflect that of the current subject. The new labels can be found in the .21E file (open with TextEdit). You should see a list of channels that were active during the recording in that file, and you'll take those electrode labels and rename the old one on the `tagnameorder_script.m` in order with no repeats, excluding anything labels that doesn't start with L or R (EKG and DC should always be included last). Once that is done, save the file, and restart Matlab (Matlab sometimes doesn't recognize the tag name file until a reboot, so perform this step just in case).

To split the EEG, open up `nk_split_wrapper.m` and modify the subject name and the raw folder name of the session and click run. The command prompt should say that a jacksheet and the `noreref` files for each electrode was created and pause in debug mode. Check the jacksheet in docs and make sure there are only 130 channels and all the labels seem appropriate. If that's the case type in 'dbcont' to continue the script. The command prompt should then display a list of electrodes will that be included or excluded in the re-reference step; all electrodes should be included in the reference except for EKG and DC. Continue debugging if you see that and the re-referenced electrode files should populate in the `reref` folder in the subject directory.

It should be mentioned that although the macro recordings from both Parkland and Zale are saved as .EEG files, they are slightly different. The Zale EEG is in something called an extended format which allows for recording that includes more than 128 channels. Instead of the using `nk_split_wrapper`, you'll need to use `EEG_splitter` to split Zale EEG. `EEG_splitter` is a GUI and just follow the instructions that it has to complete the process.

6.4 Making Event Structures

To make events for FR1 specifically, use the function `RAM_FR_CreateAllEvents.m`. The inputs are the subject name, session folder directory, and session folder number. Here's an example:



```
1 - tagNameOrder={...
2     'RB',...
3     'RA',...
4     'RI',...
5     'RT',...
6     'RC',...
7     'RF',...
8     'RO',...
9     'RQ',...
10    'RX',...
11    'RY',...
12    'LB',...
13    'LA',...
14    'LI',...
15    'LC',...
16    'LT',...
17    'LO',...
18    'LY',...
19    'ECG',...
20    'DC',...
21 };|
```

Figure 2: tagnameorder_script.m formatting.

```
RAM_FR.CreateAllEvents('UT084','/Users/JL/ExperimentalData/subjFiles/UT084/behavioral/FR1,0)
```

If everything runs smoothly, you should see that the function generate a file for both the verbal and math events.

6.5 Align Events

Event alignment involves determining the EEG recording time offset from the unix timestamps of the sync pulses generated during the memory task for all trials. The process is done using a GUI called align-Tool.

The first step involves marking a group of sync pulses on the DC channel (Figure 3). Using the 'load file' option on the GUI to upload the signal from DC09 (usually channel 129) in the subject's noreref folder, then zoom in on the peaks of the spikes. Select the mark option and highlight a section of the EEG you want to mark. Make sure all the consecutive pulses within the time window is marked, then click on 'Save Pulses' to save it back to the noreref directory.

Next you want to run the fixEEGLog

6.6 Processing Microwire Data

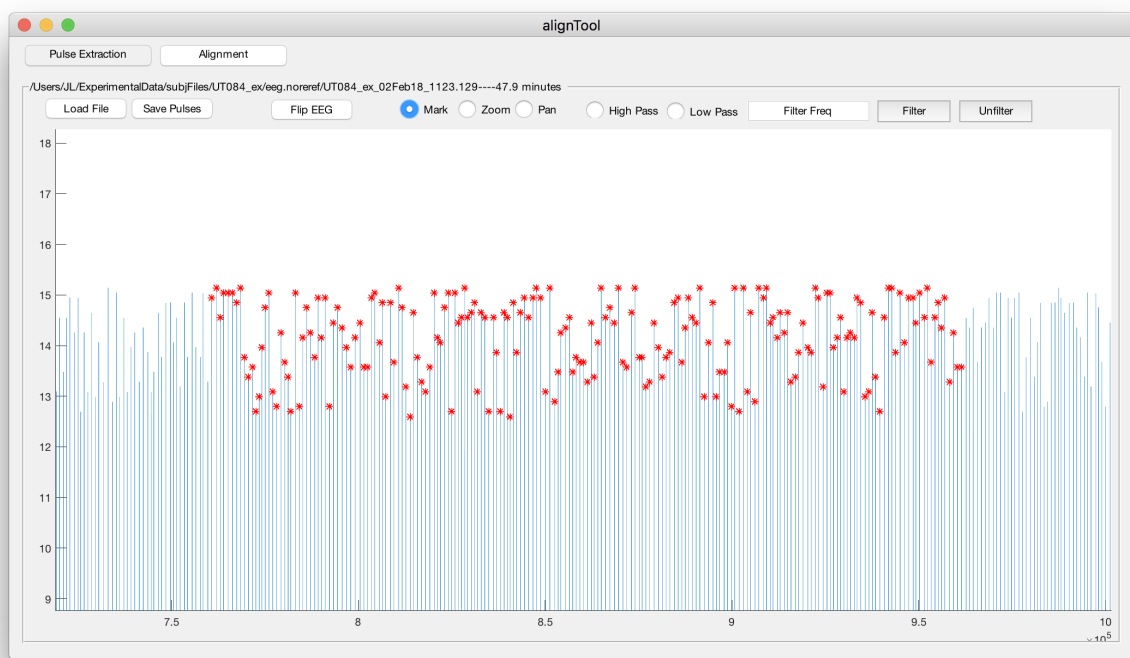


Figure 3: Sync pulse marking example