```
classdef CrabolaEphysRec
    properties
        stims;
        neurons Neuron;
        ball MiceData;
        date;
        folder;
        crabID;
    end
    methods
        function obj = CrabolaEphysRec(type, input, varargin)
            % CrabolaEphysRec es el constructor de la clase.
            % Tengo que decirle que tipo de input le doy:
            % 'file' si el input es el path a la carpeta con la salida
 del
            % spike sorting
            % 'data' si input es un cell array con:
            % 1- un objeto de la clase MiceData con los datos de la
 crabola
            % del registro
            % 2- un vector de objetos de la clase Neurons con las
 neuronas
            % del registro
            % 3- los estimulos
            % el argumento optativo "samplefreq" me permite setear la
            % frecuencia de sampleo a la que cargar las neuronas
            sf = 30000;
            saveFile = false;
            for arg = 1:2:length(varargin)
                switch lower(varargin{arg})
                    case 'samplefreq'
                        sf = varargin{arg+1};

                    otherwise
                        error([varargin{arg} 'is not a valid
 argument'])
                end
            end
            if strcmp(type, 'file')
                path = input;
                cd(input)
                list = dir;
                for f = 3:length(list)
                    fileList{f-2} = list(f).name;
                end
                %me fijo si existe el archivo recording.mat en la
 carpeta
                %del registro
                for f = 1:length(fileList)
                    if contains(fileList{f}, 'recording')
                        rec = load(replace(fileList{f}, '', ''));
                        obj = rec.obj;
```

```
                       return
                   elseif f == length(fileList)
                       %si no existe lo genero
                       disp('I can find the recording.mat file, we
need to generate it')
                       disp('creating recording.mat file...')
                       neurons = obj.loadClusters(path, 'samplefreq',
sf);
                       ballData = loadBallData(path);
                       obj.stims = neurons(1).stims;
                       saveFile = true;
                   end
               end

           elseif strcmp(type, 'data')
               ballData = input{1};
               neurons = input{2};
               stims = input{3};
               for s = 1:length(stims)
               stimList(s) = struct('code', stims(s,1), ...
                   'start', stims(s,2), ...
                   'finish', stims(s,3), ...
                   'running', false);
               end
               obj.stims = stimList;
           end
           obj.ball = ballData;
           obj.neurons = neurons;
           obj.crabID = ballData.crabID;
           obj.folder = obj.neurons(1).folder;
           if saveFile
               save('recording.mat', 'obj');
               disp('recoding.mat file saved')
           end
           if length(obj.ball.trial) < length(obj.stims)
               disp('There are missing trials on the crabola')
               disp(['i have ' num2str(length(obj.stims)) ' on the
ephys'])
               disp(['but only ' num2str(length(obj.ball.trial)), '
on the crabola']);
           end
       end

       function stimIND = getStimIndex(obj, stimCodes, varargin)
           condition = 'all';
           for arg = 1:2:length(varargin)
               switch lower(varargin{arg})
                   case 'condition'
                       if sum(strcmp({'all', 'ball', 'air'},
varargin{arg+1}))
                           condition = varargin{arg+1};
                       else
                           error('invalid "condition", only "all",
"ball" and "air" are permited')
```

```
                       end
                   end
               end
               %find selected stims
               stimIND = find(ismember([obj.stims.code], stimCodes));
               if strcmp(condition, 'ball')
                   for n = flip(1:length(stimIND))
                       if ~obj.stims(stimIND(n)).running
                           stimIND(n) = [];
                       end
                   end
               elseif strcmp(condition, 'air')
                   for n = flip(1:length(stimIND))
                       if obj.stims(stimIND(n)).running
                           stimIND(n) = [];
                       end
                   end
               end

           end


           function makeMixedPlots(obj, stim, cluster, varargin)
               condition = 'all';
               xlimit = [-10, 15];
               binSize = 50;
               titleTxt = '';
               behavior = 'tras';
               for arg = 1:2:length(varargin)
                   switch lower(varargin{arg})
                       case 'condition'
                           if sum(strcmp({'all', 'ball', 'air'},
varargin{arg+1}))
                               condition = varargin{arg+1};
                           else
                               error('invalid "condition", only "all",
"ball" and "air" are permited')
                           end
                       case 'xlim'
                           xlimit = varargin{arg+1};
                       case 'binsize'
                           binSize = varargin{arg+1};
                       case 'title'
                           titleTxt = varargin{arg+1};
                       case 'behavior'
                           if sum(strcmp({'tras', 'rot', 'dir'},
varargin{arg+1}))
                               behavior = varargin{arg+1};
                           else
                               error('invalid "behavior", only "all",
"ball" and "air" are permited')
                           end

                   end
```

```
            end
            nBins = round((xlimit(2)-xlimit(1))*(1000/binSize));

            stimIND = obj.getStimIndex(stim, 'condition', condition);
            [raster, index, stimList] =
 obj.neurons(cluster).getRasters(stim, 'durations', [10,
 15], 'stimIndex', stimIND);

            for i = unique(index)'
                disp([' trial ' num2str(i) ' has ' num2str(sum(index
 == i)) ' spikes'])
            end

            lTopLimit = 0;
            rTopLimit = 0;
            figure;
            suptitle(titleTxt)
            hold on
            for ns = 1:length(stimIND)
                s = stimIND(ns);
                subplot(length(stimIND), 1,ns)
                run = obj.ball.interpolateRuns(s, binSize/1000);
                if strcmp(behavior, 'tras')
                    runPar = run.vTras;
                    behLabel = 'traslational speed (cm/s)';
                elseif strcmp(behavior, 'rot')
                    runPar = run.vRot;
                    behLabel = 'rotational speed (deg/s)';
                else
                    runPar = run.Dir;
                    behLabel = 'direction (deg)';
                end

                if ~isempty(runPar)
                    yyaxis left
                    plot(run.time-10, smooth(runPar, 5), 'linewidth',
 2)

                end
                if lTopLimit < max(ylim)
                    lTopLimit = max(ylim);

                end
                if ns == round(length(stimIND)/2)
                    ylabel(behLabel);
                end
                [raster, index, stimList] =
 obj.neurons(cluster).getRasters(2, 'durations', [abs(xlimit(1)),
 abs(xlimit(2))], 'stimIndex', s);
%                 [freq,~] = SyncHist(raster(index == ns),
 index(index==ns),'mode', 'mean' ,'durations',...
%                                     [xlimit(1); xlimit(2)], 'nBins',
 nBins);
```

```
                [freq, t] = obj.neurons(cluster).getPSH(raster, index,
 xlimit, nBins);

%                 freq = smooth(freq, 5);
%                 t = (xlimit(1):(xlimit(2) - xlimit(1))/
(nBins-1):xlimit(2))';
                if isempty(freq)
                    freq = zeros(size(t));
                end
                yyaxis right
                plot(t, freq)
                if rTopLimit < max(ylim)
                    rTopLimit = max(ylim);
                end
                if ns == round(length(stimIND)/2)
                    ylabel('firing freq (Hz)');
                end
                if ns == length(stimIND)
                    xlabel('time (s)')
                end
            end

            for ns = 1:length(stimIND)
                subplot(length(stimIND), 1,ns)
                s = stimIND(ns);
                yyaxis left
                ylim([min(ylim) lTopLimit])
                line([obj.stims(s).finish - obj.stims(s).start,
 obj.stims(s).finish - obj.stims(s).start], [0, lTopLimit])
                %addPSHDecorations(stim, obj.stims(s).finish -
 obj.stims(s).start ,lTopLimit, 'StimUnderPlot', true)
                %PlotRasters_oneColor(raster(index == ns),
 index(index==ns),[-10, 15], max(ylim), 'RelativeSize', 0.1,
 'position', 'botom')
                yyaxis right
                ylim([min(ylim) rTopLimit])

                %addPSHDecorations(stim, obj.ball.trial(s).duration,
 40, 'stimUnderPlot', false, 'heigth', 0.2)
                xlim(xlimit)
            end

        end


        function neurons = loadClusters(obj, path, varargin)

            % loadClusters toma el path de la carpeta donde estan los
            % archivos ya sorteados y levanta los clusters (ignorando
 el 0
            % que corresponde a artefatos). Devuelve un vector de
 neuronas
            % de la clase "Neurons"
```

```
           %con el argunmento optativo "samplefreq" puedo setear la
frecuencia de
           %sampleo del registro
           sf = 30000;
           for arg = 1:2:length(varargin)
               switch lower(varargin{arg})
                   case 'samplefreq'
                       if varargin{arg+1} > 0
                           sf = varargin{arg+1};
                       else
                           error('sample frequency must be > 0')
                       end
               end
           end
           cd (path)

           %cargo los estimulos
           load('Estimulos.mat');
           %cargo los monitores
           load('Monitores.mat');
```

Levanto los datos de los clusters

```
           id = path(end-7:end);
           %busco el archivo .clu (contiene el cluster asignado a
cada spike) en la
           %carpeta del experimento
           files = dir;
           for f = 1:length(files)
               name = string(files(f).name);
               if name.contains([id '.clu'])
                   cluDataFile = name;
                   break
               end
               if f == length(files)
                   error('I cannot find the .clu file');
               end
           end
           %genro un vector con el numero de cluster
           clusterData = importdata(cluDataFile(1,:));
           nCluster = clusterData(2:end);
           clear cluDataFile
           clear clusterData


           %busco el archivo con los tiempos de cada spike
           for f = 1:length(files)
               name = string(files(f).name);
               if name.contains([id '.res'])
                   timeDataFile = name;
                   break
               end
               if f == length(files)
```

```
                    error('I cannot find the .res file');
                end
            end

            %llevo los tiempos de los spikes de samples a segundos
            tSpikes = importdata(timeDataFile) / sf;
            clear timeDataFile
            for cluster = 1:max(nCluster)
                neuron.data = tSpikes(nCluster == cluster);
                neuron.file = path;
                neuron.cluster = cluster;
                neuron.Estimulos = Estimulos;
                neuron.Monitores = Monitores;
                neuron.name = [id '-C' num2str(cluster)];
                neurons(cluster) = Neuron(neuron);
            end

        end
    end
end
```

Published with MATLAB® R2018a