

# Neuropixels Toolkit Live Script

## Build and execute a processing session with pipelines

### Package importing

```
import npxtoolkit.session.Session
import npxtoolkit.pipeline.Pipeline
import npxtoolkit.config.Config
import npxtoolkit.config.PipelineConfig
```

### Setup Logger

```
import npxtoolkit.internal.thirdparty.logging.log4m
logger = log4m.getLogger("npx.log");
logger.clearLog();
logger.setLogLevel(logger.DEBUG);
```

### Processing Session

```
% define session
logger.info("live_script.mlx", "Creating a session...");
session = Session('Session Info', logger);
```

### Setup Python Env

```
PYENV_PATH = '/home/ubuntu/anaconda3/envs/npx/bin/python';
logger.info("live_script.mlx", "Setting up Python environment...");
session.setPyEnv(PYENV_PATH);
logger.info("live_script.mlx", "Done with setting up Python environment!");
% init session by pipelines, stages and jobs
```

### Configuration

The example config file is in the `configs` directory

```
logger.info("live_script.mlx", "Parsing config file for pipeline...");
json = Config.parseJson("configs/test_config.json");
```

### Method 1 - Auto-assembled pipeline

```
logger.info("live_script.mlx", "Auto-assembling a pipeline based on the config file...");
pipelineConfig = PipelineConfig(json.pipeline);
pipeline = Pipeline('Pipeline0 Info', pipelineConfig, logger);
pipeline.autoAssemble(json);

logger.info("live_script.mlx", "Adding the pipeline to session...");
session.addPipeline(pipeline);
logger.info("live_script.mlx", "Executing session...");
session.parExecute();
logger.info("live_script.mlx", "Session execution done!");
```

### Method 2 - Manually-asmubled pipeline

```

logger.info("live_script.mlx", "Manually assemble a pipeline...");

%% Configuration
logger.info("live_script.mlx", "Parsing config file for pipeline...");
json = Config.parseJson("configs/test_config.json");

%% Pipeline 1
logger.info("live_script.mlx", "Set PipelineConfig...");
pipelineConfig1 = PipelineConfig(json.pipeline);
pipeline1 = Pipeline('Pipeline0 Info', pipelineConfig1, logger);

% CatGT stage
logger.info("live_script.mlx", "Creating CatGT stage...");
stageCatgt = Stage('CatGT', logger);
% append stage to pipeline first, in order to pass Pipeline/CommonConfig to tasks
pipeline1.addStage(stageCatgt);
logger.info("live_script.mlx", "CatGT stage was added!");
% CatGT task 1
logger.info("live_script.mlx", "Creating CatGT task config...");
configCatgt1 = TaskConfig(json.CatGT);
logger.info("live_script.mlx", "Creating CatGT task...");
taskCatgt1 = CatGT('CatGT probe 0', '0', 1, configCatgt1, logger);
stageCatgt.addTask(taskCatgt1);
logger.info("live_script.mlx", "CatGT task was added!");

% KiloSort stage
logger.info("live_script.mlx", "Creating KiloSort stage...");
stageKilo = Stage('KiloSort', logger);
pipeline1.addStage(stageKilo);
logger.info("live_script.mlx", "KiloSort stage was added!");
% KiloSort task 1
logger.info("live_script.mlx", "Creating KiloSort task config...");
configKilo1 = TaskConfig(json.KiloSort);
logger.info("live_script.mlx", "Creating KiloSort task...");
taskKilo1 = KiloSort('KiloSort probe 0', '0', 'cortex', configKilo1, logger);
stageKilo.addTask(taskKilo1);
logger.info("live_script.mlx", "KiloSort task was added!");

% TPrime stage
logger.info("live_script.mlx", "Creating TPrime stage...");
stageTPrime = Stage('TPrime', logger);
pipeline1.addStage(stageTPrime);
logger.info("live_script.mlx", "TPrime stage was added!");
% TPrime task 1
logger.info("live_script.mlx", "Creating TPrime task config...");
configTPrime1 = TaskConfig(json.TPrime);
logger.info("live_script.mlx", "Creating TPrime task...");
taskTPrime1 = TPrime('TPrime probe 0', '0', configTPrime1, logger);
stageTPrime.addTask(taskTPrime1);
logger.info("live_script.mlx", "TPrime task was added!");

% append pipeline to session
logger.info("live_script.mlx", "Adding the pipeline to session...");

```

```
session.addPipeline(pipeline1);
%% Execution
logger.info("live_script.mlx", "Executing session...");
session.parExecute();
logger.info("live_script.mlx", "Session execution done!");
```

## Single Task Execution for testing or debugging

```
%% Configuration
logger.info("live_script.mlx", "Parsing config file for pipeline...");
json = Config.parseJson("configs/test_config.json");

pipelineConfig = PipelineConfig(json.pipeline);
config = TaskConfig(json.CatGT);
task = CatGT('CatGT probe 0', '0', 1, config, logger);
% config = TaskConfig(json.KiloSort);
% task = KiloSort('KiloSort probe 0', '0', 'cortex', config, logger);
% config = TaskConfig(json.TPrime);
% task = TPrime('TPrime probe 0', '0', config, logger);

task.CommonConfig = pipelineConfig;
task.execute();
```

## Visualization Examples

These are currently not included in the Neuropixels-toolkit package, just to demonstrate the pipeline's results.

Data files are coming from the data directory in the config json file config["data"], in this case, they are in: `/home/ubuntu/neuropixel/data_for_ecephys/subject1_session1/catgt_SC011_021919_g0/SC011_021919_g0_imec0/imec0_ks2`

```
% this is the same as config["pipeline"]["tools"]["numpyMatlabRepo"]
addpath('/home/ubuntu/neuropixel/npymatlab/npymatlab/');
addpath('.');
```

## Mean Waveforms

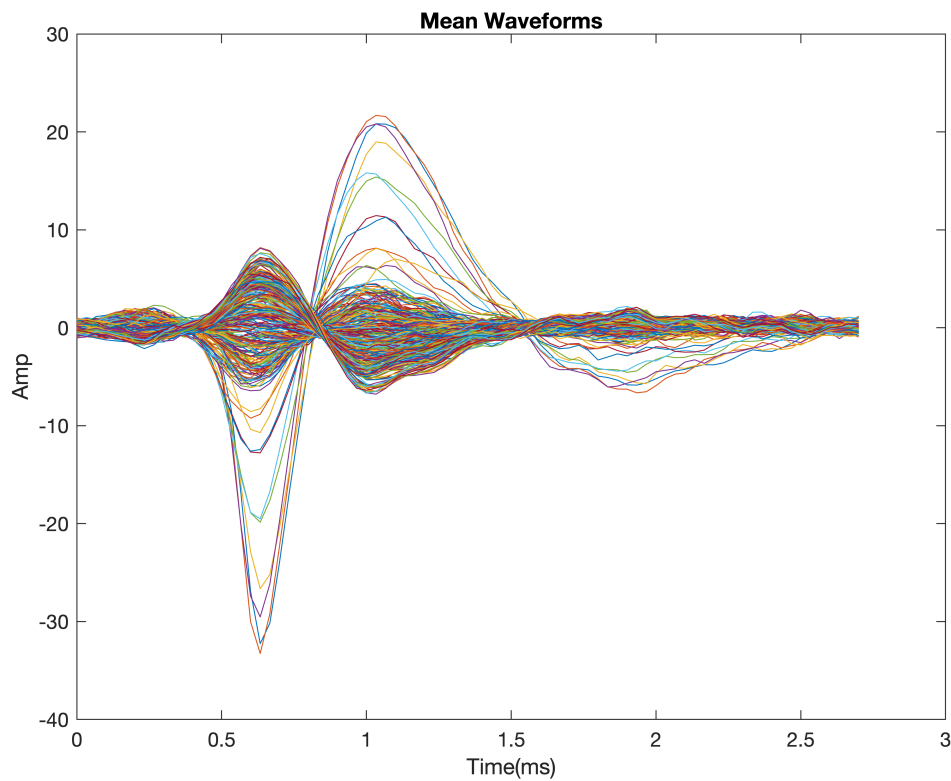
```
meanWaveforms = readNPY('./test_data/mean_waveforms.npy');
disp(size(meanWaveforms));
```

```
275    301    82
```

```
samplingRate = 29999.9
```

```
samplingRate = 3.0000e+04
```

```
one_unit = squeeze(meanWaveforms(2, :, :))';
plot((0:size(one_unit, 1)-1)/samplingRate*1000, one_unit);
title('Mean Waveforms');
xlabel('Time(ms)');
ylabel('Amp');
```



## Spike Time

```
spikeTime = readNPY('./test_data/spike_times.npy');
disp(size(spikeTime));
```

```
7971090      1
```

```
spikeCluster = readNPY('./test_data/spike_clusters.npy');
disp(size(spikeCluster));
```

```
7971090      1
```

```
scatter(spikeTime(1:10000:7971090), spikeCluster(1:10000:7971090));
title('Spike Time vs Spike Clusters downsampled by 10000');
xlabel('Spike Time');
ylabel('Spike Cluster');
```

