driver (Calls: 1, Time: 310.932 s)

Generated 27-Dec-2021 14:29:39 using performance time.
Script in file C:\Users\zmeri\Documents\GitHub\NSE\Generate Data\driver.m
Copy to new window for comparing multiple runs

Parents (calling functions)

No parent

Lines that take the most time

Line Number	Code	Calls	Total Time (s)	% Time	Time Plot
196	`= stitch_pdf(sample,rndom.filename,send_file_nam…`	60	158.150	50.9%	
251	`kl_info_se = dist_list(kl_info_se);`	60	28.091	9.0%	
172	`rndom = dist_list(rndom);`	60	25.898	8.3%	
255	`kl_info_nmem = dist_list(kl_info_nmem);`	60	25.787	8.3%	
285	`rndom.dist_list();`	60	25.738	8.3%	
All other lines			47.269	15.2%	
Totals			310.932	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time (s)	% Time	Time Plot
stitch_pdf	Function	60	158.108	50.8%	
distributions>distributions.dist_list	Class method	260	114.758	36.9%	
dlmwrite	Function	385	23.650	7.6%	
EstimatePDF	MEX-file	60	13.621	4.4%	
misc_functions>misc_functions.sqr	Class method	120	0.111	0.0%	
interp1	Function	492	0.073	0.0%	
num2str	Function	360	0.031	0.0%	
datasample	Function	120	0.007	0.0%	
std	Function	20	0.005	0.0%	
linspace	Function	65	0.003	0.0%	
mean	Function	25	0.002	0.0%	
misc_functions>misc_functions.sample_pow	Class method	5	0.001	0.0%	
distributions>distributions.distributions	Class method	1	0.000	0.0%	
Self time (built-ins, overhead, etc.)			0.561	0.2%	
Totals			310.932	100%	

Code Analyzer results

Line Number	Message
55	`The value assigned to variable 'distribution_vector' might be unused.`

Coverage results

Show coverage for parent folder

Total lines in function	361
Non-code lines (comments, blank lines)	146
Code lines (lines that can run)	215
Code lines that did run	199
Code lines that did not run	16
Coverage (did run/can run)	92.56 %

Function listing

Time	Calls	Line	
< 0.001	1	2	`clc;clear;`

```
                3  % class assignment
< 0.001         1  4  actual = distributions;
< 0.001         1  5  actual.generate_data = false;
                6
                7  % SUBSAMPLING PARAMETERS-----------------------------------------------------
                8
< 0.001         1  9  BootSample = "off";
< 0.001         1 10  if BootSample == "off"
               11      % percentage of sample used to create subsample
< 0.001         1 12      percSample = 1;
               13      % number of subsamples to generate
< 0.001         1 14      numSubs = 1;
               15  else
               16      % percentage of sample used to create subsample
               17      percSample = 0.4;
               18      % number of subsamples to generate
               19      numSubs = 30;
< 0.001         1 20  end
               21
               22  %-------------------------------------------------------------------------
               23
< 0.001         1 24  tic
               25  % User Options ============================================================
               26  % script switching board
< 0.001         1 27  estimator_call_flag =       true;   %<- true/false call SE on/off
< 0.001         1 28  estimator_plot_flag =       false;   %<- true/false plot SE results on/off
< 0.001         1 29  data_type_flag =            true;   %<- true/false integer powers of 2/real powers of 2
< 0.001         1 30  save_graphics =             false;   %<- true/false save .png of plots on/off
               31  % rndom data generation parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
< 0.001         1 32  max_pow =                   10; %<---- maximum exponent to generate samples
< 0.001         1 33  min_pow =                   8; %<---- minimum exponent to generate samples
< 0.001         1 34  trials =                    4;  %<--- trials to run to generate heuristics for programs
< 0.001         1 35  step =                      1;  %<---- control synthetic rndom samples to skip being created
< 0.001         1 36  temp_min_limit =            0; %<---- set upper limit for both
< 0.001         1 37  actual.min_limit =          temp_min_limit;  %<--- lower limit to plot
< 0.001         1 38  temp_max_limit =            1000; %<---- set upper limit for both
< 0.001         1 39  actual.max_limit =          temp_max_limit; %<--- upper limit to plot
< 0.001         1 40  x_resolution =              1000;
               41  % changes how data files are accessed when using data generated from the
               42  % Univariant Random Sample Generator available on zmerino's github
< 0.001         1 43  cpu_type =                   '\';%<--- '\' or '/' for windows or linux
               44
               45  % Example distribution to test %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
               46  %{
               47   distribution_vector = ["Beta-a0p5-b1p5","Beta-a2-b0p5","Beta-a0p5-b0p5",...
               48      "Bimodal-Normal","BirnbaumSaunders","Burr",...
               49      "Exponential","Extreme-Value","Gamma","Generalized-Extreme-Value",...
               50      "Generalized-Pareto","HalfNormal","Normal","Square-periodic",...
               51      "tLocationScale","Uniform","Uniform-Mix","Weibull","Chisquare",...
               52      "InverseGaussian","Trimodal-Normal","Stable",...
               53      "Stable2","Stable3","Stable1","BirnbaumSaunders-Stable"];
               54  %}
< 0.001         1 55  distribution_vector = ["Generalized-Pareto","Stable","Trimodal-Normal","Normal", "Uniform","Beta-a0p5-b1p5","Beta-
< 0.001         1 56  distribution_vector = ["Generalized-Pareto","Stable","Trimodal-Normal","Normal","Uniform"];
               57  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
               58  %% Main Function Call Loop used to lable plot figures
               59
               60  % find any of the strings in "str" inside of "distribtuionVector"
< 0.001         1 61  str = ["Beta-a0p5-b1p5","Beta-a2-b0p5","Beta-a0p5-b0p5"];
< 0.001         1 62  flag = zeros(1,length(distribution_vector));
< 0.001         1 63  for a = 1:size(distribution_vector,2)
< 0.001         5 64      for b = 1:size(str,2)
< 0.001        15 65          if strcmp(distribution_vector(a),str(b))
               66              flag(a) = 1;
               67              break
< 0.001        15 68          else
< 0.001        15 69              flag(a) = 0;
```

```
< 0.001    15    70          end
< 0.001    15    71      end
< 0.001     5    72  end
                 73
                 74  % Initialize 3D matirx with zeros that is trials by # of samples sizes by
                 75  % # number of distributions.
                 76  % NOTE: only works for division with no remainders
< 0.001     1    77  fail_nmem = zeros(trials,(max_pow-min_pow)/step,length(distribution_vector));
< 0.001     1    78  fail_se = zeros(trials,(max_pow-min_pow)/step,length(distribution_vector));
                 79
                 80  % Loop over every distribution type
< 0.001     1    81  for j = 1:length(distribution_vector)
                 82      % Define plot vector for dist_list from 0-1
< 0.001     5    83      if flag(j)
                 84          actual.min_limit = 0;
                 85          actual.max_limit = 1;
                 86          actual.x = linspace(actual.min_limit,actual.max_limit,x_resolution); %<----******** need to update code
< 0.001     5    87      else
                 88          % Define plot vector for distribution from actual.min_limit-actual.max_limit
< 0.001     5    89          actual.min_limit = 0;
< 0.001     5    90          actual.max_limit = 10;
  0.003     5    91          actual.x = linspace(actual.min_limit,actual.max_limit,x_resolution);
< 0.001     5    92      end
                 93      % Current distribution name
< 0.001     5    94      actual.dist_name = distribution_vector(j);
                 95      % file name for actual distribution. "A_" puts at the top of the folder.
  0.001     5    96      actual.filename = sprintf(['A_', char(actual.dist_name),'_Act']);
                 97      % creat rndom object
  0.002     5    98      rndom = actual;
                 99
                100      % Initialize empty or zero filled matrices/arrays ---------------------
                101
                102      % track calculated threshold per distrobution
< 0.001     5   103      T_track = zeros(max_pow-min_pow,1);
                104      % track BR per branch level
< 0.001     5   105      BR_track = zeros(max_pow-min_pow,trials);
                106      % track BR at the zeroth  branch level
< 0.001     5   107      BR0_track = zeros(max_pow-min_pow,trials);
< 0.001     5   108      sample_data = zeros(max_pow-min_pow,1);
< 0.001     5   109      sample_data_box = [];
< 0.001     5   110      kl_se = [];
< 0.001     5   111      kl_nmem = [];
< 0.001     5   112      Hellinger_se = [];
< 0.001     5   113      Hellinger_nmem = [];
< 0.001     5   114      max_LG_se = zeros(max_pow-min_pow,trials);
< 0.001     5   115      max_LG_nmem = zeros(max_pow-min_pow,trials);
< 0.001     5   116      cpu_vec_se = zeros(max_pow-min_pow,trials);
< 0.001     5   117      cpu_vec_nmem = zeros(max_pow-min_pow,trials);
                118  %     dist_BR0_track = cell(max_pow-min_pow);
                119  %     dist_BR_track = cell(max_pow-min_pow);
< 0.001     5   120      dist_BR0_track = cell(length(distribution_vector));
< 0.001     5   121      dist_BR_track = cell(length(distribution_vector));
< 0.001     5   122      MSE_se = [];
< 0.001     5   123      MSE_nmem = [];
< 0.001     5   124      all_se = cell(max_pow-min_pow,trials);
< 0.001     5   125      all_nmem = cell(max_pow-min_pow,trials);
< 0.001     5   126      cpu_time_se = [];
< 0.001     5   127      cpu_time_nmem = [];
< 0.001     5   128      sample_track = [];
                129
                130      % Create vector of  samples
  0.003     5   131      sample_vec = misc_functions.sample_pow(min_pow,max_pow,data_type_flag,step);
< 0.001     5   132      BoxCPUtimeSE = zeros((max_pow-min_pow)*trials,2);
< 0.001     5   133      BoxCPUtimeNMEM = zeros((max_pow-min_pow)*trials,2);
                134
< 0.001     5   135      for i = 1:trials
                136
```

```
9.311    20  137          actual = actual.dist_list();
             138
0.001    20  139          for k = 1:length(sample_vec)
             140
< 0.001  60  141              interp_etimate = [];
0.002    60  142              estimate_data = {};
             143
             144              % initialize to 0. used to count number of failed NMEM pdfe
< 0.001  60  145              fail_flag = 0;
             146
0.001    60  147              rndom.Ns = sample_vec(k);
0.007    60  148              realx = linspace(actual.min_limit,actual.max_limit,rndom.Ns);
             149              % p-vector definition for Rtree
0.002    60  150              p = [1,0.55,1,0.33,2,ceil(0.0625*rndom.Ns^0.5),40];
             151
             152              % Create rndom.filename for each distribtuion
0.007    60  153              rndom.filename = sprintf(['D_', char(actual.dist_name),'_T_','%d', '_S_','%d'],i, rndom.Ns);
             154
             155              % run estimator and store data
0.001    60  156              if estimator_call_flag
             157                  % file path name
0.005    60  158                  send_file_name = ['D_',...
             159                      char(actual.dist_name),...
             160                      cpu_type,char(rndom.filename),...
             161                      '.dat'];
             162
             163                  %=========================================================
             164                  % % % % % % % % start of estimate % % % % % % % % % %
             165                  %=========================================================
0.001    60  166                  tintialSE = cputime;
             167
             168                  % initial details for subsample
0.004    60  169                  sendFileName1 = ['D_',char(actual.dist_name),cpu_type,char(rndom.filename),'.dat'];
             170
0.002    60  171                  rndom.randomVSactual = "random";
25.898   60  172                  rndom = dist_list(rndom);
0.002    60  173                  sample = rndom.rndData;
             174
             175                  % while loop to replace non-finite values with finite values
             176
             177                  %                 pd = rndom.distInfo;
             178                  %                 non_finite_vals = sum(~isfinite(sample));
             179                  %                 % check sample
             180                  %                 for inf_index = 1:non_finite_vals
             181                  %                     index = isinf(sample);
             182                  %                     num_inf = non_finite_vals;
             183                  %
             184                  %                     new_data = inf;
             185                  %                     disp(['Old: ', num2str(sample(inf_index))])
             186                  %                     while isinf(new_data)
             187                  %                         new_data = random(pd);
             188                  %                         sample(inf_index) = new_data;
             189                  %                     end
             190                  %                     disp(['New: ', num2str(sample(inf_index))])
             191                  %                 end
             192                  %
             193
             194                  % mixture model does not return random sample
158.153  60  195                  [fail_code,x,SE_pdf,SE_cdf,SE_u,SE_SQR,nBlocks,Blacklist,rndom.Ns,binrndom.Ns, max_LG, sum_LG,T,BR
             196                      = stitch_pdf(sample,rndom.filename,send_file_name,actual.min_limit,actual.max_limit,p);
             197
< 0.001  60  198                  fail_se(i,k,j) = fail_code;
             199
             200                  % track T,BR per trial
             201
< 0.001  60  202                  T_track(k) = T;
< 0.001  60  203                  BR0_track(k,i) = BR0;
```

```
0.005   60  204                 BR_track(k,i) = sum(BRlevel{end,1});
            205
            206                 % store subsample estimate data
0.002   60  207                 estimate_data.x = x;
< 0.001 60  208                 estimate_data.pdf = SE_pdf;
< 0.001 60  209                 estimate_data.u = SE_u;
0.003   60  210                 estimate_data.sqr = SE_SQR;
            211
            212
0.001   60  213                 tcpuSE = cputime-tintialSE;
< 0.001 60  214                 tintialNMEM = cputime;
            215
            216                 %-- NMEM start
            217                 %[failed, x_NMEM, pdf_NMEM, cdf_NMEM,sqr_NMEM, ~,lagrange] = EstimatePDF(sample);
< 0.001 60  218                 try
13.634  60  219                     [failed, x_NMEM, pdf_NMEM, cdf_NMEM,sqr_NMEM, ~,lagrange] = EstimatePDF(sample);
< 0.001 60  220                     fail_nmem(i,k,j) = 0;
1.162   60  221                     dlmwrite(['NMEM_pdf_',rndom.filename,'.dat'],[x_NMEM, pdf_NMEM],'Precision',12)
1.152   60  222                     dlmwrite(['NMEM_cdf_',rndom.filename,'.dat'],[x_NMEM, cdf_NMEM],'Precision',12)
            223
< 0.001 60  224                     n = length(sqr_NMEM);
< 0.001 60  225                     dx = 1 / (n + 1);
0.001   60  226                     u_NMEM = dx:dx:(n * dx);
            227
0.963   60  228                     dlmwrite(['NMEM_sqr_',rndom.filename,'.dat'],[u_NMEM', sqr_NMEM],'Precision',12)
            229                 catch
            230                     warning('Problem using function.  Assigning a value of 0.');
            231                     lagrange = 0;
            232                     x_NMEM = linspace(min(sample),max(sample),length(sample))';
            233                     pdf_NMEM = 0*ones(length(sample),1);
            234                     cdf_NMEM = 0*ones(length(sample),1);
            235                     fail_nmem(i,k,j) = 1;
< 0.001 60  236                 end
            237                 %------------------------------------------------------------
< 0.001 60  238                 tcpuNMEM = cputime-tintialNMEM ;
0.025   60  239                 disp(['NMEM elapsed time is ',num2str(tcpuNMEM),' seconds.'])
            240                 %============================================================
            241                 % % % % % % % % end of estimate % % % % % % % % %
            242                 %============================================================
            243
            244                 % calculate LG multipliers
< 0.001 60  245                 max_LG_se(k,i) = sum(max_LG);
< 0.001 60  246                 max_LG_nmem(k,i) = sum(length(lagrange));
            247
            248                 % calculate kl values
0.021   60  249                 kl_info_se = actual;
0.004   60  250                 kl_info_se.x = x;
28.091  60  251                 kl_info_se = dist_list(kl_info_se);
            252
0.018   60  253                 kl_info_nmem = actual;
0.004   60  254                 kl_info_nmem.x = x_NMEM';
25.787  60  255                 kl_info_nmem = dist_list(kl_info_nmem);
0.003   60  256                 kl_info_nmem;
            257                 %------------------------------------------------------------
            258
0.020   60  259                 SE_test_x = interp1(kl_info_se.x,kl_info_se.pdf_y,x);
0.012   60  260                 NMEM_test_x = interp1(kl_info_nmem.x,kl_info_nmem.pdf_y,x_NMEM');
            261
0.029   60  262                 if sum(~isfinite(interp1(kl_info_se.x,kl_info_se.pdf_y,x)))...
        60  263                         || sum(~isfinite(SE_pdf))... % for some reason indefinite values
        60  264                         || sum(~isfinite(pdf_NMEM))...
        60  265                         || sum(~isfinite(interp1(kl_info_nmem.x,kl_info_nmem.pdf_y,x_NMEM')))
            266                     disp('non-finite values')
< 0.001 60  267                 end
0.017   60  268                 actual_se_pdf = interp1(kl_info_se.x,kl_info_se.pdf_y,x);
0.012   60  269                 actual_nmem_pdf = interp1(kl_info_nmem.x,kl_info_nmem.pdf_y,x_NMEM');
```

```
                    % find rand sample to find kl for
0.010    60  272    [s_se,idx1] = datasample(SE_pdf,length(x_NMEM));
0.013    60  273    [s_nmem,idx2] = datasample(pdf_NMEM',k);
                    
0.001    60  275    if max(x_NMEM) > max(x) && min(x) > min(x_NMEM)
0.005    20  276        kl_se_test = sum(SE_pdf' - interp1(actual.x',actual.pdf_y',x'));
0.006    20  277        kl_nmem_test = sum(interp1(x_NMEM,pdf_NMEM,x') - interp1(actual.x',actual.pdf_y',x'));
0.003    40  278    elseif max(x_NMEM) < max(x) && min(x) < min(x_NMEM)
0.018    24  279        kl_se_test = sum(interp1(x',SE_pdf',x_NMEM) - interp1(actual.x',actual.pdf_y',x_NMEM));
0.004    24  280        kl_nmem_test = sum(pdf_NMEM - interp1(actual.x',actual.pdf_y',x_NMEM));
< 0.001  60  281    end
                    
                    % read in actual distribution
                    
25.738   60  285    rndom.dist_list();
                    
                    % Create final answer file
0.013    60  288    rndom.filename = sprintf(['D_', char(rndom.dist_name),'_T_','%d', '_S_','%d'],i, rndom.Ns);
                    
0.004    60  290    Sanswer = [x',SE_pdf'];
0.002    60  291    S_sqr = [SE_u',SE_SQR'];
0.002    60  292    S_cdf = [x',SE_cdf'];
< 0.001  60  293    if fail_code == 0
9.756    60  294        dlmwrite(['SE_pdf_',rndom.filename,'.dat'],Sanswer, 'delimiter',' ','precision',12)
1.013    60  295        dlmwrite(['SE_sqr_',rndom.filename,'.dat'],S_sqr, 'delimiter',' ','precision',12)
9.587    60  296        dlmwrite(['SE_cdf_',rndom.filename,'.dat'],S_cdf, 'delimiter',' ','precision',12)
< 0.001  60  297    end
< 0.001  60  298end
                    
                % SQR plot function
0.101    60  301[u_estimate,sqr_estimate] = misc_functions.sqr(x,SE_pdf,sample);
0.042    60  302[u_NMEM,sqr_NMEM] = misc_functions.sqr(x_NMEM,pdf_NMEM,sample);
                
                % store pdf/sqr solutions for all dist/trials/samples
0.006    60  305all_se{i,k}.x(:,1) = x;
0.004    60  306all_se{i,k}.pdf(:,1) = SE_pdf;
0.004    60  307all_nmem{i,k}.x(:,1) = x_NMEM;
0.004    60  308all_nmem{i,k}.pdf(:,1) = pdf_NMEM;
0.004    60  309all_se{i,k}.u(:,1) = u_estimate;
0.003    60  310all_se{i,k}.sqr(:,1) = sqr_estimate;
0.003    60  311all_nmem{i,k}.u(:,1) = u_NMEM;
0.004    60  312all_nmem{i,k}.sqr(:,1) = sqr_NMEM;
                
0.031    60  314disp([char(actual.dist_name),...
         60  315    ', Trial: ',num2str(i),'/', num2str(trials), ...
         60  316    ' sample size: ',num2str(sample_vec(k)), ...
         60  317    ' failSE: ', num2str(fail_se(i,k,j)), ' failNMEM: ', num2str(fail_nmem(i,k,j))])
                
                % store time of computation
< 0.001  60  320cpu_vec_se(k,i) = tcpuSE;
< 0.001  60  321cpu_vec_nmem(k,i) = tcpuNMEM;
0.011    60  322    end
                
0.008    20  324    se_time_per_trial = horzcat(sample_vec',cpu_vec_se(:,i));
0.001    20  325    nmem_time_per_trial = horzcat(sample_vec',cpu_vec_nmem(:,i));
                
0.013    20  327    [p,q] = size(se_time_per_trial);
< 0.001  20  328    BoxCPUtimeSE(end-p+1:end, end-q+1:end) = nmem_time_per_trial;
0.004    20  329    [p,q] = size(nmem_time_per_trial);
< 0.001  20  330    BoxCPUtimeNMEM(end-p+1:end, end-q+1:end) = nmem_time_per_trial;
                
0.002    20  332    rndom.filename = sprintf(['D_', char(actual.dist_name),'_T_','%d'],i);
0.005    20  333end
                
< 0.001   5  335disp(j)
< 0.001   5  336disp(size(dist_BR0_track))
```

< 0.001	5	337	disp(size(dist_BR0_track{j}))
< 0.001	5	338	disp(size(BR0_track))
		339	
< 0.001	5	340	dist_BR0_track{j} = BR0_track;
0.003	5	341	dist_BR_track{j} = BR_track;
		342	
		343	%--
		344	
0.014	5	345	stdCPUtimeSE = horzcat(sample_vec',std(cpu_vec_se,[],2));
0.009	5	346	stdCPUtimeNMEM = horzcat(sample_vec',std(cpu_vec_nmem,[],2));
0.010	5	347	avgCPUtimeSE = horzcat(sample_vec',mean(cpu_vec_se,2));
0.001	5	348	avgCPUtimeNMEM = horzcat(sample_vec',mean(cpu_vec_nmem,2));
		349	
0.006	5	350	cpu.stdTimeSE{j} = horzcat(sample_vec',std(cpu_vec_se,[],2));
0.006	5	351	cpu.stdTimeNMEM{j} = horzcat(sample_vec',std(cpu_vec_nmem,[],2));
0.002	5	352	cpu.timeSE{j} = horzcat(sample_vec',mean(cpu_vec_se,2));
0.002	5	353	cpu.timeNMEM{j} = horzcat(sample_vec',mean(cpu_vec_nmem,2));
		354	
0.012	5	355	dlmwrite(['cpu_SE_',rndom.filename,'.dat'],horzcat(sample_vec',cpu_vec_se), 'delimiter',' ','precision',12)
0.013	5	356	dlmwrite(['cpu_NMEM_',rndom.filename,'.dat'],horzcat(sample_vec',cpu_vec_nmem), 'delimiter',' ','precision',1
0.011	5	357	dlmwrite(['BRTrack_SE_',rndom.filename,'.dat'],horzcat(sample_vec',mean(dist_BR0_track{j},2)), 'delimiter','
0.011	5	358	dlmwrite(['Failures_SE_',rndom.filename,'.dat'],horzcat(sample_vec',sum(fail_se(:,:,j),1)'/trials), 'delimite
0.010	5	359	dlmwrite(['Failures_NMEM_',rndom.filename,'.dat'],horzcat(sample_vec',sum(fail_nmem(:,:,j),1)'/trials), 'delir
		360	
0.002	5	361	end
< 0.001	1	362	toc

Local functions in this file are not included in this listing.