EC2 Instance (13.212.186.33)	
SSH from WSL to EC2	`ssh -i MyKeyPair.pem ec2-user@13.212.186.33`
SCP (Copy) from WSL to EC2	`scp -rp -i ./MyKeyPair.pem ~/.aws ~/cluster-config.yaml ./MyKeyPair.pem ~/EE3801/PyHipp/update_snapshot.sh ec2-user@13.212.186.33:~/`
Cluster Startup Routine	

Fix Node	`sudo apt-get remove nodejs`
	`nvm install 16.15.1`
	`nvm --version`
	`node --version`

Create Cluster	`pcluster create-cluster -c ~/cluster-config.yaml -n MyCluster01`
Describe Cluster	`pcluster describe-cluster -n MyCluster01`
SSH to Cluster	`pcluster ssh -i ~/MyKeyPair.pem -n MyCluster01`
Access Data Drive	`cd /data`

Initialise Miniconda	`miniconda3/bin/conda init`
	`source ~/.bashrc`
	`conda activate env1`

Copy AWS Access Perms	`sudo cp -r /data/aws ~/.aws`

AWS Perms Setup (first-time)	`AWS Public Access Key: AKIAQQ2QXKEGF7RXSZ74`
	`AWS Secret Access Key: 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`
	
	`Setup:`
	`aws configure`
	
	`Settings: (Apart from these and access keys, leave blank)`
	`Default region name [None]: ap-southeast-1`
	`Default output format [None]: json`

EC2 Cluster Administration	
Check active clusters	`pcluster list-clusters`

Update snapshot manually	`update_snapshot.sh data 2 MyCluster01`
	`OR`
	`bash update_snapshot.sh data 2 MyCluster01`

Delete Cluster	`pcluster delete-cluster -n MyCluster01`

Stop compute node and update cluster based on new config file	`pcluster update-compute-fleet --status STOP_REQUESTED --region ap-southeast-1 --cluster-name MyCluster01`
	
	`pcluster update-cluster --cluster-configuration ~/cluster-config.yaml --cluster-name MyCluster01`

Login to compute node	`srun --pty /bin/bash`
SCP from EC2 to Cluster	`scp -i ~/MyKeyPair.pem ~/MyKeyPair.pem ec2-user@xx.xxx.xxx.xxx:/data`
SCP from Cluster to EC2 / WSL	`scp -i ~/MyKeyPair.pem -p "ec2-user@54.251.188.19:/data/picasso/unity*.hkl" picasso/`
AWS / Jobs-Related	
Publish to AWS SNS (run in command line or shellscript)	`aws sns publish --topic-arn arn:aws:sns:ap-southeast-1:036139651340:awsnotify --message “ClusterTest”`
Cancel Job	`scancel {2..7}`

Submit job with dependancy	`sbatch --dependency=afterok:12:13:14:15:16 /data/src/PyHipp/consol_jobs.sh`

Copying Snapshot

Copy --> Rename description (e.g. to *'data'*) --> ensure *Encrypt this Snapshot* is NOT selected --> Copy

Fresh-Install PyHipp	

If NO PyHipp repo present in cluster
`cd /data/src`
`git clone https://github.com/hsienrong/PyHipp`
 	
If PyHipp repo already present
`git reset --hard HEAD`
`git pull`
	Installing PyHipp using pip
	cd PyHipp
	pip install -r requirements.txt
	pip install -e .
	cd /data

Data Type	Input	Function	Output
Ripple Parallel Port Data	180702_Block1.nev	RPLParallel	rplparallel_xxxx.hkl
Unity Data	session_1_272018105911.txt rplparallel_xxx.hkl	Unity	unity_xxxx.hkl
Eyelink Data	180702.edf, P7_2.edf rplparallel_xxxx.hkl	EDFSplit	eyelink_xxxx.hkl
Aligned Data	rplparallel_xxxx.hkl unity_xxxx.hkl eyelink_xxxx.hkl	aligning_objects	unity_xxxx.hkl eyelink_xxxx.hkl
Raycast Data	unity_xxxx.hkl eyelink_xxxx.hkl	raycast	bindata.hdf slist.txt ...
Ripple Neural Data	181105_Block1.ns5	RPLSplit	rplraw_xxxx.hkl
Low-Pass Neural Data	rplraw_xxxx.hkl	RPLLFP	rpllfp_xxxx.hkl
High-Pass Neural Data	rplraw_xxxx.hkl	RPLHighPass	rplhighpass_xxxx.hkl
Spiketrain Data	rplhighpass_xxxx.hkl (from both navigation and fixation sessions)	mountain_batch	spiketrain_xxx.hkl

Data Type	Directory	Function
Ripple Parallel Port Data	session01 sessioneye	pyh.RPLParallel(saveLevel=1)
Unity Data	session01	pyh.Unity(saveLevel=1)
Eyelink Data	20181105	pyh.EDFSplit()
Aligned Data	session01	pyh.aligning_objects()
Raycast Data	session01	pyh.raycast(1)
Ripple Neural Data	session01 sessioneye	pyh.RPLSplit(channel=[9])
Low-Pass Neural Data	session01/array01/ channel009 sessioneye/array01/ channel009	pyh.RPLLFP(saveLevel=1)
High-Pass Neural Data	session01/array01/ channel009 sessioneye/array01/ channel009	pyh.RPLHighPass(saveLevel=1)
Spiketrain Data	session01/array01/ channel009	mountain_batch.mountain_batch()

a. RPLParallel (for both session01 and sessioneye)
b. RPLSplit to create a RPLRaw object for each of the 110 channels (for both session01 and sessioneye)
c. RPLLFP (which needs the RPLRaw object) for each of the 110 channels (for both session01 and sessioneye)
d. RPLHighPass (which needs the RPLRaw object) for each of the 110 channels (for both session01 and sessioneye)
e. Spike sorting (which needs the RPLHighPass objects for both session01 and sessioneye) for each of the 110 channels
f. Unity (needs RPLParallel object)
g. EDFSplit to create Eyelink objects (needs RPLParallel, and Unity if available) (for both session01 and sessioneye)
h. Aligning_objects (needs RPLParallel, Unity, and Eyelink objects)
i. Raycasting (needs Unity and Eyelink objects)

DPT.objects.processDirs: Executes commands in all necessary subdirectories		

Example:	`DPT.objects.processDirs(dirs=None, objtype=pyh.RPLParallel, saveLevel=1); \`
	`DPT.objects.processDirs(dirs=None, objtype=pyh.Unity, saveLevel=1); \`
	
	`DPT.objects.processDirs(dirs=['sessioneye/array01','session01/array01'], cmd='import PyHipp as pyh; import DataProcessingTools as DPT; DPT.objects.processDirs(None, pyh.RPLLFP, saveLevel=1); DPT.objects.processDirs(None, pyh.RPLHighPass, saveLevel=1);');`	

RPLParallel: Process Ripple .nev files containing signals sent by Unity via Parallel Port		
Input .nev file *(Both session01 and sessioneye)*	**Output** RPLParallel Object *(rplparallel_d41d.hkl)*	**Run in** session01, sessioneye
RPLUnity: Process RPLParallel object to generate unity object		
Input RPLParallel object	**Output** Unity Object *(unity_71bf.hkl)*	**Run in** session01, sessioneye
EDFSplit: Process RPLParallel object to generate unity object		
Input RPLParallel (and Unity if available)	**Output** Eyelink Object *(eyelink_24d5.hkl)*	**Run in** 20181105 (???)
Aligning Objects: Aligns Ripple, Unity and Eyelink Data		
Input RPLParallel , Unity, Eyelink	**Output** (In-place)	**Run in** session01
Raycasting		
Input Unity, Eyelink	**Output** (Log-file: VirtualMazeBatchLog.txt)	**Run in** session01

Example:
Myrplparallel-slurm.sh
	`#!/bin/bash`
	
	`# Submit this script with: sbatch <this-filename>`	

Generate RPLParallel objects, THEN generate Unity objects, and perform aligning_objects(),raycast()

Execute in:
Preferably 20181105, etc

Note that processDirs will auto-scope so it does both session01 and sessioneye

```
#SBATCH --time=24:00:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH -J "rplpl"   # job name

## /SBATCH -p general # partition (queue)
#SBATCH -o rplpl-slurm.%N.%j.out # STDOUT
#SBATCH -e rplpl-slurm.%N.%j.err # STDERR

# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE
python -u -c "import PyHipp as pyh; \
import DataProcessingTools as DPT; \
import os; \
import time; \
t0 = time.time(); \
print(time.localtime()); \
DPT.objects.processDirs(dirs=None, objtype=pyh.RPLParallel, saveLevel=1); \
DPT.objects.processDirs(dirs=None, objtype=pyh.Unity, saveLevel=1); \
pyh.EDFSplit(); \
os.chdir('session01'); \
pyh.aligning_objects(); \
pyh.raycast(1); \
print(time.localtime()); \
print(time.time()-t0);"

aws sns publish --topic-arn arn:aws:sns:ap-southeast-1:036139651340:awsnotify --
message "RPLParallelJobDone"
```

RPLSplit: Generate RPLRaw objects from ns5 files

Input	**Output**	**Run in**
.ns5 files (Pre-provided in session01)	RPLRaw Object (rplraw_d41d.hkl)	session01 and sessioneye

Example:
(MODIFIED)
myrs2-slurm.sh

What does it do?
Performs RPLSplit ONLY on channels in array02

<u>Execute in</u>
20181105

<u>Expected Outputs</u>
array02 directory containing channel033 to channel064. All of the channels should have a rplraw.d41d file

```
#!/bin/bash

# Submit this script with: sbatch <this-filename>

#SBATCH --time=24:00:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH --cpus-per-task=5   # number of CPUs for this task

#SBATCH -J "rs2m"   # job name

## /SBATCH -p general # partition (queue)
#SBATCH -o rs2m-slurm.%N.%j.out # STDOUT
#SBATCH -e rs2m-slurm.%N.%j.err # STDERR

# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE
python -u -c "import PyHipp as pyh; \
        import DataProcessingTools as DPT; \
        import os; \
        import time; \
        t0 = time.time(); \
        print(time.localtime()); \
        DPT.objects.processDirs(dirs=None, objtype=pyh.RPLSplit,
channel=[*range(33,65)]); \
        print(time.localtime()); \
        print(time.time()-t0);"

aws sns publish --topic-arn arn:aws:sns:ap-southeast-1:036139651340:awsnotify --
message "RS2MJobDone"
```

Example:
(Unmodified)
myrs2-slurm.sh

What does it do?
Performs RPLSplit ONLY on channels in array02. Subsequently, executes RPLLFP, RPLHighPass and Spike Sorting.

```
#!/bin/bash

# Submit this script with: sbatch <this-filename>

#SBATCH --time=24:00:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH --cpus-per-task=5   # number of CPUs for this task

#SBATCH -J "rs2"   # job name

## /SBATCH -p general # partition (queue)
#SBATCH -o rs2-slurm.%N.%j.out # STDOUT
#SBATCH -e rs2-slurm.%N.%j.err # STDERR
```

Note that these are NOT being executed as their own tasks so it will take longer **<u>Execute in</u>** 20181105	# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE python -u -c "import PyHipp as pyh; \ import DataProcessingTools as DPT; \ import os; \ import time; \ t0 = time.time(); \ print(time.localtime()); \ DPT.objects.processDirs(dirs=None, objtype=pyh.RPLSplit, channel=[*range(33,65)]); \ DPT.objects.processDirs(dirs=['sessioneye/array02','session01/array02'], cmd='import PyHipp as pyh; import DataProcessingTools as DPT; DPT.objects.processDirs(None, pyh.RPLLFP, saveLevel=1); DPT.objects.processDirs(None, pyh.RPLHighPass, saveLevel=1);'); \ os.chdir('session01/array02'); \ DPT.objects.processDirs(level='channel', cmd='import PyHipp as pyh; from PyHipp import mountain_batch; mountain_batch.mountain_batch(); from PyHipp import export_mountain_cells; export_mountain_cells.export_mountain_cells();'); \ print(time.localtime()); \ print(time.time()-t0);" aws sns publish --topic-arn arn:aws:sns:ap-southeast-1:036139651340:awsnotify --message "RS2JobDone"
Example: Modified python code for RPLSplit. Uses extra argument SkipHPC=False to know to search for slurm scripts to auto-run rpllfp-slurm.sh, rplhighpass-sort-slurm.sh.	python -u -c "import PyHipp as pyh; \ import DataProcessingTools as DPT; \ import time; \ import os; \ t0 = time.time(); \ print(time.localtime()); \ DPT.objects.processDirs(dirs=None, objtype=pyh.RPLSplit, channel=[*range(1,33)], SkipHPC=False, HPCScriptsDir = '/data/src/PyHipp/', SkipLFP=False, SkipHighPass=False, SkipSort=False); \ print(time.localtime()); \ print(time.time()-t0);"

RPLLFP: Generate Low-Pass Filtered Signals		
Input RPLRaw Object (rplraw_d41d.hkl)	**Output** RPLLFP Object (rpllfp_6eca.hkl)	**Run in** array01/channel009 *(Both session01 and sessioneye)*
Example: rpllfp-slurm.sh	#!/bin/bash # Submit this script with: sbatch <this-filename> #SBATCH --time=1:00:00 # walltime #SBATCH --ntasks=1 # number of processor cores (i.e. tasks) #SBATCH --nodes=1 # number of nodes #SBATCH -J "rpllfp" # job name ## /SBATCH -p general # partition (queue) #SBATCH -o rpllfp-slurm.%N.%j.out # STDOUT #SBATCH -e rpllfp-slurm.%N.%j.err # STDERR # LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE python -u -c "import PyHipp as pyh; \ import time; \ pyh.RPLLFP(saveLevel = 1); \ print(time.localtime());"	
Example: Shell script executed from within arrray02 to iterate through 2nd set of channels	#!/bin/bash cwd=`pwd` for i in `find . -name "channel*" | sort` do echo $i cd $i sbatch /data/src/PyHipp/rpllfpfs-slurm.sh sbatch /data/src/PyHipp/rplhpfs-slurm.sh cd $cwd done	

RPLHighPass: Generate High-Pass Filtered Signals		
Input RPLRaw Object (rplraw_d41d.hkl)	**Output** RplHighPass Object (rplhighpass_b59f.hkl)	**Run in** array01/channel009 *(Both session01 and sessioneye)*

Example:
Rplhighpass-sort-slurm.sh

This script runs RPLHighPass on RPLRaw objects, and subsequently does spike sorting.

To generate RPLHighPass without spike sorting, remove all the conda-related stuff and mountain_batch related stuff.

```
#!/bin/bash

# Submit this script with: sbatch <this-filename>

#SBATCH --time=24:00:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH -J "rplhps"   # job name

## /SBATCH -p general # partition (queue)
#SBATCH -o rplhps-slurm.%N.%j.out # STDOUT
#SBATCH -e rplhps-slurm.%N.%j.err # STDERR

# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE
/data/miniconda3/bin/conda init
source ~/.bashrc
envarg=`/data/src/PyHipp/envlist.py`
conda activate $envarg

python -u -c "import PyHipp as pyh; \
        import time; \
        pyh.RPLHighPass(saveLevel = 1); \
        from PyHipp import mountain_batch; \
        mountain_batch.mountain_batch(); \
        from PyHipp import export_mountain_cells; \
        export_mountain_cells.export_mountain_cells(); \
        print(time.localtime());"

conda deactivate
/data/src/PyHipp/envlist.py $envarg
```

Example:
Shell script executed from within arrray02 to iterate through 2nd set of channels

```
#!/bin/bash
cwd=`pwd`
for i in `find . -name "channel*" | sort`
do
        echo $i
        cd $i
        sbatch /data/src/PyHipp/rpllfpfs-slurm.sh
        sbatch /data/src/PyHipp/rplhpfs-slurm.sh
        cd $cwd
done
```

FreqSpectrum: Generate FreqSpectrum		
Input RPLLFP Object (rpllfp_6eca.hkl) OR RplHighPass Object (rplhighpass_b59f.hkl)	**Output** FreqSpectrum objects	**Run in** array01/channel009 *(Both session01 and sessioneye)*

Example:
Freq-slurm.sh

First call creates FreqSpectrum object from RPLLFP object

Second call creates FreqSpectrum object from RplHighPass object (due to specified args)

```
#!/bin/bash

# Submit this script with: sbatch <this-filename>

#SBATCH --time=1:00:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH -J "freqslurm"   # job name

## /SBATCH -p general # partition (queue)
#SBATCH -o freqslurm-slurm.%N.%j.out # STDOUT
#SBATCH -e freqslurm-slurm.%N.%j.err # STDERR

# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE
python -u -c "import PyHipp as pyh; \
        import time; \
        pyh.FreqSpectrum(saveLevel=1); \
        pyh.FreqSpectrum(loadHighPass=True, pointsPerWindow=3000, saveLevel=1); \
        print(time.localtime());"
```

Comments	Run AFTER RPLLFP and/or RplHighPass, INSIDE channel009 (etc) folder

Spike Sorting: Generate FreqSpectrum		
Input RplHighPass Object (rplhighpass_b59f.hkl)	**Output** firings.mda in each channel	**Run in** array01/channel009 *(session01)*

Example:
rs2-slurm.sh

After running RPLHighPass, change into array02 directory and run spike sorting within each channel using processDirs

```
#!/bin/bash

# Submit this script with: sbatch <this-filename>

#SBATCH --time=24:00:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH --cpus-per-task=5   # number of CPUs for this task

#SBATCH -J "rs2"   # job name

## /SBATCH -p general # partition (queue)
#SBATCH -o rs2-slurm.%N.%j.out # STDOUT
#SBATCH -e rs2-slurm.%N.%j.err # STDERR

# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE
python -u -c "import PyHipp as pyh; \
        import DataProcessingTools as DPT; \
        import os; \
        import time; \
        t0 = time.time(); \
        print(time.localtime()); \
        DPT.objects.processDirs(dirs=None, objtype=pyh.RPLSplit, channel=[*range(33,65)]); \
        DPT.objects.processDirs(dirs=['sessioneye/array02','session01/array02'], cmd='import
PyHipp as pyh; import DataProcessingTools as DPT; DPT.objects.processDirs(None, pyh.RPLLFP,
saveLevel=1); DPT.objects.processDirs(None, pyh.RPLHighPass, saveLevel=1);'); \
        os.chdir('session01/array02'); \
        DPT.objects.processDirs(level='channel', cmd='import PyHipp as pyh; from PyHipp
import mountain_batch; mountain_batch.mountain_batch(); from PyHipp import
export_mountain_cells; export_mountain_cells.export_mountain_cells();'); \
        print(time.localtime()); \
        print(time.time()-t0);"

aws sns publish --topic-arn arn:aws:sns:ap-southeast-1:036139651340:awsnotify --message
"RS2JobDone"
```

Example
sort-slurm.sh

Run within each array02/channelxxx directory, etc.

Can get rid of all the conda stuff

```
#!/bin/bash

# Submit this script with: sbatch <this-filename>

#SBATCH --time=24:00:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH -J "sort"   # job name

## /SBATCH -p general # partition (queue)
#SBATCH -o sort-slurm.%N.%j.out # STDOUT
#SBATCH -e sort-slurm.%N.%j.err # STDERR

# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE
/data/miniconda3/bin/conda init
source ~/.bashrc
envarg=`/data/src/PyHipp/envlist.py`
conda activate $envarg

python -u -c "import PyHipp as pyh; import time; from PyHipp import mountain_batch;
mountain_batch.mountain_batch(); from PyHipp import export_mountain_cells;
export_mountain_cells.export_mountain_cells(); print(time.localtime());"

conda deactivate
/data/src/PyHipp/envlist.py $envarg
```

Appending of Unity Objects for Plotting

Input Unity objects	**Output** *uyall object (unity*.hkl)*	**Run in** *Picasso* (or whichever directory is fit: utilizes processDirs to auto-get all Unity objects)

Example

Execute in: /data/picasso (or whichever is fit)

In ipython:

```
In [ ]: import PyHipp as pyh
In [ ]: import DataProcessingTools as DPT
In [ ]: uyall = DPT.objects.processDirs(dirs=None, objtype=pyh.Unity)
In [ ]: uyall.save()
```

Exporting

In WSL

```
scp -i ~/MyKeyPair.pem -p "ec2-user@54.251.188.19:/data/picasso/unity*.hkl" picasso/
```

Appending of Spike Sorting Objects into Waveform Object for Plotting

Input Low- and high-frequency objects	**Output** *wfall object (waveform*.hkl)*	**Run in** *Picasso* (or whichever directory is fit: utilizes processDirs to auto-get all Unity objects)

Example

Execute in: /data/picasso (or whichever is fit)

In ipython:

```
In [ ]: wfall = DPT.objects.processDirs(dirs=None, exclude=['*eye*', '*mountains*'],
objtype=pyh.Waveform, saveLevel=1)
In [ ]: wfall.save()
```

Exporting

In WSL

```
scp -i ~/MyKeyPair.pem -p "ec2-user@54.251.188.19:/data/picasso/waveform*.hkl" picasso/
```

Cumulative objects containing low and high frequency spectrums of channels (fsall)

Input Spike Sorting output	**Output** *freqspectrum_9c80.hkl* *freqspectrum_660e.hkl*	**Run in** *See what scope we want*

Example
fsall-slurm.sh

Execute in: 20181101 or session01 or array01 (whichever is relevant)

```
#!/bin/bash

# Submit this script with: sbatch <this-filename>

#SBATCH --time=1:00:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH -J "fsall"   # job name

## /SBATCH -p general # partition (queue)
#SBATCH -o fsall-slurm.%N.%j.out # STDOUT
#SBATCH -e fsall-slurm.%N.%j.err # STDERR

# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE
python -u -c "import PyHipp as pyh; \
        import DataProcessingTools as DPT; \
        lfall = DPT.objects.processDirs(dirs=None, exclude=['*eye*', '*mountains*'],
objtype=pyh.FreqSpectrum, saveLevel=1); \
        lfall.save(); \
        hfall = DPT.objects.processDirs(dirs=None, exclude=['*eye*', '*mountains*'],
objtype=pyh.FreqSpectrum, loadHighPass=True, pointsPerWindow=3000, saveLevel=1); \
        hfall.save();"

aws sns publish --topic-arn arn:aws:sns:ap-southeast-1:036139651340:awsnotify --message
"FSJobDone"
```

Remarks

Should only be run AFTER all freq-slurm.sh (lowpass and highpass) jobs are done

Example script to set up dependencies on all other jobs: (consol_fsjobs.sh)

	#!/bin/sh
	
	temp1=($(squeue))
	
	cmd1="sbatch --dependency=afterok:"
	
	counter1=0
	for i in "${temp1[@]}"; do
	  if [["$i" == "queue1"]]; then
	  id1=${temp1[$counter1-1]}
	  cmd1="${cmd1}${id1}:"
	  fi
	  counter1=$((counter1+1))
	done
	
	cmd1=${cmd1::-1}
	cmd1="${cmd1} /data/src/PyHipp/fsall-slurm.sh"
	
	echo $cmd1
	eval $cmd1

Plotting (Spyder Side Prompt) OPEN IN ANACONDA PROMPT!!!!!!	
Load Unity Objects (Paths) Execute in side-prompt --> In[ ]: **Look at data by session** Right-click → PlotTypes → Routes **Look at data across all 45 sessions** Right-click → PlotTypes → Proportion of Trials	(IN ANACONDA PROMPT) conda activate aws spyder (ALTERNATIVELY): ipython and cd to directory containing folders (IN SPYDER SIDE PROMPT) import PyHipp as pyh import PanGUI cd ~/Documents/picasso uy = pyh.Unity(loadFrom='unity_71bf.hkl') puy = PanGUI.create_window(uy)
Load Waveform Object **Look at multiple channels**: Right click → PlotTypes → Array (Arranged by electrode positions!)	wf = pyh.Waveform(loadFrom='waveform_ed79.hkl') pwf = PanGUI.create_window(wf)
Load Low Freq Spectrum Object **Arrange by electrode:** Right click → PlotTypes → Array	import PyHipp as pyh import PanGUI lf = pyh.FreqSpectrum(loadFrom='freqspectrum_9c80.hkl') plf = PanGUI.create_window(lf)
Load High Freq Spectrum Object **Arrange by electrode:** Right click → PlotTypes → Array	import PyHipp as pyh import PanGUI hf = pyh.FreqSpectrum(loadFrom='freqspectrum_660e.hkl') phf = PanGUI.create_window(hf)

Redo for missing things	

	#!/bin/bash
	
	# Find all channels
	find . -name "channel*" | grep -v -e eye -e mountain | sort > chs_hkl.txt
	find . -name "rplhighpass*hkl" | grep -v -e eye | sort | cut -d "/" -f 1-4 > hps.txt
	
	echo "--- Channels Incomplete: ---"
	comm -23 chs_hkl.txt hps.txt
	echo "------"
	cwd=$(pwd)
	echo "CWD: $cwd"
	echo "------"
	
	for i in $(comm -23 chs_hkl.txt hps.txt)
	do
	  cd $i
	  echo "NOW IN: $i"
	  sbatch /data/src/PyHipp/rplhighpass-sort-slurm.sh
	  cd $cwd
	done

	#!/bin/bash
	
	# Find all channels
	find . -name "channel*" | grep -v -e eye -e session01 | sort | cut -d "/" -f 3 >
	chs_mountains.txt
	find . -name "firings.mda" | grep -v -e eye | sort | cut -d "/" -f 3 > mda_list.txt
	
	echo "--- Channels Incomplete: ---"
	comm -23 chs_mountains.txt mda_list.txt
	echo "------"
	cwd=$(pwd)
	echo "CWD: $cwd"
	echo "------"
	
	for i in $(comm -23 chs_mountains.txt mda_list.txt)
	do
	  echo "CD: $(find . -name $i | grep -v -e eye -e mountains)"
	  cd $(find . -name $i | grep -v -e eye -e mountains)
	  sbatch /data/src/PyHipp/rplhighpass-sort-slurm.sh
	  cd $cwd
	done

Github Token: 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

Force Git Pull (Overwrite with GitHub version)

```
git reset --hard HEAD
git pull
```

Cloning from GitHub

```
cd /data/src
git clone https://github.com/hsienrong/PyHipp
```

OR:

Download files from upstream repo (the one from Prof) without integrating

Set Upstream: `git remote add upstream https://github.com/shihchengyen/PyHipp.git`
Fetch Upstream: `git fetch upstream`

Checkout main branch into local
`git checkout main`

Merge in upstream files
`git merge upstream/main`

Resolving conflicting files

```
git status
vim xxxxxxx.xxx (whatever conflicting file), then resolve issues
git add xxxxxx.xxx
```

Adding in local files (new files)

```
git add my*.sh (etc etc)
git commit
git push
```

Expected number of files (run checkfiles2.sh in date directory)

20181101, 20181102, 20181105 → 665 hkl, 110 mda files

Nano Commands

Delete Line: Ctrl + K

Undo: Alt + U