

Matlab Toolbox Heterogeneous Agents Dynamic Programming

Fan Wang

2020-06-22

Contents

Preface	5
1 Summarize Policy and Value	7
1.1 FF_SUMM_ND_ARRAY Examples	7
2 Distributional Analysis	13
2.1 FF_SIMU_STATS Examples	13
2.2 FF_DISC_RAND_VAR_STATS Examples	18
2.3 FF_DISC_RAND_VAR_MASS2OUTCOMES Examples	23
2.4 FF_DISC_RAND_VAR_MASS2COVCOR Examples	26
3 Graphs	31
3.1 FF_GRAPH_GRID Examples: X, Y and Color Line Plots	31
4 Support Tools	37
4.1 FF_CONTAINER_MAP_DISPLAY Examples	37
A Index and Code Links	43
A.1 Summarize Policy and Value links	43
A.2 Distributional Analysis links	43
A.3 Graphs links	43
A.4 Support Tools links	44

Preface

This is a work-in-progress Matlab package consisting of functions that facilitate Dynamic Programming and Related Tasks. Materials gathered from various [projects](#) in which Matlab code is used. Files are the [MEconTools](#) repository. Matlab files are linked below by section with livescript files. Tested with [Matlab](#) 2019a ([The MathWorks Inc, 2019](#)).

This bookdown file is a collection of mlx based vignettes for functions that are available from [MEconTools](#). Each Vignette file contains various examples for invoking each function. The goal of this repository is to make it easier to find/re-use codes produced for various projects.

From other repositories: For dynamic borrowing and savings problems, see [Dynamic Asset Repository](#); For code examples, see also [R Example Code](#), [Matlab Example Code](#), and [Stata Example Code](#); For intro stat with R, see [Intro Statistics for Undergraduates](#), and intro Math with Matlab, see [Intro Mathematics for Economists](#). See [here](#) for all of [Fan](#)'s public repositories.

The site is built using [Bookdown](#) ([Xie, 2020](#)).

Please contact [FanWangEcon](#) for issues or problems.

Chapter 1

Summarize Policy and Value

1.1 FF_SUMM_ND_ARRAY Examples

Go back to [fan's MEconTools Toolbox \(bookdown\)](#), [Matlab Code Examples Repository \(bookdown\)](#), or [Math for Econ with Matlab Repository \(bookdown\)](#).

This is the example vignette for function: [ff_summ_nd_array](#) from the [MEconTools Package](#). This function summarizes policy and value functions over states.

1.1.1 Test FF_SUMM_ND_ARRAY Defaults

Call the function with defaults.

```
ff_summ_nd_array();
```

```
xxx Summ over (a,z), condi age as cols, kids/marriage as rows xxxxxxxxxxxxxxxxxxxxxxxxxxxx
    group   marry   kids   mn_age_18   mn_age_19   mn_age_20   mn_age_21   cv_age_18   cv_ag
-----
    1       0       1       0.59079    0.53324    0.55055    0.48708    1.9219    1.81
    2       1       1       0.49876    0.5033    0.48682    0.45402    1.7356    1.59
    3       0       2       0.50857    0.4829    0.49712    0.52998    1.7159    1.67
    4       1       2       0.45619    0.51721    0.50414    0.56312    1.6098    1.77
    5       0       3       0.52992    0.56536    0.41866    0.50231    1.8123    1.89
    6       1       3       0.53958    0.54057    0.52793    0.4703    1.8546    1.89
    7       0       4       0.46439    0.49755    0.52478    0.55786    1.5849    1.81
    8       1       4       0.4126     0.48144    0.47836    0.48858    1.4588    1.60
```

1.1.2 Test FF_SUMM_ND_ARRAY with Random 2 Dimensional Matrix

Summarize over 6 dimensional array, iteratively change how many dimensions to group over.

First, generate matrix:

```
st_title = "Random 2D dimensional Array Testing Summarizing";
rng(123)
mn_polval = rand(5,4);
bl_print_table = true;
ar_st_stats = ["mean"];
cl_mp_datasetdesc = {};
cl_mp_datasetdesc{1} = containers.Map({'name', 'labval'}, ...
    {'a', linspace(0,1,size(mn_polval,1))});
cl_mp_datasetdesc{2} = containers.Map({'name', 'labval'}, ...
    {'z', linspace(-1,1,size(mn_polval,2))});
disp(mn_polval);
```

0.6965	0.4231	0.3432	0.7380
0.2861	0.9808	0.7290	0.1825
0.2269	0.6848	0.4386	0.1755
0.5513	0.4809	0.0597	0.5316
0.7195	0.3921	0.3980	0.5318

Second, show the entire matrix (no labels):

```
it_aggd = 0;
bl_row = 1;
ff_summ_nd_array(st_title, mn_polval, bl_print_table, ar_st_stats, it_aggd, bl_row);
```

```
xxx Random 2D dimensional Array Testing Summarizing xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
group   vardim2   mn_vardim1_1   mn_vardim1_2   mn_vardim1_3   mn_vardim1_4   mn_vardim1_5
-----
1         1       0.69647       0.28614       0.22685       0.55131       0.71947
2         2       0.42311       0.98076       0.68483       0.48093       0.39212
3         3       0.34318       0.72905       0.43857       0.059678      0.39804
4         4       0.738        0.18249       0.17545       0.53155       0.53183
```

Third, rotate row and column, and now with labels:

```
it_aggd = 0;
bl_row = 1;
ar_permute = [2,1];
ff_summ_nd_array(st_title, mn_polval, bl_print_table, ar_st_stats, it_aggd, bl_row, ...
    cl_mp_datasetdesc, ar_permute);
```

```
xxx Random 2D dimensional Array Testing Summarizing xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
group   a   mn_z__1   mn_z__0_33333   mn_z_0_33333   mn_z_1
-----
1         0   0.69647   0.42311       0.34318       0.738
2       0.25  0.28614   0.98076       0.72905       0.18249
3       0.5   0.22685   0.68483       0.43857       0.17545
4       0.75  0.55131   0.48093       0.059678      0.53155
5         1   0.71947   0.39212       0.39804       0.53183
```

Fourth, dimension one as columns, average over dim 2:

```
it_aggd = 1;
bl_row = 1;
ff_summ_nd_array(st_title, mn_polval, bl_print_table, ar_st_stats, it_aggd, bl_row, ...
    cl_mp_datasetdesc);
```

```
xxx Random 2D dimensional Array Testing Summarizing xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
group   x   mn_z__1   mn_z__0_33333   mn_z_0_33333   mn_z_1
-----
1         1   0.49605   0.59235       0.3937       0.43186
```

Fifth, dimension one as rows, average over dim 2:

```
it_aggd = 1;
bl_row = 0;
ff_summ_nd_array(st_title, mn_polval, bl_print_table, ar_st_stats, it_aggd, bl_row, ...
    cl_mp_datasetdesc);
```

```
xxx Random 2D dimensional Array Testing Summarizing xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
group   z   mean   std   coefvari   min   max
```


-----	-----	-----	-----	-----	-----	-----
1	-1	0.49605	0.22895	2.1666	0.22685	0.71947
2	-0.33333	0.59235	0.24524	2.4154	0.39212	0.98076
3	0.33333	0.3937	0.23907	1.6468	0.059678	0.72905
4	1	0.43186	0.24575	1.7573	0.17545	0.738

Sixth, dimension two as rows, average over dim 1:

```
ar_permute = [2,1];
it_aggd = 1;
bl_row = 0;
ff_summ_nd_array(st_title, mn_polval, bl_print_table, ar_st_stats, it_aggd, bl_row, ...
    cl_mp_datasetdesc, ar_permute);
```

xxx	Random 2D dimensional Array Testing Summarizing					xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
group	a	mean	std	coefvari	min	max
-----	----	-----	-----	-----	-----	-----
1	0	0.55019	0.19636	2.8019	0.34318	0.738
2	0.25	0.54461	0.37514	1.4518	0.18249	0.98076
3	0.5	0.38143	0.23212	1.6432	0.17545	0.68483
4	0.75	0.40587	0.23269	1.7443	0.059678	0.55131
5	1	0.51036	0.15361	3.3226	0.39212	0.71947

1.1.3 Test FF_SUMM_ND_ARRAY with Random 6 Dimensional Matrix

Summarize over 6 dimensional array, iteratively change how many dimensions to group over.

First, generate matrix:

```
st_title = "Random ND dimensional Array Testing Summarizing";
rng(123)
mn_polval = rand(8,7,6,5,4,3);
bl_print_table = true;
ar_st_stats = ["mean"];
```

Second, summarize over the first four dimensions, row group others:

```
it_aggd = 4;
bl_row = 0;
ff_summ_nd_array(st_title, mn_polval, bl_print_table, ar_st_stats, it_aggd, bl_row);
```

xxx	Random ND dimensional Array Testing Summarizing					xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	
group	vardim5	vardim6	mean	std	coefvari	min	max
-----	-----	-----	-----	-----	-----	-----	-----
1	1	1	0.49808	0.29255	1.7026	8.1888e-05	0.99964
2	2	1	0.50128	0.28968	1.7305	6.7838e-05	0.99936
3	3	1	0.49491	0.28851	1.7154	0.00091373	0.99989
4	4	1	0.50232	0.28154	1.7842	0.00012471	0.99731
5	1	2	0.4994	0.2911	1.7156	0.00029749	0.99938
6	2	2	0.49453	0.28634	1.7271	0.00027113	0.9992
7	3	2	0.49559	0.28682	1.7279	0.00035994	0.99936
8	4	2	0.48835	0.29032	1.6821	0.00096259	0.99896
9	1	3	0.51819	0.29111	1.7801	0.0010616	0.99951
10	2	3	0.50874	0.28458	1.7877	0.001884	0.99965
11	3	3	0.49898	0.2891	1.726	0.0019192	0.99945
12	4	3	0.50169	0.2877	1.7438	0.00016871	0.99963

Third, summarize over the first four dimensions, column group 5th, and row group others:

```

it_aggd = 4;
bl_row = 1;
ff_summ_nd_array(st_title, mn_polval, bl_print_table, ar_st_stats, it_aggd, bl_row);

```

```

xxx Random ND dimensional Array Testing Summarizing xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  group   vardim6   mn_vardim5_1   mn_vardim5_2   mn_vardim5_3   mn_vardim5_4
  -----
    1      1      0.49808      0.50128      0.49491      0.50232
    2      2      0.4994      0.49453      0.49559      0.48835
    3      3      0.51819      0.50874      0.49898      0.50169

```

Fourth, summarize over the first five dimensions, column group 6th, no row groups:

```

it_aggd = 5;
bl_row = 1;
ff_summ_nd_array(st_title, mn_polval, bl_print_table, ar_st_stats, it_aggd, bl_row);

```

```

xxx Random ND dimensional Array Testing Summarizing xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  group   x   mn_vardim6_1   mn_vardim6_2   mn_vardim6_3
  -----
    1      1   0.49915      0.49447      0.5069

```

Fifth, summarize over all six dimensions, summary statistics over the entire dataframe:

```

it_aggd = 6;
bl_row = 0;
ff_summ_nd_array(st_title, mn_polval, bl_print_table, ar_st_stats, it_aggd, bl_row);

```

```

xxx Random ND dimensional Array Testing Summarizing xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  group   x   mean      std      coefvari      min      max
  -----
    1      1   0.50017   0.28831   1.7349   6.7838e-05   0.99989

```

1.1.4 Test FF_SUMM_ND_ARRAY with Random 7 Dimensional Matrix with All Parameters

Given a random seven dimensional matrix, average over the 2nd, 4th and 5th dimensionals. Show as row groups the 3, 6 and 7th dimensions, and row groups the 1st dimension.

```

st_title = "avg VALUE 2+4+5th dims. groups 3+6+7th dims, and row groups the 1st dim.";
rng(123)
mn_polval = rand(3,10,2,10,10,2,3);
ar_permute = [2,4,5,1,3,6,7];
bl_print_table = true;
ar_st_stats = ["mean", "coefvari"];
it_aggd = 3; % mean over 3 dims
bl_row = 1; % one var for row group
cl_mp_datasetdesc = {};
cl_mp_datasetdesc{1} = containers.Map({'name', 'labval'}, ...
    {'age', [18, 19, 20]});
cl_mp_datasetdesc{2} = containers.Map({'name', 'labval'}, ...
    {'savings', linspace(0,1,10)});
cl_mp_datasetdesc{3} = containers.Map({'name', 'labval'}, ...
    {'borrsave', [-1,+1]});
cl_mp_datasetdesc{4} = containers.Map({'name', 'labval'}, ...
    {'shocka', linspace(-5,5,10)});
cl_mp_datasetdesc{5} = containers.Map({'name', 'labval'}, ...
    {'shockb', linspace(-5,5,10)});

```

```

cl_mp_datasetdesc{6} = containers.Map({'name', 'labval'}, ...
    {'marry', [0,1]});
cl_mp_datasetdesc{7} = containers.Map({'name', 'labval'}, ...
    {'region', [1,2,3]});
% call function
ff_summ_nd_array(st_title, mn_polval, bl_print_table, ar_st_stats, it_aggd, bl_row, cl_mp_datasetdes

```

```

xxx  avg VALUE 2+4+5th dims. groups 3+6+7th dims, and row groups the 1st dim. xxxxxxxxxxxxxxxxxxxxxxxx
      group      borrsave      marry      region      mn_age_18      mn_age_19      mn_age_20      cv_age_18      cv_a
      -----      -
      1          -1          0          1          0.50503      0.50389      0.49788      1.7607      1.7
      2           1          0          1          0.4829      0.50795      0.49205      1.6566      1.7
      3          -1          1          1          0.48123      0.50734      0.50109      1.6608      1.7
      4           1          1          1          0.49987      0.49852      0.49519      1.756      1.7
      5          -1          0          2          0.49859      0.50866      0.51752      1.7314      1.7
      6           1          0          2          0.50451      0.49802      0.50439      1.7347      1.
      7          -1          1          2          0.50967      0.49651      0.50556      1.7811      1.
      8           1          1          2          0.50209      0.49224      0.50252      1.7445      1.7
      9          -1          0          3          0.48885      0.49229      0.49692      1.7025      1.7
     10           1          0          3          0.49534      0.50183      0.50266      1.74      1.7
     11          -1          1          3          0.50312      0.50535      0.48959      1.7147      1.7
     12           1          1          3          0.51204      0.49998      0.50738      1.7919      1.7

```


Chapter 2

Distributional Analysis

2.1 FF_SIMU_STATS Examples

Go back to [fan's MEconTools Toolbox \(bookdown\)](#), [Matlab Code Examples Repository \(bookdown\)](#), or [Math for Econ with Matlab Repository \(bookdown\)](#).

This is the example vignette for function: `ff_simu_stats` from the **MEconTools Package**. This is a gate-way function that computes mean, percentiles, covariance etc between several variables.

2.1.1 Test FF_SIMU_STATS Defaults

Call the function with defaults.

```
ff_simu_stats();
```

```
xxx tb_outcomes: all stats xxx
OriginalVariableNames    cl_mt_pol_a    cl_mt_pol_c
-----
{'mean'                  }    -0.11081      8.8423
{'sd'                    }      4.1239      6.5845
{'coefofvar'             }    -37.215      0.74466
{'min'                   }      -7          -6.3772
{'max'                   }       9          21.786
{'pYis0'                 }    0.064259      0
{'pYls0'                 }    0.54867      0.027329
{'pYgr0'                 }    0.38707      0.97267
{'pYisMINY'              }    0.051764      0.015232
{'pYisMAXY'              }    0.027329      0.046484
{'p1'                    }      -7          -6.3772
{'p10'                   }      -6          0.27238
{'p25'                   }      -3          5.2138
{'p50'                   }      -1          6.5321
{'p75'                   }       3          13.799
{'p90'                   }       5          16.887
{'p99'                   }       9          21.786
{'fl_cov_cl_mt_pol_a'}    17.007      -22.084
{'fl_cor_cl_mt_pol_a'}     1          -0.81327
{'fl_cov_cl_mt_pol_c'}   -22.084      43.356
{'fl_cor_cl_mt_pol_c'}   -0.81327      1
{'fracByP1'              }    3.2699      -0.010985
{'fracByP10'             }    5.9889      -0.013362
{'fracByP25'             }    14.165      0.041007
{'fracByP50'             }    16.208      0.1893
```

{'fracByP75'}	}	12.702	0.59539
{'fracByP90'}	}	6.6611	0.8307
{'fracByP99'}	}	1	1

2.1.2 Test FF_SIMU_STATS Four States-Points Matrix

Over some (a,z) states that is 3 by 3, c matrix, generate all stats

```
% Set Parameters
mt_x_of_s = [1, 2, 3.0;...
            3, 1, 1.5;...
            4, 3, 2.0];
mt_y_of_s = [2, -10, 9.0;...
            5, 1.1, 3.0;...
            1, 3, -1.5];
mt_z_of_s = [1.1, 2, 3.3;...
            2.3, 1, 1.5;...
            4, 2.5, 2.0];
mp_cl_mt_xyz_of_s = containers.Map('KeyType','char', 'ValueType','any');
mp_cl_mt_xyz_of_s('cl_mt_x_of_s') = {mt_x_of_s, zeros(1)};
mp_cl_mt_xyz_of_s('cl_mt_y_of_s') = {mt_y_of_s, zeros(1)};
mp_cl_mt_xyz_of_s('cl_mt_z_of_s') = {mt_z_of_s, zeros(1)};
mp_cl_mt_xyz_of_s('ar_st_y_name') = ["cl_mt_x_of_s", "cl_mt_y_of_s", "cl_mt_z_of_s"];
% Mass
rng(123);
mt_f_of_s = rand(size(mt_x_of_s));
mt_f_of_s = mt_f_of_s/sum(mt_f_of_s, 'all');
% Call Function
mp_cl_mt_xyz_of_s_out = ff_simu_stats(mt_f_of_s, mp_cl_mt_xyz_of_s);
```

```
xxx tb_outcomes: all stats xxx
OriginalVariableNames    cl_mt_x_of_s    cl_mt_y_of_s    cl_mt_z_of_s
-----
{'mean'                  }          2.0763          1.9323          2.0668
{'sd'                    }          0.9071          5.2239          0.9042
{'coefofvar'             }          0.43688         2.7034          0.43749
{'min'                   }          1             -10             1
{'max'                   }          4              9              4
{'pYis0'                 }          0              0              0
{'pYls0'                 }          0             0.20441         0
{'pYgr0'                 }          1             0.79559         1
{'pYisMINY'              }          0.28039         0.10917         0.14247
{'pYisMAXY'              }          0.044922        0.19422         0.044922
{'p1'                    }          1             -10             1
{'p10'                   }          1             -10             1
{'p25'                   }          1             1.1             1.1
{'p50'                   }          2              2              2
{'p75'                   }          3              5              2.5
{'p90'                   }          3              9              3.3
{'p99'                   }          4              9              4
{'fl_cov_cl_mt_x_of_s'}    0.82282         1.589           0.78646
{'fl_cor_cl_mt_x_of_s'}    1              0.33534         0.95887
{'fl_cov_cl_mt_y_of_s'}    1.589          27.289          1.8353
{'fl_cor_cl_mt_y_of_s'}    0.33534         1              0.38856
{'fl_cov_cl_mt_z_of_s'}    0.78646         1.8353          0.81758
{'fl_cor_cl_mt_z_of_s'}    0.95887         0.38856         1
{'fracByP1'              }          0.13504        -0.56498        0.068934
{'fracByP10'             }          0.13504        -0.56498        0.068934
```

{'fracByP25'}	}	0.13504	-0.53456	0.14234
{'fracByP50'}	}	0.42991	-0.39181	0.43856
{'fracByP75'}	}	0.91346	0.095425	0.60296
{'fracByP90'}	}	0.91346	1	0.91306
{'fracByP99'}	}	1	1	1

2.1.3 Test FF_SIMU_STATS Four States-Points Matrix Single Column Inputs

Same as before, but now inputs are single column, should have identical results:

```
% Array Inputs
mp_cl_ar_xyz_of_s = containers.Map('KeyType','char', 'ValueType','any');
mp_cl_mt_xyz_of_s('cl_mt_x_of_s') = {mt_x_of_s(:), zeros(1)};
mp_cl_mt_xyz_of_s('cl_mt_y_of_s') = {mt_y_of_s(:), zeros(1)};
mp_cl_mt_xyz_of_s('cl_mt_z_of_s') = {mt_z_of_s(:), zeros(1)};
mp_cl_mt_xyz_of_s('ar_st_y_name') = ["cl_mt_x_of_s", "cl_mt_y_of_s", "cl_mt_z_of_s"];
% Call Function
mp_cl_mt_xyz_of_s_out = ff_simu_stats(mt_f_of_s(:), mp_cl_mt_xyz_of_s);
```

```
xxx tb_outcomes: all stats xxx
```

OriginalVariableNames	cl_mt_x_of_s	cl_mt_y_of_s	cl_mt_z_of_s
-----	-----	-----	-----
{'mean'}	2.0763	1.9323	2.0668
{'sd'}	0.9071	5.2239	0.9042
{'coefofvar'}	0.43688	2.7034	0.43749
{'min'}	1	-10	1
{'max'}	4	9	4
{'pYis0'}	0	0	0
{'pYls0'}	0	0.20441	0
{'pYgr0'}	1	0.79559	1
{'pYisMINY'}	0.28039	0.10917	0.14247
{'pYisMAXY'}	0.044922	0.19422	0.044922
{'p1'}	1	-10	1
{'p10'}	1	-10	1
{'p25'}	1	1.1	1.1
{'p50'}	2	2	2
{'p75'}	3	5	2.5
{'p90'}	3	9	3.3
{'p99'}	4	9	4
{'fl_cov_cl_mt_x_of_s'}	0.82282	1.589	0.78646
{'fl_cor_cl_mt_x_of_s'}	1	0.33534	0.95887
{'fl_cov_cl_mt_y_of_s'}	1.589	27.289	1.8353
{'fl_cor_cl_mt_y_of_s'}	0.33534	1	0.38856
{'fl_cov_cl_mt_z_of_s'}	0.78646	1.8353	0.81758
{'fl_cor_cl_mt_z_of_s'}	0.95887	0.38856	1
{'fracByP1'}	0.13504	-0.56498	0.068934
{'fracByP10'}	0.13504	-0.56498	0.068934
{'fracByP25'}	0.13504	-0.53456	0.14234
{'fracByP50'}	0.42991	-0.39181	0.43856
{'fracByP75'}	0.91346	0.095425	0.60296
{'fracByP90'}	0.91346	1	0.91306
{'fracByP99'}	1	1	1

2.1.4 Test FF_SIMU_STATS Print Many Details

The Same As before, but now control which percentiles and other details to display.

```

% Array Inputs
mp_cl_ar_xyz_of_s = containers.Map('KeyType','char', 'ValueType','any');
mp_cl_ar_xyz_of_s('cl_ar_x_of_s') = {mt_x_of_s(:), zeros(1)};
mp_cl_ar_xyz_of_s('cl_ar_z_of_s') = {mt_z_of_s(:), zeros(1)};
mp_cl_ar_xyz_of_s('ar_st_y_name') = ["cl_ar_x_of_s", "cl_ar_z_of_s"];

% controls
mp_support = containers.Map('KeyType','char', 'ValueType','any');
mp_support('bl_display_detail') = false;
mp_support('bl_display_final') = true;
mp_support('bl_display_drvm2outcomes') = false;
mp_support('ar_fl_percentiles') = [25 50 75];
mp_support('bl_display_drvstats') = true;
mp_support('bl_display_drvm2covcor') = false;

% Call Function
mp_cl_mt_xyz_of_s_out = ff_simu_stats(mt_f_of_s(:), mp_cl_ar_xyz_of_s, mp_support);

-----
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Summary Statistics for: cl_ar_x_of_s
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
-----

fl_choice_mean
    2.0763

fl_choice_sd
    0.9071

fl_choice_coefofvar
    0.4369

fl_choice_prob_zero
    0

fl_choice_prob_below_zero
    0

fl_choice_prob_above_zero
    1

fl_choice_prob_max
    0.0449

tb_disc_cumu
    cl_ar_x_of_sDiscreteVal    cl_ar_x_of_sDiscreteValProbMass    CDF    cumsumFrac
    -----
            1                0.28039                28.039                0.13504
           1.5                0.13561                41.6                0.23301
            2                0.20441                62.041                0.42991
            3                0.33466                95.508                0.91346
            4                0.044922                100                1

    cl_ar_x_of_sDiscreteVal    cl_ar_x_of_sDiscreteValProbMass    CDF    cumsumFrac
    -----
            1                0.28039                28.039                0.13504

```


1.5	0.13561	41.6	0.23301
2	0.20441	62.041	0.42991
3	0.33466	95.508	0.91346
4	0.044922	100	1

tb_prob_drv percentiles	cl_ar_x_of_sDiscreteValPercentileValues	fracOfSumHeldBelowThisPercentile
-----	-----	-----
25	1	0.13504
50	2	0.42991
75	3	0.91346

 xxx
 Summary Statistics for: cl_ar_z_of_s
 xxx

fl_choice_mean
 2.0668

fl_choice_sd
 0.9042

fl_choice_coefofvar
 0.4375

fl_choice_prob_zero
 0

fl_choice_prob_below_zero
 0

fl_choice_prob_above_zero
 1

fl_choice_prob_max
 0.0449

tb_disc_cumu cl_ar_z_of_sDiscreteVal	cl_ar_z_of_sDiscreteValProbMass	CDF	cumsumFrac
-----	-----	-----	-----
1	0.14247	14.247	0.068934
1.1	0.13792	28.039	0.14234
1.5	0.13561	41.6	0.24076
2	0.20441	62.041	0.43856
2.3	0.056663	67.708	0.50162
2.5	0.083786	76.086	0.60296
3.3	0.19422	95.508	0.91306
4	0.044922	100	1

cl_ar_z_of_sDiscreteVal	cl_ar_z_of_sDiscreteValProbMass	CDF	cumsumFrac
-----	-----	-----	-----
1	0.14247	14.247	0.068934
1.1	0.13792	28.039	0.14234
1.5	0.13561	41.6	0.24076


```
fl_choice_mean
-1.0000
```

```
fl_choice_sd
2.5100
```

```
fl_choice_coefofvar
-2.5100
```

```
fl_choice_prob_zero
0.1416
```

```
fl_choice_prob_below_zero
0.5888
```

```
fl_choice_prob_above_zero
0.2696
```

```
fl_choice_prob_max
2.0589e-16
```

```
tb_disc_cumu
  binomDiscreteVal  binomDiscreteValProbMass  CDF  cumsumFrac
  -----
      -10          2.2539e-05      0.0022539  0.00022539
      -9          0.00028979      0.031233   0.0028335
      -8          0.0018008       0.21132   0.01724
      -7          0.0072034       0.93166   0.067664
      -6          0.020838        3.0155    0.19269
      -5          0.04644         7.6595    0.42489
      -4          0.082928       15.952    0.75661
      -3          0.12185        28.138    1.1222
      -2          0.15014        43.152    1.4224
      -1          0.15729        58.881    1.5797
```

```
  binomDiscreteVal  binomDiscreteValProbMass  CDF  cumsumFrac
  -----
      11          6.0392e-06      100      1
      12          1.0588e-06      100      1
      13          1.5784e-07      100      1
      14          1.973e-08       100      1
      15          2.0293e-09      100      1
      16          1.6725e-10      100      1
      17          1.0619e-11      100      1
      18          4.8762e-13      100      1
      19          1.4412e-14      100      1
      20          2.0589e-16      100      1
```

```
tb_prob_drv
  percentiles  binomDiscreteValPercentileValues  fracOfSumHeldBelowThisPercentile
  -----
      0.1          -8          0.01724
      1           -6          0.19269
      5           -5          0.42489
     10           -4          0.75661
```

15	-4	0.75661
20	-3	1.1222
25	-3	1.1222
35	-2	1.4224
50	-1	1.5797
65	0	1.5797
75	1	1.4694
80	1	1.4694
85	2	1.3197
90	2	1.3197
95	3	1.1865
99	5	1.0412
99.9	7	1.0052

2.2.2 Test FF_DISC_RAND_VAR_STATS 0 and 1 Random Variable

The simplest discrete random variable has two values, zero or one. The probability of zero is 30 percent, and 70 percent is the probability of one.

```
% Parameters
% 1. specify the random variable
st_var_name = 'bernoulli';
ar_choice_unique_sorted = [0, 1];
ar_choice_prob = [0.3, 0.7];
% 2. percentiles of interest
ar_fl_percentiles = [0.1 5 25 50 75 95 99.9];
% 3. print results
bl_display_drvstats = true;
% Call Function
[ds_stats_map] = ff_disc_rand_var_stats(st_var_name, ...
    ar_choice_unique_sorted, ar_choice_prob, ...
    ar_fl_percentiles, bl_display_drvstats);
```

```
-----
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Summary Statistics for: bernoulli
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
-----
```

```
fl_choice_mean
    0.7000
```

```
fl_choice_sd
    0.4583
```

```
fl_choice_coefofvar
    0.6547
```

```
fl_choice_prob_zero
    0.3000
```

```
fl_choice_prob_below_zero
    0
```

```
fl_choice_prob_above_zero
    0.7000
```

```
fl_choice_prob_max
    0.7000
```

```

tb_disc_cumu
  bernoulliDiscreteVal    bernoulliDiscreteValProbMass    CDF    cumsumFrac
  -----
          0                0.3                30          0
          1                0.7               100          1

  bernoulliDiscreteVal    bernoulliDiscreteValProbMass    CDF    cumsumFrac
  -----
          0                0.3                30          0
          1                0.7               100          1

tb_prob_drv
  percentiles    bernoulliDiscreteValPercentileValues    fracOfSumHeldBelowThisPercentile
  -----
    0.1                0                                0
     5                0                                0
    25                0                                0
    50                1                                1
    75                1                                1
    95                1                                1
   99.9                1                                1

```

2.2.3 Test FF_DISC_RAND_VAR_STATS with Poisson

[Poisson random variable](#), with mean equals to ten, summarize over unsymmetric percentiles. Note that the poisson random variable has no upper bound.

```

% Parameters
% 1. specify the random variable
st_var_name = 'poisson';
mu = 10;
ar_choice_unique_sorted = 0:1:50;
ar_choice_prob = poisspdf(ar_choice_unique_sorted, mu);
% 2. percentiles of interest, unsymmetric
ar_fl_percentiles = [0.1 5 10 25 50 90 95 99 99.9 99.99 99.999 99.9999];
% 3. print results
bl_display_drvstats = true;
% Call Function
[ds_stats_map] = ff_disc_rand_var_stats(st_var_name, ...
    ar_choice_unique_sorted, ar_choice_prob, ...
    ar_fl_percentiles, bl_display_drvstats);

-----
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Summary Statistics for: poisson
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
-----

fl_choice_mean
    10

fl_choice_sd
    3.1623

fl_choice_coefofvar
    0.3162

```

```
fl_choice_prob_zero
4.5400e-05
```

```
fl_choice_prob_below_zero
0
```

```
fl_choice_prob_above_zero
1.0000
```

```
fl_choice_prob_max
1.4927e-19
```

```
tb_disc_cumu
```

poissonDiscreteVal	poissonDiscreteValProbMass	CDF	cumsumFrac
-----	-----	-----	-----
0	4.54e-05	0.00454	0
1	0.000454	0.04994	4.54e-05
2	0.00227	0.27694	0.0004994
3	0.0075667	1.0336	0.0027694
4	0.018917	2.9253	0.010336
5	0.037833	6.7086	0.029253
6	0.063055	13.014	0.067086
7	0.090079	22.022	0.13014
8	0.1126	33.282	0.22022
9	0.12511	45.793	0.33282

poissonDiscreteVal	poissonDiscreteValProbMass	CDF	cumsumFrac
-----	-----	---	-----
41	1.3571e-13	100	1
42	3.2313e-14	100	1
43	7.5146e-15	100	1
44	1.7079e-15	100	1
45	3.7953e-16	100	1
46	8.2506e-17	100	1
47	1.7554e-17	100	1
48	3.6572e-18	100	1
49	7.4636e-19	100	1
50	1.4927e-19	100	1

```
tb_prob_drv
```

percentiles	poissonDiscreteValPercentileValues	fracOfSumHeldBelowThisPercentile
-----	-----	-----
0.1	2	0.0004994
5	5	0.029253
10	6	0.067086
25	8	0.22022
50	10	0.45793
90	14	0.86446
95	15	0.91654
99	18	0.98572
99.9	21	0.99841
99.99	24	0.99988
99.999	26	0.99998
100	28	1

```
% Print out full Stored Matrix
% Note that the outputs are single row arrays.
ff_container_map_display(ds_stats_map, 100, 100)

-----
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
CONTAINER NAME: ds_stats_map ND Array (Matrix etc)
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

          i      idx      ndim      numel      rowN      colN      mean      std      coe
          -      ---      ----      -----      ----      ----      -
ar_choice_perc_fracheld      1      1      2      12      1      12      0.62833      0.435      0.6
ar_choice_percentiles      2      2      2      12      1      12      14.75      8.7399      0.5
ar_fl_percentiles      3      3      2      12      1      12      64.499      42.887      0.6

xxx TABLE:ar_choice_perc_fracheld XXXXXXXXXXXXXXXXXXXXXXX
      c1      c2      c3      c4      c5      c6      c7      c8
      -----
r1      0.0004994      0.029253      0.067086      0.22022      0.45793      0.86446      0.91654      0.98572

xxx TABLE:ar_choice_percentiles XXXXXXXXXXXXXXXXXXXXXXX
      c1      c2      c3      c4      c5      c6      c7      c8      c9      c10      c11      c12
      --      --      --      --      --      --      --      --      --      ---      ---      ---
r1      2      5      6      8      10      14      15      18      21      24      26      28

xxx TABLE:ar_fl_percentiles XXXXXXXXXXXXXXXXXXXXXXX
      c1      c2      c3      c4      c5      c6      c7      c8      c9      c10      c11      c12
      ---      --      --      --      --      --      --      --      ---      ---      ---      ---
r1      0.1      5      10      25      50      90      95      99      99.9      99.99      99.999      100

-----
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
CONTAINER NAME: ds_stats_map Scalars
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

          i      idx      value
          --      ---      -----
fl_choice_coefofvar      1      4      0.31623
fl_choice_max      2      5      50
fl_choice_mean      3      6      10
fl_choice_min      4      7      0
fl_choice_prob_above_zero      5      8      0.99995
fl_choice_prob_below_zero      6      9      0
fl_choice_prob_max      7      10      1.4927e-19
fl_choice_prob_min      8      11      4.54e-05
fl_choice_prob_zero      9      12      4.54e-05
fl_choice_sd      10      13      3.1623
```

2.3 FF_DISC_RAND_VAR_MASS2OUTCOMES Examples

Go back to [fan's MEconTools Toolbox \(bookdown\)](#), [Matlab Code Examples Repository \(bookdown\)](#), or [Math for Econ with Matlab Repository \(bookdown\)](#).

This is the example vignette for function: [ff_disc_rand_var_mass2outcomes](#) from the [MEconTools Package](#). This function generates sorted discrete random variable from state-space joint distri-

bution.

2.3.1 Test FF_DISC_RAND_VAR_MASS2OUTCOMES Defaults

Call the function with defaults.

```
ff_disc_rand_var_mass2outcomes();
```

```
INPUT f(a,z): mt_dist_bystates
  0.0289  0.0465  0.0228  0.0036  0.0001
  0.0241  0.0930  0.0857  0.0241  0.0015
  0.0080  0.0744  0.1285  0.0643  0.0074
  0.0013  0.0297  0.0964  0.0857  0.0186
  0.0001  0.0059  0.0361  0.0571  0.0232
  0.0000  0.0005  0.0054  0.0152  0.0116
```

```
INPUT y(a,z): mt_choice_bystates
  -5  -4  -5  -4  -4
  -3  -2  -3  -2  -3
  -1  -1  -1   0   0
   1   1   2   3   1
   4   3   3   4   3
   5   6   5   6   6
```

```
OUTPUT f(y): ar_choice_prob_byY
  0.0518
  0.0502
  0.1113
  0.1171
  0.2109
  0.0717
  0.0497
  0.0964
  0.1510
  0.0572
  0.0054
  0.0273
```

```
OUTPUT f(y,z): mt_choice_prob_byYZ
  0.0289  0  0.0228  0  0
    0  0.0465  0  0.0036  0.0001
  0.0241  0  0.0857  0  0.0015
    0  0.0930  0  0.0241  0
  0.0080  0.0744  0.1285  0  0
    0  0  0  0.0643  0.0074
  0.0013  0.0297  0  0  0.0186
    0  0  0.0964  0  0
    0  0.0059  0.0361  0.0857  0.0232
  0.0001  0  0  0.0571  0
  0.0000  0  0.0054  0  0
    0  0.0005  0  0.0152  0.0116
```

```
OUTPUT f(y,a): mt_choice_prob_byYA
  0.0518  0  0  0  0  0
  0.0502  0  0  0  0  0
    0  0.1113  0  0  0  0
    0  0.1171  0  0  0  0
    0  0  0.2109  0  0  0
    0  0  0.0717  0  0  0
```


0	0	0	0.0497	0	0
0	0	0	0.0964	0	0
0	0	0	0.0857	0.0653	0
0	0	0	0	0.0572	0
0	0	0	0	0	0.0054
0	0	0	0	0	0.0273

OUTPUT f(y) and y in table: tb_choice_drv_cur_byY

binomtestOutcomes	probMassFunction
-------------------	------------------

-5	0.051764
-4	0.050217
-3	0.11126
-2	0.11706
-1	0.21092
0	0.071696
1	0.049682
2	0.096388
3	0.15102
4	0.057231
5	0.0054256
6	0.027329

2.3.2 Test FF_DISC_RAND_VAR_MASS2OUTCOMES Four States-Points

Over some (a,z) states that is 2 by 2, matrix or vectorized inputs identical results.

```
% Set Parameters
st_y_name = 'consumption';
% consumption matrix: c(a,z)
mt_c_of_s = [1,2;3,1];
% stationary mass over assets adn shocks: f(a,z)
mt_f_of_s = rand(size(mt_c_of_s));
mt_f_of_s = mt_f_of_s/sum(mt_f_of_s, 'all');
% Call Function
[ar_f_of_y, ar_y_unique_sorted] = ...
    ff_disc_rand_var_mass2outcomes(st_y_name, mt_c_of_s, mt_f_of_s);
% print
disp([ar_f_of_y ar_y_unique_sorted]);

0.4039    1.0000
0.2971    2.0000
0.2990    3.0000
```

Same as before, but now inputs are single column:

```
% Call Function
[ar_f_of_y, ar_y_unique_sorted] = ...
    ff_disc_rand_var_mass2outcomes(st_y_name, mt_c_of_s(:), mt_f_of_s);
disp([ar_f_of_y ar_y_unique_sorted]);

0.4039    1.0000
0.2971    2.0000
0.2990    3.0000
```

2.3.3 Test FF_DISC_RAND_VAR_MASS2OUTCOMES Conditional Mass Outputs

Same inputs as before, but now, also output additional conditional statistics, $f(y, a)$, where a is the row state variable for $f(a, z)$. For conditional statistics, must provide matrix based inputs.

```
% Set Parameters
st_y_name = 'consumption';
% consumption matrix: c(a,z)
mt_c_of_s = [1,2,0.5;
             3,1,2.0];
% stationary mass over assets and shocks: f(a,z)
mt_f_of_s = rand(size(mt_c_of_s));
mt_f_of_s = mt_f_of_s/sum(mt_f_of_s, 'all');
% Call Function
[ar_f_of_y, ar_y_unique_sorted, mt_f_of_y_srow, mt_f_of_y_scol] = ...
    ff_disc_rand_var_mass2outcomes(st_y_name, mt_c_of_s, mt_f_of_s);
% print
disp([ar_f_of_y ar_y_unique_sorted]);

    0.2695    0.5000
    0.3765    1.0000
    0.2649    2.0000
    0.0891    3.0000

disp(mt_f_of_y_srow);

    0.2695         0
    0.1215    0.2550
    0.1217    0.1432
         0    0.0891

disp(mt_f_of_y_scol);

         0         0    0.2695
    0.1215    0.2550         0
         0    0.1217    0.1432
    0.0891         0         0
```

2.4 FF_DISC_RAND_VAR_MASS2COVCOR Examples

Go back to [fan's MEconTools Toolbox \(bookdown\)](#), [Matlab Code Examples Repository \(bookdown\)](#), or [Math for Econ with Matlab Repository \(bookdown\)](#).

This is the example vignette for function: [ff_disc_rand_var_mass2covcor](#) from the [MEconTools Package](#). This function calculates covariance and correlation based for two discrete random variables.

2.4.1 Test FF_DISC_RAND_VAR_MASS2COVCOR Defaults

Call the function with defaults.

```
ff_disc_rand_var_mass2covcor();

-----
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: covvar_input_map ND Array (Matrix etc)
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

      i      idx      ndim      numel      rowN      colN      mean      std      coefvari
      -      ---      ----      -----      ----      ----      -
-----
```

mt_f_of_s	1	5	2	30	6	5	0.033333	0.035743	1.0723	3.
mt_x_of_s	2	6	2	30	6	5	0.83333	5.3051	6.3661	
mt_y_of_s	3	7	2	30	6	5	8.3259	7.1913	0.86373	

xxx TABLE:mt_f_of_s xxxxxxxxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4	c5
	-----	-----	-----	-----	-----
r1	0.028917	0.046484	0.022848	0.0036146	0.000119
r2	0.024097	0.092967	0.085679	0.024097	0.0014875
r3	0.0080324	0.074374	0.12852	0.064259	0.0074374
r4	0.0013387	0.02975	0.096388	0.085679	0.018593
r5	0.00011156	0.0059499	0.036146	0.057119	0.023242
r6	3.7187e-06	0.00047599	0.0054218	0.015232	0.011621

xxx TABLE:mt_x_of_s xxxxxxxxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4	c5
	--	--	--	--	--
r1	-7	-6	-7	-6	-6
r2	-5	-3	-5	-3	-4
r3	-2	-1	-1	0	-1
r4	2	2	3	4	2
r5	6	5	5	6	5
r6	8	9	7	9	9

xxx TABLE:mt_y_of_s xxxxxxxxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4	c5
	-----	-----	-----	-----	-----
r1	13.231	21.786	18.136	19.35	13.901
r2	9.946	16.887	9.6914	15.71	8.6906
r3	16.255	6.2166	13.799	5.2138	11.641
r4	12.628	2.7525	6.5321	0.27238	13.357
r5	5.8844	4.0352	6.05	0.14102	0.50318
r6	3.5617	-0.72091	5.1855	-6.3772	-4.4805

xx

CONTAINER NAME: covvar_input_map Scalars

xx

	i	idx	value
	-	---	-----
fl_x_mean	1	1	-0.11081
fl_x_sd	2	2	4.1239
fl_y_mean	3	3	8.8423
fl_y_sd	4	4	6.5845

xx

CONTAINER NAME: covvar_output_map ND Array (Matrix etc)

xx

	i	idx	ndim	numel	rowN	colN	mean	std
	-	---	----	-----	----	----	-----	-----
mt_cov_component_weighted	1	1	2	30	6	5	-0.73612	1.0404

mt_x_devi_from_mean	2	2	2	30	6	5	0.94415	5.3051
mt_x_y_multiply	3	3	2	30	6	5	-31.321	36.564
mt_y_devi_from_mean	4	4	2	30	6	5	-0.51644	7.1913

xxx TABLE:mt_cov_component_weighted xxxxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4	c5
-----	-----	-----	-----	-----	-----
r1	-0.87434	-3.5432	-1.4628	-0.22368	-0.0035451
r2	-0.13003	-2.1607	-0.35565	-0.47814	0.00087767
r3	-0.11248	0.17365	-0.56642	-0.025838	-0.018507
r4	0.010697	-0.38241	-0.69273	-3.0184	0.17717
r5	-0.0020165	-0.14618	-0.51584	-3.0371	-0.99056
r6	-0.00015927	-0.041473	-0.14098	-2.1121	-1.4106

xxx TABLE:mt_x_devi_from_mean xxxxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4	c5
-----	-----	-----	-----	-----	-----
r1	-6.8892	-5.8892	-6.8892	-5.8892	-5.8892
r2	-4.8892	-2.8892	-4.8892	-2.8892	-3.8892
r3	-1.8892	-0.88919	-0.88919	0.11081	-0.88919
r4	2.1108	2.1108	3.1108	4.1108	2.1108
r5	6.1108	5.1108	5.1108	6.1108	5.1108
r6	8.1108	9.1108	7.1108	9.1108	9.1108

xxx TABLE:mt_x_y_multiply xxxxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4	c5
-----	-----	-----	-----	-----	-----
r1	-30.237	-76.225	-64.023	-61.882	-29.792
r2	-5.396	-23.242	-4.151	-19.842	0.59004
r3	-14.003	2.3348	-4.4073	-0.40209	-2.4884
r4	7.9905	-12.854	-7.1868	-35.23	9.5287
r5	-18.075	-24.568	-14.271	-53.172	-42.62
r6	-42.83	-87.129	-26.003	-138.66	-121.38

xxx TABLE:mt_y_devi_from_mean xxxxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4	c5
-----	-----	-----	-----	-----	-----
r1	4.389	12.943	9.2933	10.508	5.0587
r2	1.1037	8.0444	0.84902	6.8677	-0.15171
r3	7.4123	-2.6258	4.9566	-3.6286	2.7985
r4	3.7855	-6.0898	-2.3103	-8.57	4.5142
r5	-2.9579	-4.8071	-2.7924	-8.7013	-8.3392
r6	-5.2806	-9.5633	-3.6568	-15.22	-13.323

fl_cov
-22.0835

fl_cor
-0.8133

2.4.2 Test FF_DISC_RAND_VAR_MASS2COVCOR Four States-Points

Over some (a,z) states that is 2 by 2, c matrix, and y matrix, find correlation. Positively related.

% Set Parameters

```

mt_c_of_s = [1,2;3,1];
mt_y_of_s = [2,10;5,1.1];
rng(123);
mt_f_of_s = rand(size(mt_c_of_s));
mt_f_of_s = mt_f_of_s/sum(mt_f_of_s, 'all');
bl_display_drvm2covcor = false;
% Call Function
[fl_cov_xy, fl_cor_xy] = ff_disc_rand_var_mass2covcor(...
    mt_c_of_s, mt_y_of_s, mt_f_of_s, bl_display_drvm2covcor);
display(['cov=' num2str(fl_cov_xy) ',cor=', num2str(fl_cor_xy)]);

cov=1.4446,cor=0.65723

```

Same as before, but now inputs are single column:

```

% Call Function
[fl_cov_xy, fl_cor_xy] = ff_disc_rand_var_mass2covcor(...
    mt_c_of_s(:), mt_y_of_s(:), mt_f_of_s(:), bl_display_drvm2covcor);
display(['cov=' num2str(fl_cov_xy) ',cor=', num2str(fl_cor_xy)]);

cov=1.4446,cor=0.65723

```

2.4.3 Test FF_DISC_RAND_VAR_MASS2COVCOR Two Random Vectors

Generate two random vectors, with random or even mass, correlation should be zero:

```

% Set Parameters
rng(4567);
mt_c_of_s = rand([20,1])*100;
mt_y_of_s = rand([20,1])*100;
mt_f_of_s = rand(size(mt_c_of_s));
mt_f_of_s = mt_f_of_s/sum(mt_f_of_s, 'all');
bl_display_drvm2covcor = false;
% Call Function
[fl_cov_xy, fl_cor_xy] = ff_disc_rand_var_mass2covcor(...
    mt_c_of_s, mt_y_of_s, mt_f_of_s, bl_display_drvm2covcor);
display(['cov=' num2str(fl_cov_xy) ',cor=', num2str(fl_cor_xy)]);

cov=-57.6533,cor=-0.062023

```

2.4.4 Test FF_DISC_RAND_VAR_MASS2COVCOR Provide Mean and SD

Same as above, but now provide means and sd for x and y directly. The results are the same as when mean and sd are calculated inside the function.

```

% Set Parameters
rng(4567);
mt_c_of_s = rand([20,1])*100;
mt_y_of_s = rand([20,1])*100;
mt_f_of_s = rand(size(mt_c_of_s));
mt_f_of_s = mt_f_of_s/sum(mt_f_of_s, 'all');
fl_c_mean = sum(mt_f_of_s.*mt_c_of_s);
fl_c_sd = sqrt(sum(mt_f_of_s.*(mt_c_of_s-fl_c_mean).^2));
fl_y_mean = sum(mt_f_of_s.*mt_y_of_s);
fl_y_sd = sqrt(sum(mt_f_of_s.*(mt_y_of_s-fl_y_mean).^2));
bl_display_drvm2covcor = false;
% Call Function
[fl_cov_xy, fl_cor_xy] = ff_disc_rand_var_mass2covcor(...

```

```
mt_c_of_s, mt_y_of_s, mt_f_of_s, ...  
fl_c_mean, fl_c_sd, ...  
fl_y_mean, fl_y_sd, bl_display_drvm2covcor);  
display(['cov=' num2str(fl_cov_xy) ',cor=', num2str(fl_cor_xy)]);  
  
cov=-57.6533,cor=-0.062023
```

Chapter 3

Graphs

3.1 FF_GRAPH_GRID Examples: X, Y and Color Line Plots

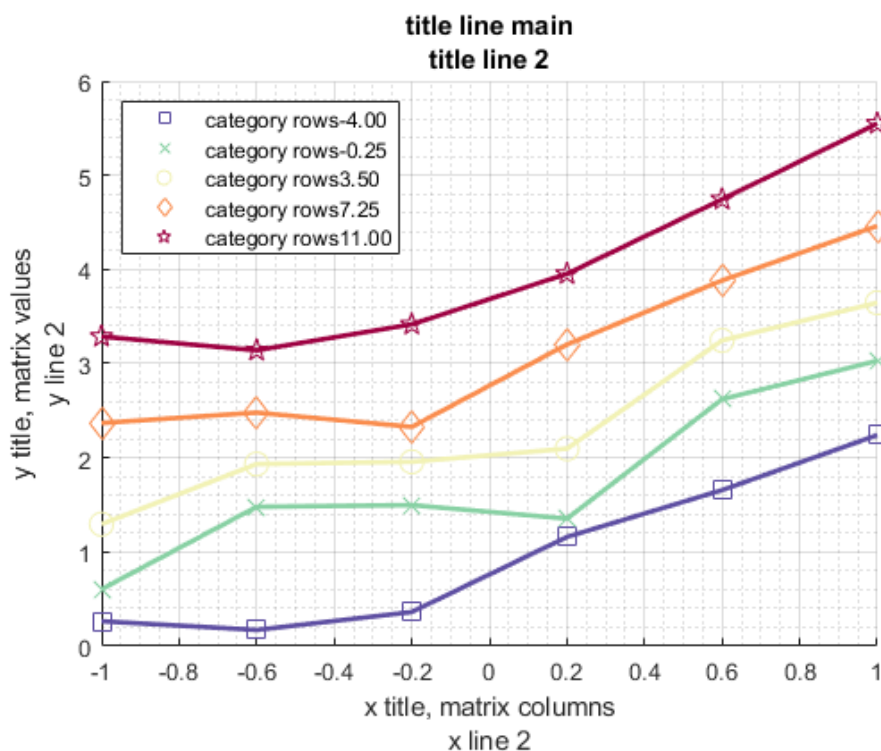
Go back to [fan's MEconTools Toolbox \(bookdown\)](#), [Matlab Code Examples Repository \(bookdown\)](#), or [Math for Econ with Matlab Repository \(bookdown\)](#).

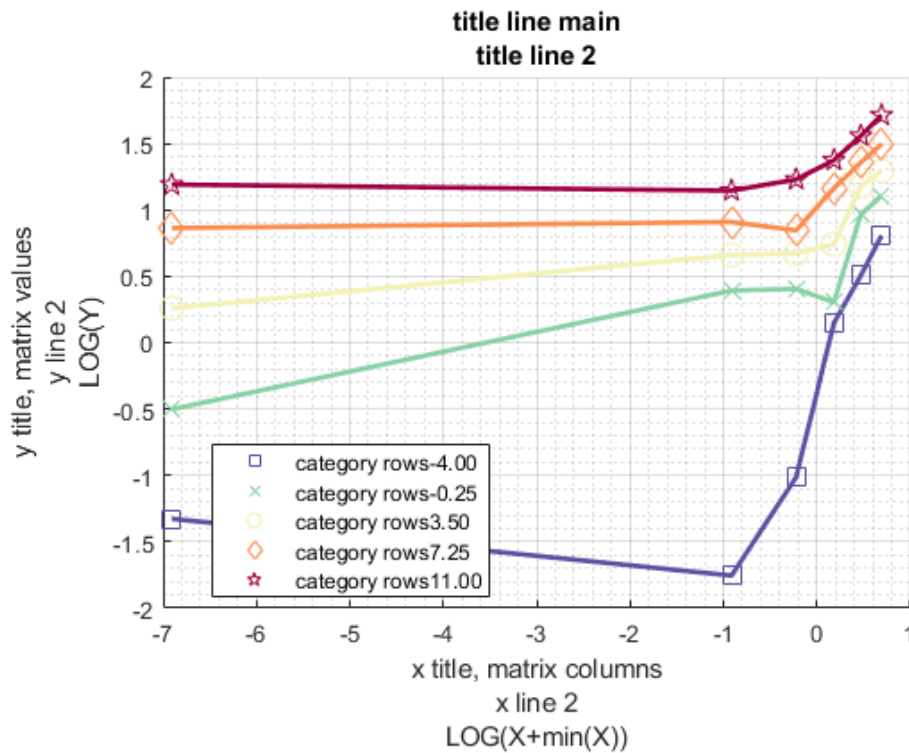
This is the example vignette for function: `ff_graph_grid` from the **MEconTools Package**. This function can graph out value and policy functions given one state vector (x-axis), conditional on other states (line groups). Can handle a few lines (scatter + lines), or many groups (jet spectrum).

3.1.1 Test FF_GRAPH_GRID Defaults

Call the function with defaults.

```
ff_graph_grid();
```

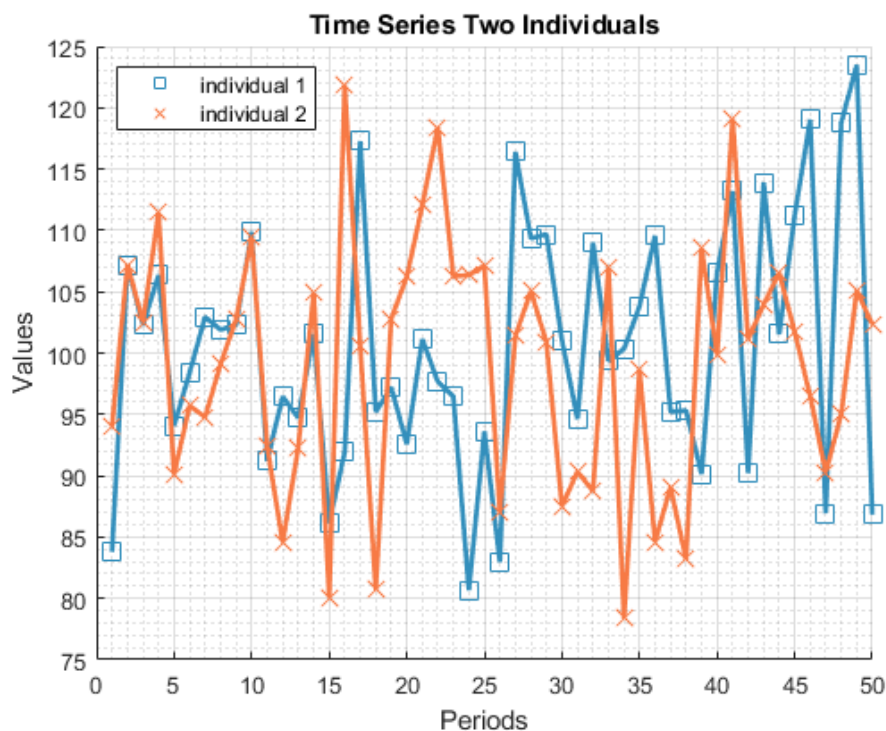




3.1.2 Test FF_GRAPH_GRID Two Random Normal Lines and Labels

There are two autoregressive time series, plot out the time two time series.

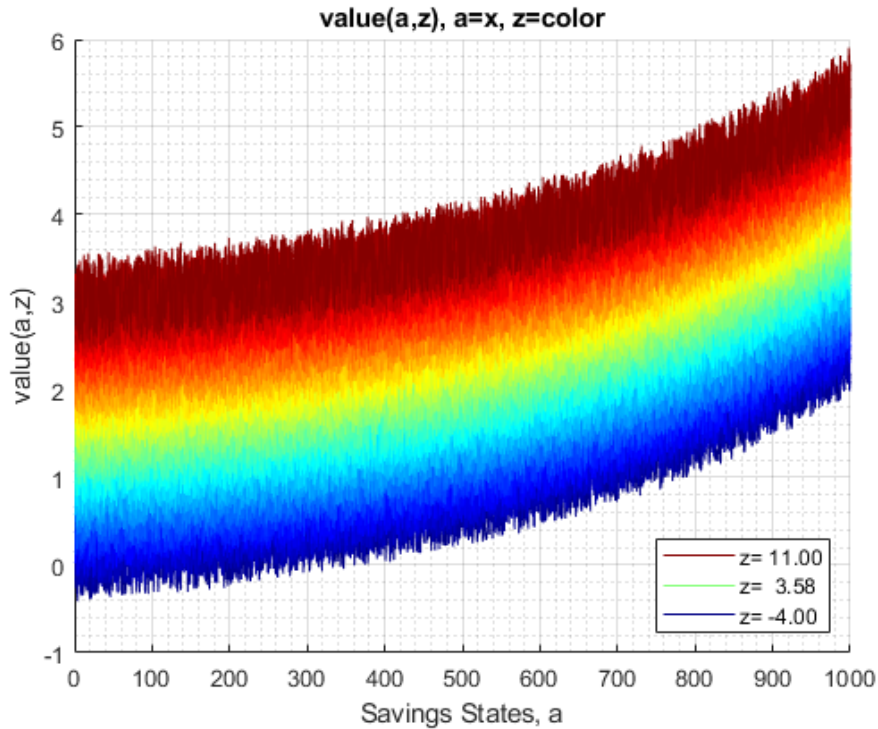
```
% Generate the two time series
rng(456);
mt_value = normrnd(100,10,[2, 50]);
ar_row_grid = ["individual 1", "individual 2"];
ar_col_grid = 1:50;
mp_support_graph = containers.Map('KeyType', 'char', 'ValueType', 'any');
mp_support_graph('cl_st_graph_title') = {'Time Series Two Individuals'};
mp_support_graph('cl_st_ytitle') = {'Values'};
mp_support_graph('cl_st_xtitle') = {'Periods'};
mp_support_graph('bl_graph_logy') = false; % do not log
ff_graph_grid(mt_value, ar_row_grid, ar_col_grid, mp_support_graph);
```

3.1.3 Test FF_GRAPH_GRID Many Lines

Plot many lines, with auto legend.

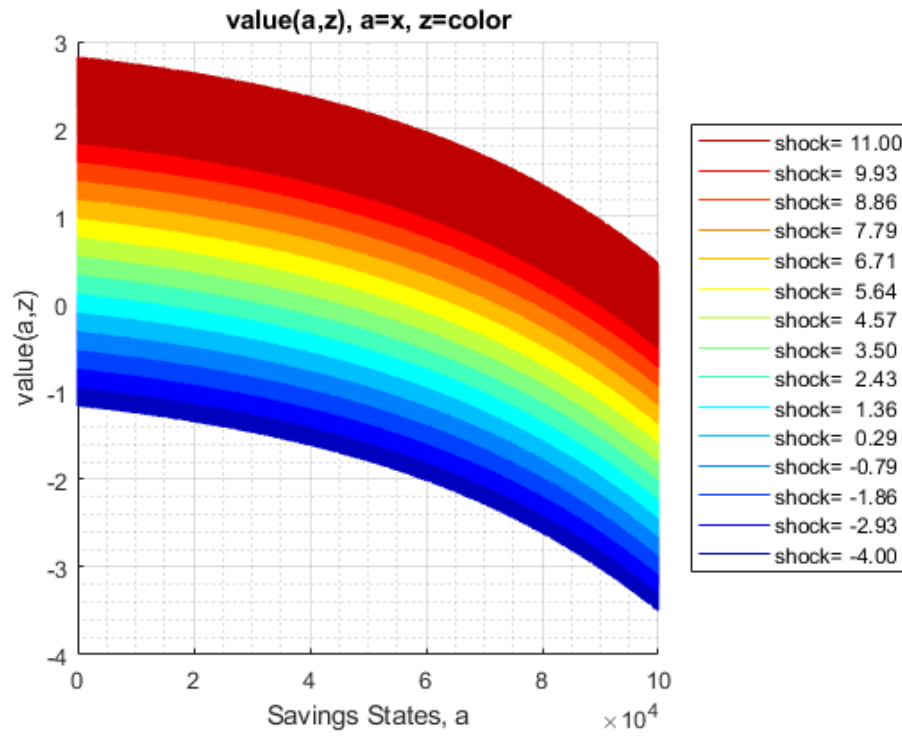
```
% Generate some Data
rng(456);
ar_row_grid = linspace(-4, 11, 100);
ar_col_grid = linspace(-1, 1, 1000);
rng(123);
mt_value = 0.2*ar_row_grid + exp(ar_col_grid) + rand([length(ar_row_grid), length(ar_col_grid)]);
% container map settings
mp_support_graph = containers.Map('KeyType', 'char', 'ValueType', 'any');
mp_support_graph('cl_st_graph_title') = {'value(a,z), a=x, z=color'};
mp_support_graph('cl_st_ytitle') = {'value(a,z)'};
mp_support_graph('cl_st_xtitle') = {'Savings States, a'};
mp_support_graph('st_legend_loc') = 'southeast';
mp_support_graph('bl_graph_logy') = false; % do not log
mp_support_graph('st_rowvar_name') = 'z=';
mp_support_graph('it_legend_select') = 3; % how many shock legends to show
mp_support_graph('st_rounding') = '6.2f'; % format shock legend
% Call function
ff_graph_grid(mt_value, ar_row_grid, ar_col_grid, mp_support_graph);
```



3.1.4 Test FF_GRAPH_GRID Many Lines Legend Exogenous

Plot many lines, exogenously set legend

```
% Generate the two time series
rng(456);
ar_row_grid = linspace(-4, 11, 15);
ar_col_grid = linspace(-1, 1, 100000);
rng(123);
mt_value = 0.2*ar_row_grid' - exp(ar_col_grid) + rand([length(ar_row_grid), length(ar_col_grid)]);
% setting shock vector name exogenously here
ar_row_grid = string(num2str(ar_row_grid', "shock=%6.2f"));
% container map settings
mp_support_graph = containers.Map('KeyType', 'char', 'ValueType', 'any');
mp_support_graph('cl_st_graph_title') = {'value(a,z), a=x, z=color'};
mp_support_graph('cl_st_ytitle') = {'value(a,z)'};
mp_support_graph('cl_st_xtitle') = {'Savings States, a'};
mp_support_graph('st_legend_loc') = 'eastoutside';
mp_support_graph('bl_graph_logy') = false; % do not log
mp_support_graph('it_legend_select') = 15;
% Call function
ff_graph_grid(mt_value, ar_row_grid, ar_col_grid, mp_support_graph);
```



Chapter 4

Support Tools

4.1 FF_CONTAINER_MAP_DISPLAY Examples

Go back to [fan's MEconTools Toolbox \(bookdown\)](#), [Matlab Code Examples Repository \(bookdown\)](#), or [Math for Econ with Matlab Repository \(bookdown\)](#).

This is the example vignette for function: `ff_container_map_display` from the [MEconTools Package](#). This function summarizes statistics of matrixes stored in a container map, as well as scalar, string, function and other values stored in container maps.

4.1.1 Test FF_CONTAINER_MAP_DISPLAY Defaults

Call the function with defaults.

```
ff_container_map_display();
```

```
-----
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ND Array (Matrix etc)
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
          i      idx      ndim      numel      rowN      colN      mean      std      coefvari
          --      ---      ----      -----      ----      ----      -
mat_1          1       7       2          12         3         4      0.54285     0.2232     0.41115
mat_2          2       8       2        2650        50        53      0.49559     0.29232    0.58985
mat_2_boolean  3       9       2        2650        50        53      0.51358     0.49991    0.97337
mat_3          4      10       2         4         2         2      0.45277     0.45111    0.99635
tensor_1       5      15       3         16         2         8      0.45652     0.27787    0.60867
tensor_2       6      16       3         75         3        25      0.53593     0.29044    0.54194
tensor_3       7      17       2         4         1         4      0.42315     0.37389    0.88359
tesseract_1    8      18       4         72         3        24      0.47669     0.26374    0.55327
tesseract_2    9      19       4         20         2        10      0.42096     0.28981    0.68846
tesseract_bl_3 10     20       4         10         1        10       0.3         0.48305    1.6102

xxx TABLE:mat_1 xxxxxxxxxxxxxxxxxxxxxxx
          c1          c2          c3          c4
          -----
r1      0.69647      0.55131      0.98076      0.39212
r2      0.28614      0.71947      0.68483      0.34318
r3      0.22685      0.42311      0.48093      0.72905

xxx TABLE:mat_2 xxxxxxxxxxxxxxxxxxxxxxx
          c1          c2          c3          c4          c50          c51          c52          c53
```

```

-----
r1      0.43857      0.6249      0.17108      0.56564      0.072152      0.67855      0.61667      0.540
r2      0.059678     0.67469     0.82911     0.084904     0.63289     0.27236     0.32528     0.249
r3      0.39804      0.84234     0.33867     0.58267     0.046367     0.44513     0.075047     0.78
r4       0.738      0.083195     0.55237     0.81484     0.50561     0.11117     0.59532     0.356
r5      0.18249      0.76368     0.57855     0.33707     0.10653     0.028681     0.7435     0.918
r46     0.6813      0.55326     0.88786     0.69983     0.83758     0.16382     0.74191     0.0656
r47     0.87546     0.85445     0.69631     0.66117     0.97069     0.79092     0.42466     0.787
r48     0.51042     0.38484     0.44033     0.049097     0.017768     0.33302     0.24401     0.979
r49     0.66931     0.31679     0.43821     0.7923      0.12979     0.75311     0.79466     0.0790
r50     0.58594     0.35426     0.7651      0.51872     0.86415     0.58281     0.84795     0.45

xxx TABLE:mat_2_boolean xxxxxxxxxxxxxxxxxxxx
      c1      c2      c3      c4      c50      c51      c52      c53
-----
r1     true     false     false     true     true     false     true     true
r2     true     false     true      true     false     false     true     true
r3     false     true     false     true     false     true     false     true
r4     false     true     false     false     false     true     true     true
r5     true      true      true     false     true     false     false     true
r46    false     true      true     false     true     true     true     true
r47    true      true      true     true      true     true     false     false
r48    true     false     false     false     true     true     false     true
r49    true      true     false     true      true     true     false     false
r50    false     false     false     false     false     false     false     false

xxx TABLE:mat_3 xxxxxxxxxxxxxxxxxxxx
      c1      c2
-----
r1     0.00012471    0.13253
r2     0.88615      0.79226

xxx TABLE:tensor_1 xxxxxxxxxxxxxxxxxxxx
      c1      c2      c3      c4      c5      c6      c7      c8
-----
r1     0.019363     0.34271     0.52167     0.53703     0.75756     0.68839     0.8345     0.26597
r2     0.018091     0.33355     0.11738     0.77857     0.81933     0.28644     0.6157     0.368

xxx TABLE:tensor_2 xxxxxxxxxxxxxxxxxxxx
      c1      c2      c3      c4      c22      c23      c24      c25
-----
r1     0.51866     0.40495     0.48278     0.99731     0.46584     0.62976     0.035924     0.10505
r2     0.028692     0.37408     0.24149     0.35201     0.66054     0.87243     0.0024293     0.81088
r3     0.87339     0.19457     0.83212     0.15315     0.77859     0.96663     0.2501     0.8056

xxx TABLE:tensor_3 xxxxxxxxxxxxxxxxxxxx
      c1      c2      c3      c4
-----
r1     0.1219     0.5119     0.91553     0.14329

xxx TABLE:tesseract_1 xxxxxxxxxxxxxxxxxxxx
      c1      c2      c3      c4      c21      c22      c23      c24

```

```

-----
r1    0.64531    0.59299    0.32115    0.67653    0.90328    0.56911    0.52562    0.12014
r2    0.74558    0.5007    0.46142    0.21384    0.35564    0.13732    0.155    0.23786
r3    0.91137    0.46403    0.18118    0.049919    0.46246    0.46842    0.75348    0.64547

xxx TABLE:tesseract_2 xxxxxxxxxxxxxxxxxxxxxxxx
      c1      c2      c3      c4      c7      c8      c9      c10
-----
r1    0.28898    0.48211    0.44359    0.97146    0.61782    0.65121    0.80715    0.11605
r2    0.094493    0.34941    0.17595    0.14192    0.16754    0.57097    0.043114    0.70518

xxx TABLE:tesseract_bl_3 xxxxxxxxxxxxxxxxxxxxxxxx
      c1      c2      c3      c4      c7      c8      c9      c10
-----
r1    false    false    true    true    false    true    false    false

-----
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Scalars
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
      i      idx      value
      -      ---      -
boolean_1    1      1      1
empty        2      2      NaN
mat_4        3      11     0.74898
string_float_1 4      13     1021.1
string_int_1  5      14     1021

-----
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
String
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
      i      idx      string
      ---      ----      -
list_string_1 "1"    "5"    "col1;col2;col3;col4"
list_string_2 "2"    "6"    "row1;row2;row3;row4"
string_1      "3"    "12"   "Table Name"

-----
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Functions
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
      i      idx      functionString
      ---      ---      -
func1    "1"    "3"    "@(x)1+2+x"
func2    "2"    "4"    "@(x,y)x*1+sqrt(y)"

```

4.1.2 Test FF_CONTAINER_MAP_DISPLAY summarize Matrix Only

Three large matrixes, show summaries

```
% Create Container
```

```
mp_container_map = containers.Map('KeyType','char', 'ValueType','any');
```

```

rng(123);
mp_container_map('mat_1') = rand(100,100);
mp_container_map('mat_2') = rand(100,100)*2 + 1;
mp_container_map('mat_2_boolean') = (rand(100,100) > 0.5);
% Will only print
ff_container_map_display(mp_container_map);

-----
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_container_map ND Array (Matrix etc)
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

```

	i	idx	ndim	numel	rowN	colN	mean	std	coefvari
	-	---	----	-----	----	----	-----	-----	-----
mat_1	1	1	2	10000	100	100	0.49823	0.28829	0.57863
mat_2	2	2	2	10000	100	100	2.0029	0.57632	0.28774
mat_2_boolean	3	3	2	10000	100	100	0.4995	0.50002	1.0011

4.1.3 Test FF_CONTAINER_MAP_DISPLAY Show Matrix Subset

A container map with three small matrixes, print only only 2 rows and 3 columns.

```

% Create Container
mp_container_map = containers.Map('KeyType','char', 'ValueType','any');
rng(789);
mp_container_map('mat_1') = rand(3,4);
mp_container_map('mat_2') = rand(50,53);
mp_container_map('mat_2_boolean') = (rand(50,53) > 0.5);
% Will only print
ff_container_map_display(mp_container_map, 2, 3);

-----
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_container_map ND Array (Matrix etc)
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

```

	i	idx	ndim	numel	rowN	colN	mean	std	coefvari
	-	---	----	-----	----	----	-----	-----	-----
mat_1	1	1	2	12	3	4	0.41564	0.33586	0.80805
mat_2	2	2	2	2650	50	53	0.49973	0.28834	0.57699
mat_2_boolean	3	3	2	2650	50	53	0.50943	0.50001	0.98149

```

xxx TABLE:mat_1 xxxxxxxxxxxxxxxxxxxxxxx
      c1      c2      c3      c4
      -----
r1    0.32333  0.62442  0.01062  0.53815
r3    0.79378  0.75889  0.11104  0.55157

xxx TABLE:mat_2 xxxxxxxxxxxxxxxxxxxxxxx
      c1      c2      c52      c53
      -----
r1    0.72837  0.20976  0.74583  0.22321
r50   0.52812  0.545   0.49521  0.29826

xxx TABLE:mat_2_boolean xxxxxxxxxxxxxxxxxxxxxxx
      c1      c2      c52      c53
      -----

```


r1	false	true	true	true
r50	true	false	false	true

Appendix A

Index and Code Links

A.1 Summarize Policy and Value links

1. [Summarize ND Array Policy and Value Functions: `mlx` | `m` | `pdf` | `html`](#)
 - Given an NDarray matrix with N1, N2, ..., ND dimensions. Generate average and standard deviation for the 3rd dimension, grouping by the other dimensions.
 - For example, show the 5th dimension as the column groups, and the other variables generate combinations shown as rows.
 - The resulting summary statistics table contains mean and standard deviation among other statistics over the policy or value contained in the ND array.
 - **MEconTools:** `ff_summ_nd_array()`

A.2 Distributional Analysis links

1. [Gateway Joint Probability Mass Statistics: `mlx` | `m` | `pdf` | `html`](#)
 - Given probability mass function $f(s)$, and information $y(s)$, $x(s)$, $z(s)$ at each element of the state-space, compute statistics for each variable, y , x , z , which are all discrete random variables.
 - Compute their correlation and covariance.
 - **MEconTools:** `ff_simu_stats()`
2. [Discrete Random Variable Distributional Statistics: `mlx` | `m` | `pdf` | `html`](#)
 - Model simulation generates discrete random variables, calculate mean, standard deviation, min, max, percentiles, and proportion of outcomes held by x percentiles, etc.
 - **MEconTools:** `ff_disc_rand_var_stats()`
3. [Generate Discrete Random Variable: `mlx` | `m` | `pdf` | `html`](#)
 - Given mass at state space points, and y , c , a , z and other outcomes or other information at each corresponding state space points, generate discrete random variable, with unique sorted values, and mass for each unique sorted values.
 - Generate additional joint distributions: if initial distribution is over $f(a,z)$, generate joint distribution of $f(y,a)$ or $f(y,z)$.
 - **MEconTools:** `ff_disc_rand_var_mass2outcomes()`
4. [Discrete Random Variable Correlation and Covariance: `mlx` | `m` | `pdf` | `html`](#)
 - Given probability mass function $f(s)$, $X(s)$, and $Y(s)$, compute the covariance and correlation between X and Y .
 - **MEconTools:** `ff_disc_rand_var_mass2covcor()`

A.3 Graphs links

1. [Multiple Line Graph Function: `mlx` | `m` | `pdf` | `html`](#)
 - Grid based Graph, x-axis one param, color another param, over outcomes.
 - **MEconTools:** `ff_graph_grid()`

A.4 Support Tools links

1. [Organizes and Prints Container Map Key and Values: `mlx` | `m` | `pdf` | `html`](#)
 - Summarizes the contents of a map container by data types. Includes, scalar, array, matrix, string, functions, tensors (3-tuples), tesseracts (4-tuples).
 - **MEconTools**: `ff_container_map_display()`

Bibliography

The MathWorks Inc (2019). *MATLAB*. Matlab package version 2019b.

Xie, Y. (2020). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.18.