LoadSlopeyResults (1 call, 20.981 sec)

Generated 21-Apr-2016 09:07:30 using cpu time.
function in file /Users/Steph/Documents/UCSF/Narlikar lab/HMM analysis Slopey/slopey/LoadSlopeyResults.m
Copy to new window for comparing multiple runs

Refresh

- [x] Show parent functions
- [x] Show busy lines
- [x] Show child functions
- [x] Show Code Analyzer results
- [x] Show file coverage
- [x] Show function listing

Parents (calling functions)

Function Name	Function Type	Calls
RunSlopeyAnalysis	function	1

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
52	`results{k}.vals(j,:) = samples...`	460046	3.791 s	18.1%	
51	`results{k}.times(j,:) = sample...`	460046	3.712 s	17.7%	
49	`results{k}.offset(j,:) = sampl...`	480048	3.291 s	15.7%	
10	`results_py = load(fullfile(mai...`	1	3.238 s	15.4%	
48	`results{k}.ch2_transform(j,:) ...`	480048	2.975 s	14.2%	
All other lines			3.975 s	18.9%	
Totals			20.981 s	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time	Time Plot
fullfile	function	2	0 s	0%	
Self time (built-ins, overhead, etc.)			20.981 s	100.0%	
Totals			20.981 s	100%	

Code Analyzer results

Line number	Message
5	`Input argument 'data_name' might be unused. If this is OK, consider replacing it by ~.`

Coverage results
Show coverage for parent directory

Total lines in function	56

```
 Slopey/slopey';

;

nd samples.

giving the

ing the red

parameter

nsformation
d ones.

l}{1},1));
}{2},1));
```

Non-code lines (comments, blank lines)	20
Code lines (lines that can run)	36
Code lines that did run	36
Code lines that did not run	0
Coverage (did run/can run)	100.00 %

Function listing

Color highlight code according to time

time	calls	line	
		5	function results = LoadSlopeyResults(data_name)
		6	
	1	7	maindir = '/Users/Steph/Documents/UCSF/Narlikar lab/HMM analysis
	1	8	names = dir(fullfile(maindir,'data','*.mat'));
		9	
3.24	1	10	results_py = load(fullfile(maindir,'results','all_results.mat'))
		11	
	1	12	results = cell(1,length(names));
		13	
	1	14	for k = 1:length(names)
0.01	48	15	struct_py = results_py.(names(k).name(1:end-4));
		16	
	48	17	results{k}.name = names(k).name(1:end-4);
	48	18	results{k}.fps = 1/struct_py.params.T_cycle;
	48	19	results{k}.data = struct_py.data;
	48	20	results{k}.start = double(struct_py.params.start);
	48	21	results{k}.end = double(struct_py.params.end);
		22	
	48	23	samples = struct_py.samples;
		24	% Each structure in allresults has 3 fields: params, data, a
		25	% Samples is a num_iterations+1-by-3 cell array. If n is
		26	% num_iterations:
		27	% currstruct.samples{n,1}{1} is a num_slopey+1-by-1 vector,
		28	% times in seconds of the start and end of each slopey bit.
		29	% currstruct.samples{n,1}{2} is a num_slopey-by-1 vector giv
		30	% intensity values for each flat bit that separates slopeys.
		31	% currstruct.samples{n,2} is a double that gives the offset
		32	% for converting from seconds to frames.
		33	% currstruct.samples{n,3} is a 2x1 vector that gives the tra
		34	% parameters to obtain real green values instead of idealize
		35	
0.01	48	36	results{k}.ch2_transform = zeros(size(samples,1),2);
	48	37	results{k}.offset = zeros(size(samples,1),1);
	48	38	try
	48	39	results{k}.times = zeros(size(samples,1),size(samples{1,
0.01	46	40	results{k}.vals = zeros(size(samples,1),size(samples{1,1

```
    46  41          non_vector = 1;
     2  42      catch
     2  43          results{k}.times = zeros(size(samples,1),2);
     2  44          results{k}.vals = zeros(size(samples,1),2);
     2  45          non_vector = 0;
     2  46      end
    48  47      for j = 1:size(samples,1)
2.97 480048 48          results{k}.ch2_transform(j,:) = samples{j,3};
3.29 480048 49          results{k}.offset(j,:) = samples{j,2};
1.58 480048 50          if non_vector == 1
3.71 460046 51              results{k}.times(j,:) = samples{j,1}{1};
3.79 460046 52              results{k}.vals(j,:) = samples{j,1}{2};
0.04  20002 53          else
0.16  20002 54              results{k}.times(j,:) = samples{j,1}(1,:);
0.18  20002 55              results{k}.vals(j,:) = samples{j,1}(2,:);
0.03  20002 56          end
1.51 480048 57      end
            58
    48  59      clear samples
    48  60  end
```