Data Structures and Dynamic Optimization with Matlab

Fan Wang

2020-07-02

Contents

Preface

This is a work-in-progress website of support files for using matlab. Materials gathered from various projects in which matlab is used. Matlab files are linked below by section with livescript files. Tested with Matlab 2019a (The MathWorks Inc, 2019). This is not a Matlab package, but a list of examples in PDF/HTML/Mlx formats. MEconTools is a package that can be installed with tools used in projects involving matlab code.

Bullet points in the Appendix show which matlab functions/commands are used to achieve various objectives. The goal of this repository is to make it easier to find/re-use codes produced for various projects. Some functions also rely on or correspond to functions from MEconTools (Wang, 2020).

From other repositories: For dynamic borrowing and savings problems, see Dynamic Asset Repository; For code examples, see also R 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

Data Structures

1.1 Matrices and Arrays

1.1.1 Array Reshape, Repeat and Expand Examples

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.1.1.1 Basic Examples of Reshape

```
a = [1,2,3,4,5,6]';
b = reshape(a, [3,2])

b = 3x2
     1     4
     2     5
     3     6

b(:)

ans = 6x1
     1
     2
     3
     4
     5
     6

a = [1,2,3;4,5,6;7,8,9;10,11,12]'

a = 3x4
     1     4     7    10
     2     5     8    11
     3     6     9    12

b = reshape(a, [6,2])

b = 6x2
     1     7
     2     8
     3     9
     4    10
     5    11
```

```
     6    12
```

1.1.1.2 Stack Two Matrix of Equal Column Count Together

```
a = [1,2;3,4];
a_stacked = [a;a;a];
disp(a_stacked);
```

```
     1     2
     3     4
     1     2
     3     4
     1     2
     3     4
```

1.1.1.3 Repeat/Duplicate Matrix Downwards

There is a 2 by 3 matrix, to be repeated 4 times, downwards. This is useful for replicating data matrix for say counterfactual purposes.

Below, we have two ways of repeating a matrix downwards. Copy as whole, or copy row by row.

```
row_count = 2;
col_count = 3;
repeat_mat_count = 2;

data_vec = 1:(row_count*col_count);
searchMatrix = reshape(data_vec,row_count,col_count);

% To repeat matrix downwards
rep_rows_idx = [1:row_count]'*ones(1,repeat_mat_count);
rep_rows_idx = rep_rows_idx(:);

rep_cols_idx = [1:col_count];
rep_cols_idx = rep_cols_idx(:);

searchMatrixRep_stack = searchMatrix(rep_rows_idx, rep_cols_idx);

% To insert repeated rows following original rows
rep_rows_idx = ([1:row_count]'*ones(1,repeat_mat_count))';
rep_rows_idx = rep_rows_idx(:);

searchMatrixRep_dup = searchMatrix(rep_rows_idx, rep_cols_idx);

disp(searchMatrix)
```

```
     1     3     5
     2     4     6
```

```
disp(searchMatrixRep_stack)
```

```
     1     3     5
     2     4     6
     1     3     5
     2     4     6
```

```
disp(searchMatrixRep_dup)
```

```
     1     3     5
     1     3     5
```

```
     2     4     6
     2     4     6
```

1.1.1.4 Index Dimension Transform

it_inner_fin = 5; it_outter_fin = 3;

it_inner_cur = it_outter_fin it_outter_cur = it_inner_fin

ar_it_cols_idx = 1:1:(it_inner_fin*it_outter_fin) ar_it_cols_inner_dim = repmat(1:it_inner_cur,

$$it\_outter\_cur, 1$$

) ar_it_cols_inner_dim(:)'

mt_it_cols_idx = reshape(ar_it_cols_idx,

$$it\_inner\_cur, it\_outter\_cur$$

)' mt_it_cols_idx(:)'

```
it_inner_fin = 5;
it_outter_fin = 3;

ar_it_cols_idx = 1:1:(it_inner_fin*it_outter_fin)

ar_it_cols_idx = 1x15
     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15

mt_it_cols_idx = reshape(ar_it_cols_idx, [it_outter_fin, it_inner_fin])'

mt_it_cols_idx = 5x3
     1     2     3
     4     5     6
     7     8     9
    10    11    12
    13    14    15

mt_it_cols_idx(:)'

ans = 1x15
     1     4     7    10    13     2     5     8    11    14     3     6     9    12    15
```

1.1.2 Array Index Slicing and Subsetting to Replace and Expand

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.1.2.1 Index Select Rows and Columns of a 2D matrix

In the example below, select by entire rows and columns:

```
% There is a 2D Matrix
rng(123);
randMatZ = rand(3,6);
disp(randMatZ);

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

% Duplicate Select Row sand Columns of Elements
```

```
disp(randMatZ([1,2,3,3,3,2], [1,1,2,2,2,1]))
```

```
    0.6965    0.6965    0.5513    0.5513    0.5513    0.6965
    0.2861    0.2861    0.7195    0.7195    0.7195    0.2861
    0.2269    0.2269    0.4231    0.4231    0.4231    0.2269
    0.2269    0.2269    0.4231    0.4231    0.4231    0.2269
    0.2269    0.2269    0.4231    0.4231    0.4231    0.2269
    0.2861    0.2861    0.7195    0.7195    0.7195    0.2861
```

1.1.2.2 Index Select Set of Elements from 2D matrix

Rather than selecting entire rows and columns, suppose we want to select only one element at row 1 col 2, the element at row 2 col 4, element at row 5 col 1, etc.

```
% Select Subset of Elements
it_row_idx = [1,2,3,1,3,2];
it_col_idx = [1,1,5,4,2,3];
% Select sub2idx
ar_lin_idx = sub2ind(size(randMatZ), it_row_idx, it_col_idx);
ar_sel_val = randMatZ(ar_lin_idx);
disp(ar_sel_val');
```

```
    0.6965
    0.2861
    0.3980
    0.3921
    0.4231
    0.6848
```

1.1.2.3 Find Closest Element of Array to Each Element of Another Array

Given scalar value, find the cloest value in array:

```
fl_a = 3.4;
ar_bb = [1,2,3,4];
[fl_min, it_min_idx] = min(abs(ar_bb-fl_a));
disp(it_min_idx);
```

```
     3
```

Given a scalar value and an array, find the closest smaller value in the array to the scalar value:

```
fl_a = 2.1;
ar_bb = [1,2,3,4];
disp(sum(ar_bb<fl_a));
```

```
     2
```

Array A is between 0 and 1, on some grid. Array B is also between 0 and 1, but scattered. Find for each element of B the index of the cloest value on A that is smaller than the element in B.

```
rng(1234);
ar_a = linspace(0,10,5);
ar_b = rand([5,1])*10;
mt_a_less_b = ar_a<ar_b;
mt_a_less_b_idx = sum(ar_a<ar_b, 2);
disp(ar_a);
```

```
         0    2.5000    5.0000    7.5000   10.0000
```

```
disp(ar_b);
```

```
    1.9152
    6.2211
    4.3773
    7.8536
    7.7998

disp(mt_a_less_b);

   1   0   0   0   0
   1   1   1   0   0
   1   1   0   0   0
   1   1   1   1   0
   1   1   1   1   0

disp(mt_a_less_b_idx);

     1
     3
     2
     4
     4
```

1.1.2.4 Matlab Index based Replacement of Subset of Matrix Values

```
rng(123);
randMatZ = rand(3,6)+1;
randMat = rand(3,6)-0.5;

output = max(-randMat,0);
randMatZ(output==0) = 999;
min(randMatZ,[],2);
 randMatZ((max(-randMat,0))==0) = 999;
disp(randMatZ);

  999.0000  999.0000  999.0000    1.3921    1.4386    1.7380
  999.0000  999.0000    1.6848    1.3432    1.0597    1.1825
  999.0000  999.0000    1.4809  999.0000    1.3980    1.1755

disp(min(randMatZ,[],2));

    1.3921
    1.0597
    1.1755
```

1.1.2.5 Matlab Matrix Index Based Matrix Expansion (Manual)

In the example below, we start with a 4 by 2 matrix, than we expand specific rows and columns of the matrix. Specifically, we expand the matrix such that the result matrix repeats the 1st, 2nd, 1st, 2nd, then 3rd, than 1st, 1st, and 1st rows. And repeats column 1, then 2nd, then 2nd, then 2nd, and finally the first column.

```
% Original Matrix
Z = 2;
N = 2;
Q = 2;
base_mat = reshape(1:(Z*N*Q),Z*N,Q);
disp(base_mat);

     1     5
```

```
     2     6
     3     7
     4     8

% Expanded Matrix
base_expand = base_mat([1,2,1,2,3,1,1,1],[1,2,2,2,1]);
disp(base_expand);

     1     5     5     5     1
     2     6     6     6     2
     1     5     5     5     1
     2     6     6     6     2
     3     7     7     7     3
     1     5     5     5     1
     1     5     5     5     1
     1     5     5     5     1
```

1.1.2.6 Duplicate Matrix Downwards N times Using Index

The example here has the same idea, but we do the operations above in a more automated way. This could be done using alternative methods.

```
% Original Matrix
Z = 2;
N = 2;
Q = 2;
base_mat = reshape(1:(Z*N*Q),Z*N,Q);
disp(base_mat);

     1     5
     2     6
     3     7
     4     8

% Generate row Index many times automatically depending on how many times
% to replicate
vmat_repeat_count = 3;
vmat_reindex_rows_repeat = [1:(Z*N)]'* ones(1,vmat_repeat_count);
vmat_reindex_rows_repeat = vmat_reindex_rows_repeat(:);
disp(vmat_reindex_rows_repeat');

     1     2     3     4     1     2     3     4     1     2     3     4

% Duplicate Matrix by the Rows specified above, and using the same number
% of columns.
mat_repdown = base_mat(vmat_reindex_rows_repeat(:), 1:Q);
disp(mat_repdown');

     1     2     3     4     1     2     3     4     1     2     3     4
     5     6     7     8     5     6     7     8     5     6     7     8
```

1.1.2.7 Max of Matrix column by Column Linear to 2d Index

Finding max of matrix column by column, then obtain the linear index associated with the max values.

```
randMat = rand(5,3);
disp(randMat);

    0.4264    0.1156    0.4830
    0.8934    0.3173    0.9856
```

```
    0.9442    0.4148    0.5195
    0.5018    0.8663    0.6129
    0.6240    0.2505    0.1206

[maxVal maxIndex] = max(randMat);
linearIndex = sub2ind(size(randMat),maxIndex,(1:1:size(randMat,2)))

linearIndex = 1x3
     3     9    12

randMat(linearIndex)

ans = 1x3
    0.9442    0.8663    0.9856

t_pV = [1,2;3,4;5,6];
t_pV_Ind = [1,1;0,0;1,1];
[maxVal maxIndex] = max(t_pV(t_pV_Ind==1))

maxVal = 6
maxIndex = 4
```

1.1.2.8 Given Array of size M, Select N somewhat equi-distance elements

```
% Subset count
it_n = 5;

% Example 1, long array
ar_fl_a = 1:1.1:100;
ar_it_subset_idx = unique(round(((0:1:(it_n-1))/(it_n-1))*(length(ar_fl_a)-1)+1));
ar_fl_a_subset = ar_fl_a(ar_it_subset_idx);
disp(ar_fl_a_subset);

    1.0000   26.3000   50.5000   75.8000  100.0000


% Example 2, Short Array
ar_fl_a = 1:1.1:3;
ar_it_subset_idx = unique(round(((0:1:(it_n-1))/(it_n-1))*(length(ar_fl_a)-1)+1));
ar_fl_a_subset = ar_fl_a(ar_it_subset_idx);
disp(ar_fl_a_subset);

    1.0000    2.1000


% Write As function
f_subset = @(it_subset_n, it_ar_n) unique(round(((0:1:(it_subset_n-1))/(it_subset_n-1))*(it_ar_n-1)+

% Select 5 out of 10
disp(f_subset(5, 10));

     1     3     6     8    10


% Select 10 out of 5
disp(f_subset(10, 5));

     1     2     3     4     5
```

```
% Select 5 out of 5
disp(f_subset(5, 5));
```

```
     1     2     3     4     5
```

1.1.3 3D, 4D, ND Arrays Reshape and Rearrange Dimensions

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.1.3.1 3D Array to Cell Array of Matrix Split by Last Dimension

Convert Multi-dimensional arrays to a cell array consistent of two dimensional arrays. In this example, we split by the 3rd dimension, so the number of output matrixes is equal to the length of the 3rd dimension.

First create a three dimensional array, two matrixes that are 4 by 3 each:

```
% Create a 3D Array
rng(123);
mn_rand = rand(4,3,2);
disp(mn_rand);
```

```
(:,:,1) =

    0.6965    0.7195    0.4809
    0.2861    0.4231    0.3921
    0.2269    0.9808    0.3432
    0.5513    0.6848    0.7290


(:,:,2) =

    0.4386    0.1825    0.6344
    0.0597    0.1755    0.8494
    0.3980    0.5316    0.7245
    0.7380    0.5318    0.6110
```

Now convert the 3 dimensional array to a 2 by 1 cell array that contains matrixes in each cell:

```
% Squeece 3D array to a Cell array of matrixes
cl_mn_rand = squeeze(num2cell(mn_rand, [1,2]));
celldisp(cl_mn_rand);
```

```
cl_mn_rand{1} =

    0.6965    0.7195    0.4809
    0.2861    0.4231    0.3921
    0.2269    0.9808    0.3432
    0.5513    0.6848    0.7290



cl_mn_rand{2} =

    0.4386    0.1825    0.6344
    0.0597    0.1755    0.8494
    0.3980    0.5316    0.7245
```

```
    0.7380    0.5318    0.6110
```

1.1.3.2 4D Array to Cell Array of Matrix Split by Last Two Dimensions

Convert 4D Multi-dimensional arrays to a cell array consistent of two dimensional arrays. In this example, the first two dimensions determine the resulting matrix size, the the 3rd and the 4th dimensions are categorical.

First create a four dimensional array, four matrixes stored each matrix is 2 by 2:

```
% Create a 3D Array
rng(123);
mn_rand = rand(2,2,2,2);
disp(mn_rand);

(:,:,1,1) =

    0.6965    0.2269
    0.2861    0.5513


(:,:,2,1) =

    0.7195    0.9808
    0.4231    0.6848


(:,:,1,2) =

    0.4809    0.3432
    0.3921    0.7290


(:,:,2,2) =

    0.4386    0.3980
    0.0597    0.7380
```

Now convert the 4 dimensional array to a 2 by 2 cell array that contains matrixes in each cell:

```
% Squeece 3D array to a Cell array of matrixes
cl_mn_rand = squeeze(num2cell(mn_rand, [1,2]));
celldisp(cl_mn_rand);


cl_mn_rand{1,1} =

    0.6965    0.2269
    0.2861    0.5513



cl_mn_rand{2,1} =

    0.7195    0.9808
    0.4231    0.6848



cl_mn_rand{1,2} =
```

```
    0.4809    0.3432
    0.3921    0.7290



cl_mn_rand{2,2} =

    0.4386    0.3980
    0.0597    0.7380
```

1.1.3.3 4D Array to Cell Array of Matrix Split by First and Fourth Dimensions Rearrange Dimensions

Suppose we store policy and value function given four state variables. The first one is age, the second one is asset, the third one is shock, and the fourth one is the number of kids. We start out with a four dimensional matrix. The objective is to create a two dimensional cell array as output where indexed by the 1st and 4th dimension of the underlying numeric array, and the elements of the 2D cell array are matrixes.

This is achieved by the permute function. We first rearrange the matrix, so that the 2nd and 3rd dimensions become the 1st and 2nd, then we use the technique used above to squeeze out the first two dimensions as matrixes with the last two as categories.

First, generate the 2 by 2 by 2 by 2, (Age, A, Z, Kids Count), matrix:

```
% Create a 3D Array
rng(123);
% (Age, A, Z, Kids Count)
mn_rand = rand(2,2,2,2);
```

Second, loop out the (A,Z) matrix by Age and Kids Count, this shows us what we want to achieve. Note that each row is Age, each column is A, each submatrix is z, and each super-matrix is kid-count. So from slicing, each column printed out are different value of A, the two submatrixes printed out are for each z. For the output structure where we want a (A,Z) matrix, the columns need to become rows, and the submatrix need to become columns.

```
% Show Matrix by Age and Kids
for it_age = 1:size(mn_rand,1)
    for it_kids = 1:size(mn_rand,4)
        disp(strcat(['it_age:' num2str(it_age) ', it_kids:' num2str(it_kids)]))
        disp(mn_rand(it_age,:,:,it_kids));
    end
end

it_age:1, it_kids:1
(:,:,1) =

    0.6965    0.2269


(:,:,2) =

    0.7195    0.9808
it_age:1, it_kids:2
(:,:,1) =

    0.4809    0.3432


(:,:,2) =
```

```
    0.4386    0.3980
it_age:2, it_kids:1
(:,:,1) =

    0.2861    0.5513


(:,:,2) =

    0.4231    0.6848
it_age:2, it_kids:2
(:,:,1) =

    0.3921    0.7290


(:,:,2) =

    0.0597    0.7380
```

Third, we permutate the matrix and squeeze to arrive at the 2 by 2 cell, note that step two is just to show via loop what we should get:

```
% Rearrange dimensions
mn_rand_2314 = permute(mn_rand, [2,3,1,4]);
% Squeeze the first two dimensiosn as before
cl_mn_rand = squeeze(num2cell(mn_rand_2314, [1,2]));
% show
celldisp(cl_mn_rand);


cl_mn_rand{1,1} =

    0.6965    0.7195
    0.2269    0.9808



cl_mn_rand{2,1} =

    0.2861    0.4231
    0.5513    0.6848



cl_mn_rand{1,2} =

    0.4809    0.4386
    0.3432    0.3980



cl_mn_rand{2,2} =

    0.3921    0.0597
    0.7290    0.7380
```

1.1.3.4 ND Array Summarize in Table

Given an ND dataframe, summarize the first two dimensions. For each possible combination of the 3rd and 4th dimension, generate mean, sd, min and max over the matrix of the first two dimensions. This is similar to a tabulation function.

First, we generate several array of information:

```
% Initialize and Squeeze
rng(123);
mn_rand = rand(2,2,2,2);
cln_mt_rand = squeeze(num2cell(mn_rand, [1,2]));
cl_mt_rand = cln_mt_rand(:);
celldisp(cl_mt_rand);


cl_mt_rand{1} =

    0.6965    0.2269
    0.2861    0.5513



cl_mt_rand{2} =

    0.7195    0.9808
    0.4231    0.6848



cl_mt_rand{3} =

    0.4809    0.3432
    0.3921    0.7290



cl_mt_rand{4} =

    0.4386    0.3980
    0.0597    0.7380
```

Second, create two arrays that tracks for each element of cl_mt_rand, which one of the 3rd and 4th dimensions they correspond to:

```
ar_dim_3 = [31,32]';
ar_dim_4 = [41,42]';
[mt_dim_3, mt_dim_4] = ndgrid(ar_dim_3, ar_dim_4);
ar_dim_3 = mt_dim_3(:);
ar_dim_4 = mt_dim_4(:);
```

Third, summarize each matrix:

```
% Over of matrix and summarize
ar_mean = zeros(size(cl_mt_rand));
ar_std = zeros(size(cl_mt_rand));
for it_mt=1:length(cl_mt_rand)
    mt_cur = cl_mt_rand{it_mt};
    ar_mean(it_mt) = mean(mt_cur, 'all');
    ar_std(it_mt) = std(mt_cur, [], 'all');
end
```

Fourth Construct a Table

```
% Constructe Table
tb_rowcols_tab = array2table([(1:length(cl_mt_rand))', ...
    ar_dim_3, ar_dim_4, ar_mean, ar_std]);
tb_rowcols_tab.Properties.VariableNames = ...
    matlab.lang.makeValidName(["i", "dim3", "dim4",  "mean", "std"]);
disp(tb_rowcols_tab);
```

```
    i    dim3    dim4     mean        std
    _    ____    ____    _______    _______

    1     31      41     0.44019    0.22156
    2     32      41     0.70204     0.2281
    3     31      42     0.48632    0.17157
    4     32      42     0.40857    0.27764
```

1.1.3.5 ND Array Two-Way Summarize in Table

Given dataframe as above, but we now want to add to the resulting summary table additional columns, rather than taking the means of the entire matrix in the first two dimensions, we only take average with respect to the rows, the first dimension, the second dimension show up as coumn statistics names, still multiple stats.

First, we generate several array of information:

```
% dimension names
st_title = 'Summarize values over a conditional on z (columns) and kids and marriage (rows)';
st_dim_1 = 'a';
st_dim_2 = 'z';
st_dim_3 = 'kid';
st_dim_4 = 'marriage';
% 3rd and fourth dimension values
ar_dim_2 = [-3, -1, 1, 3];
ar_dim_3 = [1,2,3];
ar_dim_4 = [0,1];
% Initialize and Squeeze
rng(123);
mn_rand = rand(10,4,3,2);
cln_mt_rand = squeeze(num2cell(mn_rand, [1,2]));
cl_mt_rand = cln_mt_rand(:);
```

Second, create two arrays that tracks for each element of cl_mt_rand, which one of the 3rd and 4th dimensions they correspond to:

```
[mt_dim_3, mt_dim_4] = ndgrid(ar_dim_3', ar_dim_4');
ar_dim_3 = mt_dim_3(:);
ar_dim_4 = mt_dim_4(:);
```

Third, summarize each matrix:

```
% Over of matrix and summarize
mt_mean = zeros(length(cl_mt_rand), size(mn_rand,2));
mt_std = zeros(length(cl_mt_rand), size(mn_rand,2));
for it_mt=1:length(cl_mt_rand)
    mt_cur = cl_mt_rand{it_mt};
    mt_mean(it_mt,:) = mean(mt_cur, 1);
    mt_std(it_mt,:) = std(mt_cur, [], 1);
end
```

Fourth Construct a Table

```
% Constructe Table
```

```
tb_rowcols_tab = array2table([(1:length(cl_mt_rand))', ...
    ar_dim_3, ar_dim_4, mt_mean, mt_std]);
% Column Names
cl_col_names_cate_dims = [string(st_dim_3), string(st_dim_4)];
cl_col_names_mn = strcat('mean_', st_dim_2, string(ar_dim_2));
cl_col_names_sd = strcat('sd_', st_dim_2, string(ar_dim_2));
tb_rowcols_tab.Properties.VariableNames = ...
    matlab.lang.makeValidName(["group", cl_col_names_cate_dims, cl_col_names_mn, cl_col_names_sd]);
% disp(['xxx ' st_title ' xxxxxxxxxxxxxxxxxxxxxxxxxxxx']);
disp(tb_rowcols_tab);
```

group	kid	marriage	mean_z_3	mean_z_1	mean_z1	mean_z3	sd_z_3	sd_z_1
1	1	0	0.5442	0.41278	0.53795	0.49542	0.22935	0.22945
2	2	0	0.51894	0.52262	0.52544	0.45066	0.26787	0.23615
3	3	0	0.48248	0.5238	0.50392	0.46534	0.27009	0.26676
4	1	1	0.58343	0.50529	0.54361	0.5006	0.29578	0.30182
5	2	1	0.58408	0.45941	0.50466	0.40081	0.25026	0.34704
6	3	1	0.51148	0.49531	0.48963	0.47698	0.3271	0.24336

1.1.4 Array Broadcast and Expansion Examples

back to Fan's Reusable Matlab Repository or Dynamic Asset Repository.

Matrix broadcasting was added to matlab's recent editions. This is an important step for vectorizing codes. Proper usage of broadcasting reduces memory allocation requirements for matrix matrix operations.

1.1.4.1 Broadcasting with A Row and a Column

Below we add together a 1 by 3 and 4 by 1 array, that should not work. With broadcasting, it is assumed that we will mesh the arrays and then sum up the meshed matrixes.

```
clear all
ar_A = [1,2,3];
ar_B = [4,3,2,1]';
disp(size(ar_A));
```

```
     1     3
```

```
disp(size(ar_B));
```

```
     4     1
```

```
mt_A_B_broadcast = ar_A + ar_B;
disp(mt_A_B_broadcast);
```

```
     5     6     7
     4     5     6
     3     4     5
     2     3     4
```

```
mt_A_B_broadcast_product = ar_A.*ar_B;
disp(mt_A_B_broadcast_product);
```

```
     4     8    12
     3     6     9
     2     4     6
     1     2     3
```

1.1.4.2 Broadcasting with One Row and One Matrix

Below we add together a 1 by 3 and 4 by 3 matrix, that should not work. With broadcasting, it is assumed that we will repeat the array four times, duplicating the single row four times, so the matrix dimensions match up.

```
clear all
ar_A = [1,2,3];
mt_B = [4,3,2,1;5,4,3,2;6,5,4,3]';
disp(size(ar_A));

     1     3

disp(size(mt_B));

     4     3

mt_A_B_broadcast = ar_A + mt_B;
disp(mt_A_B_broadcast);

     5     7     9
     4     6     8
     3     5     7
     2     4     6

mt_A_B_broadcast_product = ar_A.*mt_B;
disp(mt_A_B_broadcast_product);

     4    10    18
     3     8    15
     2     6    12
     1     4     9
```

1.1.4.3 Broadcasting with One Column and One Matrix

Below we add together a 4 by 1 and 4 by 3 matrix, that should not work. With broadcasting, it is assumed that we will repeat the column three times, duplicating the single column three times, so the matrix dimensions match up.

```
clear all
ar_A = [4,3,2,1]';
mt_B = [4,3,2,1;5,4,3,2;6,5,4,3]';
disp(size(ar_A));

     4     1

disp(size(mt_B));

     4     3

mt_A_B_broadcast = ar_A + mt_B;
disp(mt_A_B_broadcast);

     8     9    10
     6     7     8
     4     5     6
     2     3     4

mt_A_B_broadcast_product = ar_A.*mt_B;
disp(mt_A_B_broadcast_product);
```

```
    16    20    24
     9    12    15
     4     6     8
     1     2     3
```

1.1.4.4 Expand with Broadcast, Percentage Choice grids

```
clear all
ar_w_perc = [0.1,0.5,0.9]

ar_w_perc = 1x3
    0.1000    0.5000    0.9000

ar_w_level = [-2,0,2]

ar_w_level = 1x3
    -2     0     2

fl_b_bd = -4

fl_b_bd = -4

ar_k_max = ar_w_level - fl_b_bd

ar_k_max = 1x3
     2     4     6

ar_ak_perc = [0.1,0.3,0.7,0.9]

ar_ak_perc = 1x4
    0.1000    0.3000    0.7000    0.9000

mt_k = (ar_k_max'*ar_ak_perc)'

mt_k = 4x3
    0.2000    0.4000    0.6000
    0.6000    1.2000    1.8000
    1.4000    2.8000    4.2000
    1.8000    3.6000    5.4000

mt_a = (ar_w_level - mt_k)

mt_a = 4x3
   -2.2000   -0.4000    1.4000
   -2.6000   -1.2000    0.2000
   -3.4000   -2.8000   -2.2000
   -3.8000   -3.6000   -3.4000
```

1.1.4.5 Expand Matrix Twice

```
clear all
% Same as above
ar_w_level = [-2,-1,-0.1]

ar_w_level = 1x3
   -2.0000   -1.0000   -0.1000

fl_b_bd = -4
```

```
fl_b_bd = -4

ar_k_max = ar_w_level - fl_b_bd

ar_k_max = 1x3
    2.0000    3.0000    3.9000

ar_ak_perc = [0.001, 0.1,0.3,0.7,0.9, 0.999]

ar_ak_perc = 1x6
    0.0010    0.1000    0.3000    0.7000    0.9000    0.9990

mt_k = (ar_k_max'*ar_ak_perc)'

mt_k = 6x3
    0.0020    0.0030    0.0039
    0.2000    0.3000    0.3900
    0.6000    0.9000    1.1700
    1.4000    2.1000    2.7300
    1.8000    2.7000    3.5100
    1.9980    2.9970    3.8961

mt_a = (ar_w_level - mt_k)

mt_a = 6x3
   -2.0020   -1.0030   -0.1039
   -2.2000   -1.3000   -0.4900
   -2.6000   -1.9000   -1.2700
   -3.4000   -3.1000   -2.8300
   -3.8000   -3.7000   -3.6100
   -3.9980   -3.9970   -3.9961


% fraction of borrowing for bridge loan
ar_coh_bridge_perc = [0, 0.5, 0.999];

% Expand matrix to include coh percentage dimension
mt_k = repmat(mt_k, [1, length(ar_coh_bridge_perc)])

mt_k = 6x9
    0.0020    0.0030    0.0039    0.0020    0.0030    0.0039    0.0020    0.0030    0.0039
    0.2000    0.3000    0.3900    0.2000    0.3000    0.3900    0.2000    0.3000    0.3900
    0.6000    0.9000    1.1700    0.6000    0.9000    1.1700    0.6000    0.9000    1.1700
    1.4000    2.1000    2.7300    1.4000    2.1000    2.7300    1.4000    2.1000    2.7300
    1.8000    2.7000    3.5100    1.8000    2.7000    3.5100    1.8000    2.7000    3.5100
    1.9980    2.9970    3.8961    1.9980    2.9970    3.8961    1.9980    2.9970    3.8961

mt_a = repmat(mt_a, [1, length(ar_coh_bridge_perc)])

mt_a = 6x9
   -2.0020   -1.0030   -0.1039   -2.0020   -1.0030   -0.1039   -2.0020   -1.0030   -0.1039
   -2.2000   -1.3000   -0.4900   -2.2000   -1.3000   -0.4900   -2.2000   -1.3000   -0.4900
   -2.6000   -1.9000   -1.2700   -2.6000   -1.9000   -1.2700   -2.6000   -1.9000   -1.2700
   -3.4000   -3.1000   -2.8300   -3.4000   -3.1000   -2.8300   -3.4000   -3.1000   -2.8300
   -3.8000   -3.7000   -3.6100   -3.8000   -3.7000   -3.6100   -3.8000   -3.7000   -3.6100
   -3.9980   -3.9970   -3.9961   -3.9980   -3.9970   -3.9961   -3.9980   -3.9970   -3.9961
```

```
mt_a = mt_a

mt_a = 6x9
   -2.0020   -1.0030   -0.1039   -2.0020   -1.0030   -0.1039   -2.0020   -1.0030   -0.1039
   -2.2000   -1.3000   -0.4900   -2.2000   -1.3000   -0.4900   -2.2000   -1.3000   -0.4900
   -2.6000   -1.9000   -1.2700   -2.6000   -1.9000   -1.2700   -2.6000   -1.9000   -1.2700
   -3.4000   -3.1000   -2.8300   -3.4000   -3.1000   -2.8300   -3.4000   -3.1000   -2.8300
   -3.8000   -3.7000   -3.6100   -3.8000   -3.7000   -3.6100   -3.8000   -3.7000   -3.6100
   -3.9980   -3.9970   -3.9961   -3.9980   -3.9970   -3.9961   -3.9980   -3.9970   -3.9961


% bridge loan component of borrowing
ar_brdige_a = (ar_coh_bridge_perc'*ar_w_level)'

ar_brdige_a = 3x3
         0   -1.0000   -1.9980
         0   -0.5000   -0.9990
         0   -0.0500   -0.0999

ar_brdige_a = ar_brdige_a(:)'

ar_brdige_a = 1x9
         0         0         0   -1.0000   -0.5000   -0.0500   -1.9980   -0.9990   -0.0999


% borrowing choices excluding bridge loan
mt_a_nobridge = mt_a - ar_brdige_a

mt_a_nobridge = 6x9
   -2.0020   -1.0030   -0.1039   -1.0020   -0.5030   -0.0539   -0.0040   -0.0040   -0.0040
   -2.2000   -1.3000   -0.4900   -1.2000   -0.8000   -0.4400   -0.2020   -0.3010   -0.3901
   -2.6000   -1.9000   -1.2700   -1.6000   -1.4000   -1.2200   -0.6020   -0.9010   -1.1701
   -3.4000   -3.1000   -2.8300   -2.4000   -2.6000   -2.7800   -1.4020   -2.1010   -2.7301
   -3.8000   -3.7000   -3.6100   -2.8000   -3.2000   -3.5600   -1.8020   -2.7010   -3.5101
   -3.9980   -3.9970   -3.9961   -2.9980   -3.4970   -3.9461   -2.0000   -2.9980   -3.8962
```

1.1.5 Grid States, Choices and Optimal Choices Example

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.1.5.1 Generate State Grid

There many multiple individuals, each individual's value for each state space variable is different. We duplicate that by shockCount and choicecount:

```
stateCount = 2;
shockCount = 3;
choiceCount = 4;

state1 = rand(1,stateCount)

state1 = 1x2
    0.0571    0.6694

states1ShkDup = state1(ones(shockCount*choiceCount,1),:)

states1ShkDup = 12x2
    0.0571    0.6694
```

```
    0.0571    0.6694
    0.0571    0.6694
    0.0571    0.6694
    0.0571    0.6694
    0.0571    0.6694
    0.0571    0.6694
    0.0571    0.6694
    0.0571    0.6694
    0.0571    0.6694

states1ShkDup(:)

ans = 24x1
    0.0571
    0.0571
    0.0571
    0.0571
    0.0571
    0.0571
    0.0571
    0.0571
    0.0571
    0.0571
    0.0571
```

1.1.5.2 Generate Choices

Generate Choice Grid, Example: Each individual has minimal protein and maximal protein they can get Generate a evenly set grid of choices for each individual from min to max. Individual min and max choice is a function of some component of their state-space, such as wealth/income level, and choice is the quantity of good to purchase.

```
stateCount = 2;
shockCount = 3;
choiceCount = 4;

% 1. Min and Max Choices for each state
minprot_n = floor(rand(1,stateCount)*10)

minprot_n = 1x2
     7     7

maxprot_n = minprot_n + floor(rand(1,stateCount)*10)

maxprot_n = 1x2
    14    12

% 2. Choice Ratios, ratios of max-min difference
protChoiceGrid = linspace(0,1,choiceCount)

protChoiceGrid = 1x4
         0    0.3333    0.6667    1.0000

% 3. Each column is a different state.
searchMatrix = (protChoiceGrid'*(maxprot_n-minprot_n)+minprot_n(ones(choiceCount,1),:))

searchMatrix = 4x2
    7.0000    7.0000
    9.3333    8.6667
   11.6667   10.3333
```

```
   14.0000   12.0000

% 4. Each column is a different state, each set of rows is a different shock% for the state. In this
searchMatrix = searchMatrix([1:choiceCount]'* ones(1,shockCount), [1:stateCount]' * ones(1,1))

searchMatrix = 12x2
    7.0000    7.0000
    9.3333    8.6667
   11.6667   10.3333
   14.0000   12.0000
    7.0000    7.0000
    9.3333    8.6667
   11.6667   10.3333
   14.0000   12.0000
    7.0000    7.0000
    9.3333    8.6667

searchMatrix(:)

ans = 24x1
    7.0000
    9.3333
   11.6667
   14.0000
    7.0000
    9.3333
   11.6667
   14.0000
    7.0000
    9.3333
```

1.1.5.3 Average Utility over Shocks

Average of Shocks, E(value) For each STATE and CHOICE, x number of shocks. Need to average over shocks; The raw value output is: STATES * SHOCKS * CHOICES; Code below turn into various things, see MATLAB CODE STRUCTURE in oneNOTE GCC working notes

```
shockCount = 2;
choiceCount = 3;
stateCount = 4;

% 1. VALUE vector (STATES * SHOCKS * CHOICES by 1), this is generated by utility% evaluation functio
valuesOri = sort(rand(choiceCount*shockCount*stateCount,1))

valuesOri = 24x1
    0.0296
    0.1141
    0.1472
    0.1514
    0.1826
    0.1936
    0.2526
    0.2911
    0.3257
    0.3352

% 2. CHOICES by STATES * SHOCKS (ST1 SK1, ST1 SK2; ST2 SK1, etc), each% column are values for differ
values = reshape(valuesOri,[choiceCount,shockCount*stateCount])
```

```
values = 3x8
    0.0296    0.1514    0.2526    0.3352    0.5939    0.7065    0.8791    0.9204
    0.1141    0.1826    0.2911    0.3480    0.5992    0.7267    0.9001    0.9508
    0.1472    0.1936    0.3257    0.4578    0.6576    0.7792    0.9018    0.9658

% 3. SHOCKS by CHOICES * STATES (CH1 ST1, CH1 ST2; CH2 ST1, etc), each% column are two shocks for ea
values = reshape(values',[shockCount, choiceCount*stateCount])

values = 2x12
    0.0296    0.2526    0.5939    0.8791    0.1141    0.2911    0.5992    0.9001    0.1472    0.3257
    0.1514    0.3352    0.7065    0.9204    0.1826    0.3480    0.7267    0.9508    0.1936    0.4578

% 4. AVG: 1 by CHOICES * STATES (CH1 ST1, CH1 ST2; CH2 ST1, etc), take% average over shocks for each
valuesMn = mean(values,1)

valuesMn = 1x12
    0.0905    0.2939    0.6502    0.8997    0.1483    0.3196    0.6629    0.9254    0.1704    0.3918

% 5. AVG: CHOICES * STATES. From this matrix, one can now pick maximum% utility, and match that to t
valuesMn = reshape(valuesMn, [stateCount, choiceCount])'

valuesMn = 3x4
    0.0905    0.2939    0.6502    0.8997
    0.1483    0.3196    0.6629    0.9254
    0.1704    0.3918    0.7184    0.9338
```

1.1.5.4 Pick Optimal Choice

```
choiceCount = 3;
stateCount = 4;

% 1. Matrix, each column is a state, each row is a choice
randMat = rand(choiceCount,stateCount)

randMat = 3x4
    0.0733    0.5905    0.1731    0.1795
    0.0550    0.8539    0.1340    0.3175
    0.3232    0.2871    0.9947    0.5683

% 2. Maximum Value and Maximum Index
[maxVal maxIndex] = max(randMat)

maxVal = 1x4
    0.3232    0.8539    0.9947    0.5683

maxIndex = 1x4
     3     2     3     3

% 3. Linear index
linearIdx = maxIndex + ((1:stateCount)-1)*choiceCount

linearIdx = 1x4
     3     5     9    12

% 4. Optimal Choices
randMat(linearIdx)

ans = 1x4
```

```
    0.3232    0.8539    0.9947    0.5683
```

1.1.6 Accumarray Examples

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.1.6.1 Accumarry Basic Example

There are three unique values in ar_a, sum up the probabilities for each of the unique states. This is equivalent to sorting a matrix with a and prob, and computing sum for each.

```
ar_a = [3,2,1,3]';
ar_prob = [0.1,0.2,0.31,0.39]';
ar_sumprob = accumarray(ar_a, ar_prob);
tb_summed_prob = table(sort(unique(ar_a)), ar_sumprob);
disp(tb_summed_prob);
```

```
    Var1    ar_sumprob
    ____    __________

     1         0.31
     2          0.2
     3         0.49
```

1.1.6.2 Accumarry For Discrete Random Variable

Upon solving a model, if we look for the mass at certain choices or states, accumarray could help aggregate up probabilities

```
a1 = [1,1,2,2]

a1 = 1x4
     1     1     2     2

a2 = [3,2,1,3]

a2 = 1x4
     3     2     1     3

a3 = [1,2,3,3]

a3 = 1x4
     1     2     3     3


a = [a1;a2;a3]'/2

a = 4x3
    0.5000    1.5000    0.5000
    0.5000    1.0000    1.0000
    1.0000    0.5000    1.5000
    1.0000    1.5000    1.5000


prob_a = zeros(size(a)) + 1/12

prob_a = 4x3
    0.0833    0.0833    0.0833
    0.0833    0.0833    0.0833
```

```
    0.0833    0.0833    0.0833
    0.0833    0.0833    0.0833

[ar_idx_full, ~, ar_idx_of_unique] = unique(a)

ar_idx_full = 3x1
    0.5000
    1.0000
    1.5000

ar_idx_of_unique = 12x1
     1
     1
     2
     2
     3
     2
     1
     3
     1
     2

mt_idx_of_unique = reshape(ar_idx_of_unique, size(a))

mt_idx_of_unique = 4x3
     1     3     1
     1     2     2
     2     1     3
     2     3     3


accumarray(mt_idx_of_unique(:,1), prob_a(:,1))

ans = 2x1
    0.1667
    0.1667

accumarray(mt_idx_of_unique(:,2), prob_a(:,2))

ans = 3x1
    0.0833
    0.0833
    0.1667

accumarray(mt_idx_of_unique(:,3), prob_a(:,3))

ans = 3x1
    0.0833
    0.0833
    0.1667
```

1.1.7 Array Draw Random Index and Find Combinations or Permuations

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.1.7.1 Matlab Draw Random with and without Replacement

```
%Generate a matrix named foo, with limited numbers
```

```
rng(1234);
foo = unique((round((randn(5,1)+1)*100)));
disp(foo);
```

```
     5
    78
   154
   219
   232
```

```
% draw 10 random samples without replacement
index = randsample(1:length(foo), 4);
bar_rand_noreplace = foo(index,:);

% draw 1000 random samples with replacement
index = randsample(1:length(foo), 4, true);
bar_rand_replace = foo(index,:);

% Display
disp(table(bar_rand_noreplace, bar_rand_replace));
```

```
    bar_rand_noreplace    bar_rand_replace
    ------------------    ----------------

             5                   78
            78                  154
           154                  219
           232                  219
```

1.1.7.2 Matrix Meshgrid to Loop Permutated Vectors

Meshgrid to generate all permutations of arrays.

```
k = linspace(1,10,10);
kp = linspace(1,10,10);
z = linspace(0,1,10);

[kM kpM zM] = meshgrid(k,kp,z);
kMVec = kM(:);
kMpVec = kpM(:);
zMVec = zM(:);

outputVec = zeros(size(zMVec));
for a=1:length(zMVec)
     outputVec(a) = kMVec(a)+kMpVec(a)+zMVec(a);
end

outputTens = reshape(outputVec,size(kM));
disp(outputTens);
```

```
(:,:,1) =

     2     3     4     5     6     7     8     9    10    11
     3     4     5     6     7     8     9    10    11    12
     4     5     6     7     8     9    10    11    12    13
     5     6     7     8     9    10    11    12    13    14
     6     7     8     9    10    11    12    13    14    15
     7     8     9    10    11    12    13    14    15    16
```

```
     8     9    10    11    12    13    14    15    16    17
     9    10    11    12    13    14    15    16    17    18
    10    11    12    13    14    15    16    17    18    19
    11    12    13    14    15    16    17    18    19    20


(:,:,2) =

    2.1111    3.1111    4.1111    5.1111    6.1111    7.1111    8.1111    9.1111   10.1111   11.1111
    3.1111    4.1111    5.1111    6.1111    7.1111    8.1111    9.1111   10.1111   11.1111   12.1111
    4.1111    5.1111    6.1111    7.1111    8.1111    9.1111   10.1111   11.1111   12.1111   13.1111
    5.1111    6.1111    7.1111    8.1111    9.1111   10.1111   11.1111   12.1111   13.1111   14.1111
    6.1111    7.1111    8.1111    9.1111   10.1111   11.1111   12.1111   13.1111   14.1111   15.1111
    7.1111    8.1111    9.1111   10.1111   11.1111   12.1111   13.1111   14.1111   15.1111   16.1111
    8.1111    9.1111   10.1111   11.1111   12.1111   13.1111   14.1111   15.1111   16.1111   17.1111
    9.1111   10.1111   11.1111   12.1111   13.1111   14.1111   15.1111   16.1111   17.1111   18.1111
   10.1111   11.1111   12.1111   13.1111   14.1111   15.1111   16.1111   17.1111   18.1111   19.1111
   11.1111   12.1111   13.1111   14.1111   15.1111   16.1111   17.1111   18.1111   19.1111   20.1111


(:,:,3) =

    2.2222    3.2222    4.2222    5.2222    6.2222    7.2222    8.2222    9.2222   10.2222   11.2222
    3.2222    4.2222    5.2222    6.2222    7.2222    8.2222    9.2222   10.2222   11.2222   12.2222
    4.2222    5.2222    6.2222    7.2222    8.2222    9.2222   10.2222   11.2222   12.2222   13.2222
    5.2222    6.2222    7.2222    8.2222    9.2222   10.2222   11.2222   12.2222   13.2222   14.2222
    6.2222    7.2222    8.2222    9.2222   10.2222   11.2222   12.2222   13.2222   14.2222   15.2222
    7.2222    8.2222    9.2222   10.2222   11.2222   12.2222   13.2222   14.2222   15.2222   16.2222
    8.2222    9.2222   10.2222   11.2222   12.2222   13.2222   14.2222   15.2222   16.2222   17.2222
    9.2222   10.2222   11.2222   12.2222   13.2222   14.2222   15.2222   16.2222   17.2222   18.2222
   10.2222   11.2222   12.2222   13.2222   14.2222   15.2222   16.2222   17.2222   18.2222   19.2222
   11.2222   12.2222   13.2222   14.2222   15.2222   16.2222   17.2222   18.2222   19.2222   20.2222


(:,:,4) =

    2.3333    3.3333    4.3333    5.3333    6.3333    7.3333    8.3333    9.3333   10.3333   11.3333
    3.3333    4.3333    5.3333    6.3333    7.3333    8.3333    9.3333   10.3333   11.3333   12.3333
    4.3333    5.3333    6.3333    7.3333    8.3333    9.3333   10.3333   11.3333   12.3333   13.3333
    5.3333    6.3333    7.3333    8.3333    9.3333   10.3333   11.3333   12.3333   13.3333   14.3333
    6.3333    7.3333    8.3333    9.3333   10.3333   11.3333   12.3333   13.3333   14.3333   15.3333
    7.3333    8.3333    9.3333   10.3333   11.3333   12.3333   13.3333   14.3333   15.3333   16.3333
    8.3333    9.3333   10.3333   11.3333   12.3333   13.3333   14.3333   15.3333   16.3333   17.3333
    9.3333   10.3333   11.3333   12.3333   13.3333   14.3333   15.3333   16.3333   17.3333   18.3333
   10.3333   11.3333   12.3333   13.3333   14.3333   15.3333   16.3333   17.3333   18.3333   19.3333
   11.3333   12.3333   13.3333   14.3333   15.3333   16.3333   17.3333   18.3333   19.3333   20.3333


(:,:,5) =

    2.4444    3.4444    4.4444    5.4444    6.4444    7.4444    8.4444    9.4444   10.4444   11.4444
    3.4444    4.4444    5.4444    6.4444    7.4444    8.4444    9.4444   10.4444   11.4444   12.4444
    4.4444    5.4444    6.4444    7.4444    8.4444    9.4444   10.4444   11.4444   12.4444   13.4444
    5.4444    6.4444    7.4444    8.4444    9.4444   10.4444   11.4444   12.4444   13.4444   14.4444
    6.4444    7.4444    8.4444    9.4444   10.4444   11.4444   12.4444   13.4444   14.4444   15.4444
    7.4444    8.4444    9.4444   10.4444   11.4444   12.4444   13.4444   14.4444   15.4444   16.4444
    8.4444    9.4444   10.4444   11.4444   12.4444   13.4444   14.4444   15.4444   16.4444   17.4444
    9.4444   10.4444   11.4444   12.4444   13.4444   14.4444   15.4444   16.4444   17.4444   18.4444
```

```
   10.4444   11.4444   12.4444   13.4444   14.4444   15.4444   16.4444   17.4444   18.4444   19.4444
   11.4444   12.4444   13.4444   14.4444   15.4444   16.4444   17.4444   18.4444   19.4444   20.4444


(:,:,6) =

    2.5556    3.5556    4.5556    5.5556    6.5556    7.5556    8.5556    9.5556   10.5556   11.5556
    3.5556    4.5556    5.5556    6.5556    7.5556    8.5556    9.5556   10.5556   11.5556   12.5556
    4.5556    5.5556    6.5556    7.5556    8.5556    9.5556   10.5556   11.5556   12.5556   13.5556
    5.5556    6.5556    7.5556    8.5556    9.5556   10.5556   11.5556   12.5556   13.5556   14.5556
    6.5556    7.5556    8.5556    9.5556   10.5556   11.5556   12.5556   13.5556   14.5556   15.5556
    7.5556    8.5556    9.5556   10.5556   11.5556   12.5556   13.5556   14.5556   15.5556   16.5556
    8.5556    9.5556   10.5556   11.5556   12.5556   13.5556   14.5556   15.5556   16.5556   17.5556
    9.5556   10.5556   11.5556   12.5556   13.5556   14.5556   15.5556   16.5556   17.5556   18.5556
   10.5556   11.5556   12.5556   13.5556   14.5556   15.5556   16.5556   17.5556   18.5556   19.5556
   11.5556   12.5556   13.5556   14.5556   15.5556   16.5556   17.5556   18.5556   19.5556   20.5556


(:,:,7) =

    2.6667    3.6667    4.6667    5.6667    6.6667    7.6667    8.6667    9.6667   10.6667   11.6667
    3.6667    4.6667    5.6667    6.6667    7.6667    8.6667    9.6667   10.6667   11.6667   12.6667
    4.6667    5.6667    6.6667    7.6667    8.6667    9.6667   10.6667   11.6667   12.6667   13.6667
    5.6667    6.6667    7.6667    8.6667    9.6667   10.6667   11.6667   12.6667   13.6667   14.6667
    6.6667    7.6667    8.6667    9.6667   10.6667   11.6667   12.6667   13.6667   14.6667   15.6667
    7.6667    8.6667    9.6667   10.6667   11.6667   12.6667   13.6667   14.6667   15.6667   16.6667
    8.6667    9.6667   10.6667   11.6667   12.6667   13.6667   14.6667   15.6667   16.6667   17.6667
    9.6667   10.6667   11.6667   12.6667   13.6667   14.6667   15.6667   16.6667   17.6667   18.6667
   10.6667   11.6667   12.6667   13.6667   14.6667   15.6667   16.6667   17.6667   18.6667   19.6667
   11.6667   12.6667   13.6667   14.6667   15.6667   16.6667   17.6667   18.6667   19.6667   20.6667


(:,:,8) =

    2.7778    3.7778    4.7778    5.7778    6.7778    7.7778    8.7778    9.7778   10.7778   11.7778
    3.7778    4.7778    5.7778    6.7778    7.7778    8.7778    9.7778   10.7778   11.7778   12.7778
    4.7778    5.7778    6.7778    7.7778    8.7778    9.7778   10.7778   11.7778   12.7778   13.7778
    5.7778    6.7778    7.7778    8.7778    9.7778   10.7778   11.7778   12.7778   13.7778   14.7778
    6.7778    7.7778    8.7778    9.7778   10.7778   11.7778   12.7778   13.7778   14.7778   15.7778
    7.7778    8.7778    9.7778   10.7778   11.7778   12.7778   13.7778   14.7778   15.7778   16.7778
    8.7778    9.7778   10.7778   11.7778   12.7778   13.7778   14.7778   15.7778   16.7778   17.7778
    9.7778   10.7778   11.7778   12.7778   13.7778   14.7778   15.7778   16.7778   17.7778   18.7778
   10.7778   11.7778   12.7778   13.7778   14.7778   15.7778   16.7778   17.7778   18.7778   19.7778
   11.7778   12.7778   13.7778   14.7778   15.7778   16.7778   17.7778   18.7778   19.7778   20.7778


(:,:,9) =

    2.8889    3.8889    4.8889    5.8889    6.8889    7.8889    8.8889    9.8889   10.8889   11.8889
    3.8889    4.8889    5.8889    6.8889    7.8889    8.8889    9.8889   10.8889   11.8889   12.8889
    4.8889    5.8889    6.8889    7.8889    8.8889    9.8889   10.8889   11.8889   12.8889   13.8889
    5.8889    6.8889    7.8889    8.8889    9.8889   10.8889   11.8889   12.8889   13.8889   14.8889
    6.8889    7.8889    8.8889    9.8889   10.8889   11.8889   12.8889   13.8889   14.8889   15.8889
    7.8889    8.8889    9.8889   10.8889   11.8889   12.8889   13.8889   14.8889   15.8889   16.8889
    8.8889    9.8889   10.8889   11.8889   12.8889   13.8889   14.8889   15.8889   16.8889   17.8889
    9.8889   10.8889   11.8889   12.8889   13.8889   14.8889   15.8889   16.8889   17.8889   18.8889
   10.8889   11.8889   12.8889   13.8889   14.8889   15.8889   16.8889   17.8889   18.8889   19.8889
   11.8889   12.8889   13.8889   14.8889   15.8889   16.8889   17.8889   18.8889   19.8889   20.8889
```

```
(:,:,10) =

     3     4     5     6     7     8     9    10    11    12
     4     5     6     7     8     9    10    11    12    13
     5     6     7     8     9    10    11    12    13    14
     6     7     8     9    10    11    12    13    14    15
     7     8     9    10    11    12    13    14    15    16
     8     9    10    11    12    13    14    15    16    17
     9    10    11    12    13    14    15    16    17    18
    10    11    12    13    14    15    16    17    18    19
    11    12    13    14    15    16    17    18    19    20
    12    13    14    15    16    17    18    19    20    21
```

1.1.7.3 Given Integer Arrays, All Possible Combinations

given any sizes arrays, N of them, create all possible combinations

```
ar_it_a = 1:3;
ar_it_b = 1:2;
ar_it_c = 2:4;
ar_it_d = -1:-1:-2;
ar_it_e = 0.1;

cl_ar_all = {ar_it_a, ar_it_b, ar_it_c, ar_it_d, ar_it_e};
cl_mt_all = cl_ar_all;
[cl_mt_all{:}] = ndgrid(cl_ar_all{:});
mt_it_allcombo = cell2mat(cellfun(@(m) m(:), cl_mt_all, 'uni', 0));

disp(mt_it_allcombo)

    1.0000    1.0000    2.0000   -1.0000    0.1000
    2.0000    1.0000    2.0000   -1.0000    0.1000
    3.0000    1.0000    2.0000   -1.0000    0.1000
    1.0000    2.0000    2.0000   -1.0000    0.1000
    2.0000    2.0000    2.0000   -1.0000    0.1000
    3.0000    2.0000    2.0000   -1.0000    0.1000
    1.0000    1.0000    3.0000   -1.0000    0.1000
    2.0000    1.0000    3.0000   -1.0000    0.1000
    3.0000    1.0000    3.0000   -1.0000    0.1000
    1.0000    2.0000    3.0000   -1.0000    0.1000
    2.0000    2.0000    3.0000   -1.0000    0.1000
    3.0000    2.0000    3.0000   -1.0000    0.1000
    1.0000    1.0000    4.0000   -1.0000    0.1000
    2.0000    1.0000    4.0000   -1.0000    0.1000
    3.0000    1.0000    4.0000   -1.0000    0.1000
    1.0000    2.0000    4.0000   -1.0000    0.1000
    2.0000    2.0000    4.0000   -1.0000    0.1000
    3.0000    2.0000    4.0000   -1.0000    0.1000
    1.0000    1.0000    2.0000   -2.0000    0.1000
    2.0000    1.0000    2.0000   -2.0000    0.1000
    3.0000    1.0000    2.0000   -2.0000    0.1000
    1.0000    2.0000    2.0000   -2.0000    0.1000
    2.0000    2.0000    2.0000   -2.0000    0.1000
    3.0000    2.0000    2.0000   -2.0000    0.1000
    1.0000    1.0000    3.0000   -2.0000    0.1000
    2.0000    1.0000    3.0000   -2.0000    0.1000
    3.0000    1.0000    3.0000   -2.0000    0.1000
```

```
    1.0000    2.0000    3.0000   -2.0000    0.1000
    2.0000    2.0000    3.0000   -2.0000    0.1000
    3.0000    2.0000    3.0000   -2.0000    0.1000
    1.0000    1.0000    4.0000   -2.0000    0.1000
    2.0000    1.0000    4.0000   -2.0000    0.1000
    3.0000    1.0000    4.0000   -2.0000    0.1000
    1.0000    2.0000    4.0000   -2.0000    0.1000
    2.0000    2.0000    4.0000   -2.0000    0.1000
    3.0000    2.0000    4.0000   -2.0000    0.1000
```

1.1.8 Check Imaginary Numbers

back to Fan's Reusable Matlab Repository or Dynamic Asset Repository.

1.1.8.1 Basic Examples

```
rng(123);

% Imaginary array
ar_img = rand([1,7]) + 1i*rand([1,7]);

% Regular Array
ar_real = rand([1,10]);

% Combine arrays
ar_full = [ar_real ar_img];
ar_full = ar_full(randperm(length(ar_full)));
disp(ar_full);

  Columns 1 through 7

   0.6344 + 0.0000i   0.1755 + 0.0000i   0.5316 + 0.0000i   0.2861 + 0.4809i   0.7380 + 0.0000i   0.

  Columns 8 through 14

   0.2269 + 0.3921i   0.7245 + 0.0000i   0.8494 + 0.0000i   0.6110 + 0.0000i   0.4231 + 0.4386i   0.

  Columns 15 through 17

   0.3980 + 0.0000i   0.5513 + 0.3432i   0.7195 + 0.7290i


% real index
disp(~imag(ar_full));

   1   1   1   0   1   1   0   0   1   1   1   0   0   1   1   0   0


% Get Real and not real Components
disp(ar_full(imag(ar_full) == 0));

    0.6344    0.1755    0.5316    0.7380    0.1825    0.7245    0.8494    0.6110    0.5318    0.3980

disp(ar_full(imag(ar_full) ~= 0));

   0.2861 + 0.4809i   0.6965 + 0.6848i   0.2269 + 0.3921i   0.4231 + 0.4386i   0.9808 + 0.0597i   0.
```

1.2 Cells

1.2.1 List Comprehension with Cells

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.2.1.1 Find Index of Elements of String Cells in a larger String Cells

the function below returns the position of cl_st_param_keys in ls_st_param_key should only include in cl_st_param_keys strings that also exist in ls_st_param_key.

```
ls_st_param_key = {'fl_crra', 'fl_beta', ...
                   'fl_w', 'fl_r_save', ...
                   'fl_a_max', 'it_z_n', 'it_a_n'};

cl_st_param_keys = {'fl_w', 'fl_beta', 'it_z_n'};

cell2mat(cellfun(@(m) find(strcmp(ls_st_param_key, m)), ...
                 cl_st_param_keys, 'UniformOutput', false))

ans = 1x3
     3     2     6
```

1.2.1.2 Given Container of Arrays, Find Total Length of All Arrays for Selected Keys

```
cl_st_param_keys = {'fl_crra', 'fl_beta'};

param_tstar_map = containers.Map('KeyType','char', 'ValueType','any');
it_simu_vec_len = 5;

param_tstar_map('fl_crra') = linspace(1, 2, 5);
param_tstar_map('fl_beta') = linspace(0.94, 0.98, 10);
param_tstar_map('w') = linspace(1.1, 1.4, it_simu_vec_len);
param_tstar_map('r') = linspace(0.01, 0.04, it_simu_vec_len);

ar_it_array_len = cell2mat(cellfun(@(m) length(param_tstar_map(m)), ...
                           cl_st_param_keys, 'UniformOutput', false));

it_total_length = sum(ar_it_array_len);
disp(['ar_it_array_len: ' num2str(ar_it_array_len)])

ar_it_array_len: 5  10

disp(['it_total_length: ' num2str(it_total_length)])

it_total_length: 15
```

1.2.1.3 Given Container of Arrays, Find Min and Max of Each and Draw Random N sets

```
cl_st_param_keys = {'fl_crra', 'fl_beta'};

param_tstar_map = containers.Map('KeyType','char', 'ValueType','any');
it_simu_vec_len = 5;

param_tstar_map('fl_crra') = linspace(1, 2, 5);
param_tstar_map('fl_beta') = linspace(0.94, 0.98, 10);
param_tstar_map('w') = linspace(1.1, 1.4, it_simu_vec_len);
param_tstar_map('r') = linspace(0.01, 0.04, it_simu_vec_len);
```

```
rng(123);
it_simu_length = 20;
mt_param_rand = cell2mat(cellfun(@(m) ...
                            rand([it_simu_length,1]).*(max(param_tstar_map(m)) - min(param_tstar_map(
                            + min(param_tstar_map(m)), ...
                            cl_st_param_keys, 'UniformOutput', false));

tb_rand_draws = array2table(mt_param_rand, 'VariableNames', cl_st_param_keys);

disp(tb_rand_draws);
```

```
    fl_crra    fl_beta
    -------    -------

    1.6965     0.96538
    1.2861     0.97398
    1.2269     0.96898
    1.5513     0.96444
    1.7195      0.9689
    1.4231     0.95292
    1.9808     0.95447
    1.6848     0.94913
    1.4809     0.95175
    1.3921     0.96524
    1.3432     0.94368
     1.729     0.95735
    1.4386     0.95723
    1.0597     0.95975
     1.398     0.95703
     1.738     0.95249
    1.1825     0.95705
    1.1755     0.97574
    1.5316     0.97777
    1.5318     0.96007
```

1.2.2 All Possible Combinations of Multiple Arrays

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.2.2.1 Given Several Arrays of Possibly different Length in Container, all Possible combinations

```
param_tstar_map = containers.Map('KeyType','char', 'ValueType','any');
param_tstar_map('a') = linspace(1, 5, 5);
param_tstar_map('b') = linspace(0.87, 0.97, 6);
param_tstar_map('c') = linspace(0, 0.5, 10);

cl_st_param_keys = {'a','c'};
cl_ar_param_subset_values = values(param_tstar_map, {'a','c'});

cl_mt_all = cl_ar_param_subset_values;
[cl_mt_all{:}] = ndgrid(cl_ar_param_subset_values{:});
mt_param_vals_combi = cell2mat(cellfun(@(m) m(:), cl_mt_all, 'uni', 0));

tb_all_combi = array2table(mt_param_vals_combi, 'VariableNames', cl_st_param_keys);

disp(tb_all_combi);
```

```
a       c
-    --------

1           0
2           0
3           0
4           0
5           0
1    0.055556
2    0.055556
3    0.055556
4    0.055556
5    0.055556
1     0.11111
2     0.11111
3     0.11111
4     0.11111
5     0.11111
1     0.16667
2     0.16667
3     0.16667
4     0.16667
5     0.16667
1     0.22222
2     0.22222
3     0.22222
4     0.22222
5     0.22222
1     0.27778
2     0.27778
3     0.27778
4     0.27778
5     0.27778
1     0.33333
2     0.33333
3     0.33333
4     0.33333
5     0.33333
1     0.38889
2     0.38889
3     0.38889
4     0.38889
5     0.38889
1     0.44444
2     0.44444
3     0.44444
4     0.44444
5     0.44444
1         0.5
2         0.5
3         0.5
4         0.5
5         0.5
```

1.2.3 Combine Cells Together

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.2.3.1 String Combine with string cell

```
ls_st_param_key = {'fl_crra', 'fl_beta', ...
                   'fl_w', 'fl_r_save', ...
                   'fl_a_max', 'it_z_n', 'it_a_n'};

cl_st_param_keys = {'fl_wad', 'fl_betart', 'it_z_nfg'};

st_param = 'asdjfl';

[{st_param}, ls_st_param_key, cl_st_param_keys]

ans =
    {'asdjfl'}    {'fl_crra'}    {'fl_beta'}    {'fl_w'}    {'fl_r_save'}    {'fl_a_max'}    {'it_z_
```

1.2.4 Nested Cells

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.2.4.1 Nested Cells and access

```
cl_st_param_keys = {'fl_crra', 'fl_beta'};

it_simu_vec_len = 3;
clns_parm_tstar = cell([4,1]);
clns_parm_tstar{1} = {'fl_crra', 'CRRA', linspace(1, 2, it_simu_vec_len)};
clns_parm_tstar{2} = {'fl_beta', 'Discount', linspace(0.94, 0.98, it_simu_vec_len)};
clns_parm_tstar{3} = {'w', 'Wage', linspace(1.1, 1.4, it_simu_vec_len)};
clns_parm_tstar{4} = {'r', 'Save Interest', linspace(0.01, 0.04, it_simu_vec_len)};

disp(clns_parm_tstar(1));

    {1x3 cell}

disp(clns_parm_tstar{1}{1})

fl_crra

disp(clns_parm_tstar{1}{2});

CRRA

disp(clns_parm_tstar{1}{3});

    1.0000    1.5000    2.0000
```

1.3 Characters and Strings

1.3.1 Basic String Operations

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.3.1.1 Combine String, Numeric values etc, Single and Double Quotes

Convert a string array into a single string, note the double quotes, and the auto space between:

```
st_a = "another string";
ar_st = ["abc", num2str(2), "opq", st_a];
```

```
disp(strjoin(ar_st));
```

```
abc 2 opq another string
```

If we do not want to have spaces between words, the second parameter for strjoin allows for string connectors:

```
st_a = "another string";
ar_st = ["abc", num2str(2), "opq", st_a];
disp(strjoin(ar_st, ""));
```

```
abc2opqanother string
```

With single quotes, the str element is not an array, so does not need strjoin, but not need to have spaces:

```
st_a = 'another string';
str = ['abc ', num2str(2), ' opq ', st_a];
disp((str));
```

```
abc 2 opq another string
```

1.3.1.2 Construct String Array and String Elements of String Array

In the example below, we have a number of strings we want to put inside a string array, then join with strjoin, but two of the strings need to be constructed as strings first. Note below that double quoates are own strings, single quotes in brackets constructing additional strings.

```
st_a = "another string";
ar_st = strjoin(...
    ["Completed SNW_DS_MAIN", ...
     ['SNW_MP_PARAM=' num2str(123.345)], ...
     ['SNW_MP_CONTROL=' num2str(678.90)], ...
     st_a...
    ], ";");
disp(ar_st);
```

```
Completed SNW_DS_MAIN;SNW_MP_PARAM=123.345;SNW_MP_CONTROL=678.9;another string
```

1.3.1.3 Paste Join Strings Together with Separator

Join strings together with separator, this is similar to the paste0 function in R.

```
ar_st = ["abc", "efg", "opq"];
disp(strjoin(ar_st, '-'));
```

```
abc-efg-opq
```

1.3.1.4 Combine Char with Numeric Value

Compose a string with words and numerical values

```
st_title = strcat("Figure Title ", ...
    "(", ...
    "threedeci=%.3f,", ...
    "twodeci=%.2f,", ...
    "int=%.0f", ...
    ")");
ar_params = 123.4567 + zeros(1,3);
st_combo = compose(st_title, ar_params);
disp(st_combo);
```

```
Figure Title (threedeci=123.457,twodeci=123.46,int=123)
```

1.3.1.5 Change File Name MLX to M

```
st_file_name_mlx = 'continuous_differentiable.mlx';
at_st_split_file_name = split(st_file_name_mlx, ".");
st_file_name_m = strcat(at_st_split_file_name{1}, '_m.m');
disp(st_file_name_m);

continuous_differentiable_m.m
```

1.3.2 String Manipulations with Arrays

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.3.2.1 String Array

Three title lines, with double quotes:

```
ar_st_titles = ["Title1","Title2","Title3"]';
disp(ar_st_titles);

    "Title1"
    "Title2"
    "Title3"
```

Three words, joined together, now single quotes, this creates one string, rather than a string array:

```
st_titles = ['Title1','Title2','Title3'];
disp(st_titles);

Title1Title2Title3
```

1.3.2.2 String Cell Array

Create a string array:

```
ar_st_title_one = {'Title One Line'};
ar_st_titles = {'Title1','Title2','Title3'};
disp(ar_st_title_one);

    'Title One Line'

disp(ar_st_titles);

    'Title1'    'Title2'    'Title3'
```

Add to a string array:

```
ar_st_titles{4} = 'Title4';
disp(ar_st_titles);

    'Title1'    'Title2'    'Title3'    'Title4'
```

Update one of the strings:

```
ar_st_title_one{1} = strcat('log(', ar_st_title_one{1},')');
ar_st_titles{1} = strcat('log(', ar_st_titles{1},')');
disp(ar_st_title_one);

    'log(Title One Line)'

disp(ar_st_titles);
```

```
    'log(Title1)'    'Title2'    'Title3'    'Title4'
```

1.3.2.3 Duplicate String

```
it_duplicate_n = 10;
disp(repmat({'String'}, [1, it_duplicate_n]));
```

```
    'String'    'String'    'String'    'String'    'String'    'String'    'String'    'String'
```

1.3.2.4 String Join to form Single Element

using char() is safe

```
st_var_name = "abc"

st_var_name = "abc"

st_var_name = [st_var_name ' percentile values']

st_var_name = 1x2 string array
"abc"        " percentile values"

strjoin(st_var_name)

ans = "abc  percentile values"


st_var_name = "abc"

st_var_name = "abc"

st_var_name = [char(st_var_name) ' percentile values']

st_var_name = 'abc percentile values'


st_var_name = 'abc'

st_var_name = 'abc'

st_var_name = [char(st_var_name) ' percentile values']

st_var_name = 'abc percentile values'
```

1.3.2.5 String Join dash (Paste)

This is similar to R's paste function:

```
st_var_name = "abc";

st_var_name = "abc"

st_var_name = [st_var_name, 'efg', 'mqo'];

st_var_name = 1x3 string array
"abc"        "efg"        "mqo"

disp(strjoin(st_var_name, "_"));

ans = "abc_efg_mqo"
```

```
disp(strjoin(st_var_name, ","));
```

1.3.2.6 Numeric Array to String without Space

String replace

```
ar_it_test_grp = [3, 8, 9];
strrep(num2str(ar_it_test_grp), '  ', '_')

ans = '3_8_9'
```

1.3.2.7 Substring replace in Cell Array

```
ar_st_cells = {'shock=0.35','shock=0.40','shock=0.46'};
ar_st_updated_cells = strrep(ar_st_cells, 'shock', '$\epsilon$');
disp(ar_st_updated_cells);

    '$\epsilon$=0.35'    '$\epsilon$=0.40'    '$\epsilon$=0.46'
```

1.3.2.8 Find position of String in String Cell

```
ls_st_param_key = {'fl_crra', 'fl_beta', ...
                   'fl_w', 'fl_r_save', ...
                   'fl_a_max', 'it_z_n', 'it_a_n'};
st_param_key = 'fl_a_max';
find(strcmp(ls_st_param_key, st_param_key))

ans = 5
```

1.3.2.9 Find the positions of String Cells in Full String Cells

```
ls_st_param_key = {'fl_crra', 'fl_beta', ...
                   'fl_w', 'fl_r_save', ...
                   'fl_a_max', 'it_z_n', 'it_a_n'};

cl_st_param_keys = {'fl_w', 'fl_beta', 'it_z_n'};

cell2mat(cellfun(@(m) find(strcmp(ls_st_param_key, m)), ...
                 cl_st_param_keys, 'UniformOutput', false))

ans = 1x3
     3     2     6


find(strcmp(ls_st_param_key, st_param_key))

ans = 5
```

1.3.2.10 Cell to string Paste and Replace dash

```
cl_st_param_keys = {'fl_crra', 'fl_beta'};
display(strrep(strjoin(cl_st_param_keys, '-'), '_', '\_'));

fl\_crra-fl\_beta
```

1.3.3 Concatenate Strings Arrays with Numbers and Number Arrays with Strings

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.3.3.1 Combine A String Array with A Numeric Array using Compose

String array and numeric array, combine together using the compose function, and test different formatting functions. Formating with leading empty spaces, leading zeros, and convert to integer or not.

```
st_titles = ["%.3f",   "%.1f",   "%.0f";...
             "%6.3f",  "%6.1f",  "%6.0f";...
             "%06.3f", "%06.1f", "%06.0f"];
ar_params = 123.4567890 + zeros(3,3);
st_combo = compose(st_titles, ar_params);
disp(st_combo);
```

```
    "123.457"    "123.5"     "123"
    "123.457"    " 123.5"    "   123"
    "123.457"    "0123.5"    "000123"
```

1.3.3.2 Title from an Array of Values

There is a vector of parameter values and a vector of names for these parameter values, I want to include these in the title of a figure with the same decimal formating.

```
% Inputs
rng(123);
ar_params = rand(1,3);
ar_st_parms_names = ["param1", "param2", "param3"];
st_rounding = '.2f';
st_title_main = "this is the figure title";
% Rounding and combining
ar_st_params = strcat(ar_st_parms_names, compose(strcat("=%", st_rounding), ar_params));
% Generate a Single String that is comma separated:
st_param_pasted = strjoin(ar_st_params, ', ');
% Generate title with parameters
st_title_wth_params = strcat(st_title_main, ' (', st_param_pasted, ')');
% Display:
disp(st_title_wth_params);
```

```
this is the figure title (param1=0.70, param2=0.29, param3=0.23)
```

1.3.3.3 Combine String with Numeric Array

Example 1:

```
ar_fl_abc1 = [0.4 0.1 0.25 0.3 0.4];
disp([num2str(ar_fl_abc1', 'zw=%3.2f;'), num2str(ar_fl_abc1', 'zr=%3.2f')]);
```

Example 2:

```
close all;

rng(123);
ar_z_r_borr_mesh_wage = rand([1,5]);
ar_z_wage_mesh_r_borr = rand([1,5]);
ar_it_rows = round(rand([1,5])*10);
cl_st_full_rowscols = cellstr([num2str(ar_z_r_borr_mesh_wage', 'zr=%3.2f;'), ...
                               num2str(ar_z_wage_mesh_r_borr', 'zw=%3.2f')]);
cl_col_names = strcat('zi=', num2str(ar_it_rows([1,3,5])'), ':', cl_st_full_rowscols([1,3,5]));
```

```
disp(ar_z_r_borr_mesh_wage);

    0.6965    0.2861    0.2269    0.5513    0.7195

disp(ar_z_wage_mesh_r_borr);

    0.4231    0.9808    0.6848    0.4809    0.3921

disp(cl_st_full_rowscols);

    'zr=0.70;zw=0.42'
    'zr=0.29;zw=0.98'
    'zr=0.23;zw=0.68'
    'zr=0.55;zw=0.48'
    'zr=0.72;zw=0.39'

disp(cl_col_names);

    'zi=3:zr=0.70;zw=0.42'
    'zi=4:zr=0.23;zw=0.68'
    'zi=4:zr=0.72;zw=0.39'
```

1.3.3.4 Combine Number with String Cell Array

We have a string cell array we created from the previous section, now append numbers to it

```
% Append Common Numbers
cl_col_names_append = strcat(cl_col_names, '-String-Cell-With-Numeric-', num2str(123));
disp(cl_col_names_append);

    'zi=3:zr=0.70;zw=0.42-String-Cell-With-Numeric-123'
    'zi=4:zr=0.23;zw=0.68-String-Cell-With-Numeric-123'
    'zi=4:zr=0.72;zw=0.39-String-Cell-With-Numeric-123'
```

1.3.3.5 Combine Numeric Array with String Cell Array

Append an array of numeric values

```
% Append Common Numbers
cl_col_names_append = strcat(cl_col_names, '-String-Cell-With-Numeric-Array-', ...
    num2str(transpose(1:length(cl_col_names))));
disp(cl_col_names_append);

    'zi=3:zr=0.70;zw=0.42-String-Cell-With-Numeric-Array-1'
    'zi=4:zr=0.23;zw=0.68-String-Cell-With-Numeric-Array-2'
    'zi=4:zr=0.72;zw=0.39-String-Cell-With-Numeric-Array-3'
```

1.3.3.6 Convert Numeric Array to String, Apeend Prefix to all elements.

```
ar_fl_abc1 = [0.4 0.1 0.25 0.3 0.4];
ar_st_wth_prefix = strcat('row=', string(ar_fl_abc1));
disp(ar_st_wth_prefix);

% Does Array Exist in Longer Array as Subset
ar_abc1 = [0.4 0.1 0.25 0.3 0.4];
ar_abc2 = [0.4 0.1 0.2 0.3 0.4];
ar_efg = [0.1 0.2 0.3 0.4 0.1 0.2 0.3 0.4 0.1 0.2 0.3 0.4 0.1 0.2 0.3 0.4];
st_abc1 = strjoin(string(num2str(ar_abc1)));
st_abc2 = strjoin(string(num2str(ar_abc2)));
st_efg = strjoin(string(num2str(ar_efg)));
```

```
contains(st_efg, st_abc1)
contains(st_efg, st_abc2)

% Display Convert to String
fprintf('Display string [%s]', num2str([1,2,3]));
fprintf('Display string [%s]', num2str(1.1));
fprintf('Display string [%s]', 'abc');
```

1.4 Map Containers

1.4.1 Container Map Basics

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.4.1.1 Container Integer Keys

Given some matrix, I want to store matrix column names as well as labels for what each row and column correspond to. Achieve this using a cell array of container maps. Cell dimensions correspond to the first, second, etc dimensions, any dimension specific information can be stored in this fashion.

Can access information asssociated with the label value of the row values:

```
% Define Matrix Row and Column and additional dimension information
cl_mp_datasetdesc = {};
cl_mp_datasetdesc{1} = containers.Map({'dim', 'name', 'labval'}, {1, 'kids', [0,1,2,3]});
cl_mp_datasetdesc{2} = containers.Map({'dim', 'name', 'labval'}, {2, 'age', [18,19,20]});
% get variable labels for the first dimension (rows)
disp([...
    string(['dim 1 var name:' cl_mp_datasetdesc{1}('name') ]), ...
    string(['dim 2 var name:' cl_mp_datasetdesc{2}('name') ])...
    ]);

    "dim 1 var name:kids"    "dim 2 var name:age"
```

1.4.1.2 Is Key In Container

```
param_map_a = containers.Map('KeyType','char', 'ValueType','any');
param_map_a('fl_b_bd') = -3;
param_map_a('fl_w_max') = 50;
param_map_a('fl_kp_min') = 0;
param_map_a('it_w_i') = 100;

disp([...
    string(['has it_w_i as key? ' num2str(isKey(param_map_a, 'it_w_i'))]), ...
    string(['has it_w_i1 as key? ' num2str(isKey(param_map_a, 'it_w_i1'))]) ...
    ]);

    "has it_w_i as key? 1"    "has it_w_i1 as key? 0"
```

1.4.1.3 Container Key Loop

Generate new container key within loop dynamically

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

rng(123);
for st_cur = ["abc", "efg", "qqq"]

    if (strcmp(st_cur, "abc"))
```

```
       data = rand([1,1]);
    elseif (strcmp(st_cur, "efg"))
       data = 123.123;
    elseif (strcmp(st_cur, "qqq"))
       data = -123;
    end

    % common function
    fl_sh_0p1pc_j = data*2 + 1;
    fl_sh_5pc_j = data/2 - 1;

    % generate map keys
    st_key_sh_0p1pc_j = strjoin([st_cur, 'sh_0p1pc_j'], "_");
    st_key_sh_5pc_j = strjoin([st_cur, 'sh_5pc_j'], "_");

    % store
    param_map_a(st_key_sh_0p1pc_j) = fl_sh_0p1pc_j;
    param_map_a(st_key_sh_5pc_j) = fl_sh_5pc_j;

end

disp([...
    string(['param_map_a.keys:' param_map_a.keys]), ...
    string(['param_map_a.values:' string(param_map_a.values)]) ...
    ]);
```

```
  Columns 1 through 7

    "param_map_a.keys:"    "abc_sh_0p1pc_j"    "abc_sh_5pc_j"    "efg_sh_0p1pc_j"    "efg_sh_5pc_j"

  Columns 8 through 14

    "param_map_a.values:"    "2.3929"    "-0.65177"    "247.246"    "60.5615"    "-245"    "-62.5"
```

1.4.2 Container Map Display Key and Values and Subseting

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.4.2.1 Print Keys and Values

Define container:

```
% Define Container
param_map = containers.Map('KeyType','char', 'ValueType','any');
param_map('share_unbanked_j') = 12;
param_map('equi_r_j') = 2;
param_map('equi_w_j') = 'abc';
param_map('equi_P_j') = 1.2;
```

Print the key and values of the container:

```
param_map_keys = keys(param_map);
param_map_vals = values(param_map);
for i = 1:length(param_map)
    st_key = param_map_keys{i};
    ob_val = param_map_vals{i};
    st_display = strjoin(['pos =' num2str(i) '; key =' string(st_key) '; val =' string(ob_val)]);
    disp(st_display);
end
```

```
pos = 1 ; key = equi_P_j ; val = 1.2
pos = 2 ; key = equi_r_j ; val = 2
pos = 3 ; key = equi_w_j ; val = abc
pos = 4 ; key = share_unbanked_j ; val = 12
```

1.4.2.2 Select of Subset of Key/Values from a Container Map

There is a larger container map, I want to create a new container map, that keeps a subset of the keys/values of the full container map.

```
% Original Container map
param_map = containers.Map('KeyType','char', 'ValueType','any');
param_map('equi_r_j') = 0.05;
param_map('equi_w_j') = 1.05;
param_map('equi_P_j') = 1;
% To select a subset of keys
ls_st_keys_select = {'equi_w_j', 'equi_P_j'};
% Select
param_map_subset = containers.Map(ls_st_keys_select, values(param_map, ls_st_keys_select));
% display
disp(param_map_subset.keys);

    'equi_P_j'    'equi_w_j'

disp(param_map_subset.values);

    [1]    [1.0500]
```

1.4.3 Generate Container Maps

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.4.3.1 Generate a Container map with any time of data

Create a container map with float, int, string, and matrix

```
close all;
clear all;

% Create A Map with String Keys and any values
param_map = containers.Map('KeyType','char', 'ValueType','any');
param_map('share_unbanked_j') = 12;
param_map('equi_r_j') = 2;
param_map('equi_w_j') = 'abc';
param_map('equi_P_j') = zeros(2,3);
disp(param_map.keys);

    'equi_P_j'    'equi_r_j'    'equi_w_j'    'share_unbanked_j'

disp(param_map.values);

    [2x3 double]    [2]    'abc'    [12]
```

1.4.3.2 Access Multiple Values of a container map

Values been accessed need to be of the same type

```
% Parameter Dealing from Map
params_group = values(param_map, {'share_unbanked_j', 'equi_r_j'});
```

```
[equi_P_j, equi_r_j] = params_group{:};
disp(['equi_P_j:' num2str(equi_P_j) ', equi_r_j:' num2str(equi_r_j)]);

equi_P_j:12, equi_r_j:2

% Access Scalar Elements of Map and Convert the Array
disp(cell2mat(values(param_map, {'share_unbanked_j', 'equi_r_j'})));

    12     2
```

Create a container map of color values and generate a array of color choices:

```
% Container map with three colors
mp_colors = containers.Map('KeyType', 'char', 'ValueType', 'any');
mp_colors('blue')   = [57 106 177]./255;
mp_colors('red')    = [204 37 41]./255;
mp_colors('black')  = [83 81 84]./255;
% An selection array
ar_st_colors_pick = {'blue', 'blue', 'red', 'black', 'blue'};
ar_colors = values(mp_colors, ar_st_colors_pick);
% Print selected colors
celldisp(ar_colors);


ar_colors{1} =

    0.2235    0.4157    0.6941



ar_colors{2} =

    0.2235    0.4157    0.6941



ar_colors{3} =

    0.8000    0.1451    0.1608



ar_colors{4} =

    0.3255    0.3176    0.3294



ar_colors{5} =

    0.2235    0.4157    0.6941
```

1.4.4 Container Map Example Overriding

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

1.4.4.1 Update Container Map

There is one map with values, Container Map A. There is another container Map, Container Map B. Container Maps A and B share keys. For keys that exist in B and A, B Key value supercede values for the same keys in A. For new keys in B, they superced keys in A.

```
param_map_a = containers.Map('KeyType','char', 'ValueType','any');
param_map_a('fl_b_bd') = -3;
param_map_a('fl_w_max') = 50;
param_map_a('fl_kp_min') = 0;
param_map_a('it_w_i') = 100;

param_map_b = containers.Map('KeyType','char', 'ValueType','any');
param_map_b('fl_w_max') = 77;
param_map_b('fl_kp_min') = -231;
param_map_b('it_z_n') = 5;
param_map_b('fl_z_mu') = 0;

param_map_c = [param_map_a; param_map_b];
param_map_c.keys

ans =
    {'fl_b_bd'}    {'fl_kp_min'}    {'fl_w_max'}    {'fl_z_mu'}    {'it_w_i'}    {'it_z_n'}

param_map_c.values

ans =
    {[-3]}    {[-231]}    {[77]}    {[0]}    {[100]}    {[5]}
```

Chapter 2

Functions

2.1 varargin Default Parameters

2.1.1 varargin as a Function Parameter

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

2.1.1.1 Call Function with Two Parameters and Defaults

Call function below without overriding

```
ff_varargin(1.1, 2)
```

```
fl_a = 1.1000
it_b = 2
mt_data = 3x4
    0.6965    0.5513    0.9808    0.3921
    0.2861    0.7195    0.6848    0.3432
    0.2269    0.4231    0.4809    0.7290

ar_st_colnames = 1x4 string array
"col1"       "col2"       "col3"       "col4"

ar_st_rownames = 1x4 string array
"row1"       "row2"       "row3"       "row4"

st_table_name = "Table Name"
it_table_ctr = 1021
```

2.1.1.2 Override Subset of Varargin

```
rng(789);
mt_data_ext = rand(5,2);
ar_st_colnames = ["col1", "col2"];
ar_st_rownames = ["row1", "row2", "row3", "row4", "row5"];
ff_varargin(param_map, support_map, mt_data_ext, ar_st_colnames, ar_st_rownames);
```

```
fl_a =
  Map with properties:

        Count: 2
      KeyType: char
    ValueType: any
```

```
it_b =
  Map with properties:

        Count: 1
      KeyType: char
    ValueType: any

mt_data = 5x2
    0.3233    0.7589
    0.2302    0.0106
    0.7938    0.0247
    0.6244    0.1110
    0.9754    0.5381

ar_st_colnames = 1x2 string array
"col1"       "col2"

ar_st_rownames = 1x5 string array
"row1"       "row2"       "row3"       "row4"       "row5"

st_table_name = "Table Name"
it_table_ctr = 1021
```

2.1.1.3 Function with varargin as Inputs

Basic default structure with varargin.

```
function ff_varargin(fl_a, it_b, varargin)
% This is an example of how to use varargin:
% 1. includes array matrix
% 2. includes array
% 3. includes scalar
% 4. includes string
% 5. includes cell array

%% Catch Error
cl_params_len = length(varargin);
if cl_params_len > 5
    error('ff_mat2tab:TooManyOptionalParameters', ...
          'allows at most 5 optional parameters');
end

%% Default Folder Parameters
% by default all go to Sandbox folder with sub folders by dates
rng(123);
mt_data = rand(3,4);
% String array requires double quotes
ar_st_colnames = ["col1", "col2", "col3", "col4"];
ar_st_rownames = ["row1", "row2", "row3", "row4"];
% Others
st_table_name = "Table Name";
it_table_ctr = 1021;
cl_params = {mt_data ar_st_colnames ar_st_rownames ...
                  st_table_name it_table_ctr};

%% Parse Parameters
% numvarargs is the number of varagin inputted
[cl_params{1:cl_params_len}] = varargin{:};
```

```
% cell2mat(cl_params(1)) works with array
mt_data = cell2mat(cl_params(1));
% The structure below works with cell array
ar_st_colnames = cl_params{2};
ar_st_rownames = cl_params{3};
% Others
st_table_name = cl_params{4};
it_table_ctr = cl_params{5};

% Build Basic Matlab Table
% Suppose we want to store matrix results in a table,
% there are Q columns and N rows, The Q columns each is a different variable.
fl_a
it_b
mt_data
ar_st_colnames
ar_st_rownames
st_table_name
it_table_ctr

end
```

2.1.2 Map Based Default Parameter Structure with varargin

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

2.1.2.1 Call Function with Default Parameters

Call function below without overriding

```
ff_defaultmap()
```

```
    'c_gap'    'c_max'    'c_min'    'c_min_for_util'    'fl_crra'    'it_rown'    'st_single_double

    [1.0000e-03]    [60]    [1.0000e-03]    [1.0000e-03]    [1.5000]    [100]    'double'

Elapsed time is 0.000896 seconds.
```

2.1.2.2 Call Function overriding some Parameters

```
param_map = containers.Map('KeyType','char', 'ValueType','any');
param_map('fl_w_max') = 1.11;
param_map('it_w_i') = 2.22;

support_map = containers.Map('KeyType','char', 'ValueType','any');
support_map('bl_display') = true;
ff_defaultmap(param_map, support_map)
```

```
    'c_gap'    'c_max'    'c_min'    'c_min_for_util'    'fl_crra'    'fl_w_max'    'it_rown'    'it

    [1.0000e-03]    [60]    [1.0000e-03]    [1.0000e-03]    [1.5000]    [1.1100]    [100]    [2.2200

Elapsed time is 0.000667 seconds.
```

2.1.2.3 Function with Map Defaults and Overriding

This default parameter style is fairly succinct, allows for program testability, and easy adjustments/addition of additional parameters to models.

```
function ff_defaultmap(varargin)

% Parameters
params_len = length(varargin);
if params_len > 3
    error('ff_defaultmap:Can only have 3 container map parameters');
end
bl_input_override = 0;
if (params_len == 3)
    bl_input_override = varargin{3};
end

% Defaults
if (bl_input_override)
    % this relies on externally generated parameters, defaults do not have to be generated
    % if this file has to be invoked many times, then this saves time by avoiding
    % regenerating defaults over and over again
    [param_map, support_map, ~] = varargin{:};
else
    param_map = containers.Map('KeyType','char', 'ValueType','any');
    param_map('fl_crra') = 1.5;
    param_map('c_min') = 0.001;
    param_map('c_min_for_util') = 0.001;
    param_map('c_gap') = 10^-3;
    param_map('c_max') = 60;
    param_map('it_rown') = 100;
    param_map('st_single_double') = 'double';

    support_map = containers.Map('KeyType','char', 'ValueType','any');
    support_map('bl_display') = true;
    support_map('bl_graph') = true;
    support_map('bl_graph_onebyones') = true;
    support_map('bl_time') = true;
    support_map('bl_profile') = false;
    support_map('st_profile_path') = [pwd '/profile'];
    default_maps = {param_map, support_map};
end

% Parse Parameters
% see: C:\Users\fan\M4Econ\support\dtype\map_override.m
[default_maps{1:params_len}] = varargin{:};
param_map = [param_map; default_maps{1}];
support_map = [support_map; default_maps{2}];

params_group = values(param_map, {'fl_crra', 'c_min', 'c_min_for_util', 'c_gap', 'c_max'});
[fl_crra, c_min, c_min_for_util, c_gap, c_max] = params_group{:};
params_group = values(param_map, {'it_rown'});
[it_rown] = params_group{:};
params_group = values(param_map, {'st_single_double'});
[st_single_double] = params_group{:};

% support
params_group = values(support_map, {'bl_display', 'bl_graph', 'bl_graph_onebyones'});
[bl_display, bl_graph, bl_graph_onebyones] = params_group{:};
params_group = values(support_map, {'bl_time', 'bl_profile', 'st_profile_path'});
[bl_time, bl_profile, st_profile_path] = params_group{:};

% Tic toc starts
```

```
if (bl_time); tic; end

% Print Parameters
if (bl_display)
    disp(param_map.keys);
    disp(param_map.values);
end

% Profile On
if (bl_profile)
    close all;
    profile off;
    profile on;
end

%% Profiling
if (bl_profile)
    profile off
    profile viewer
    profsave(profile('info'), st_profile_path);
end

if (bl_time); toc; end

end
```

Chapter 3

Graphs

3.1 Figure Components

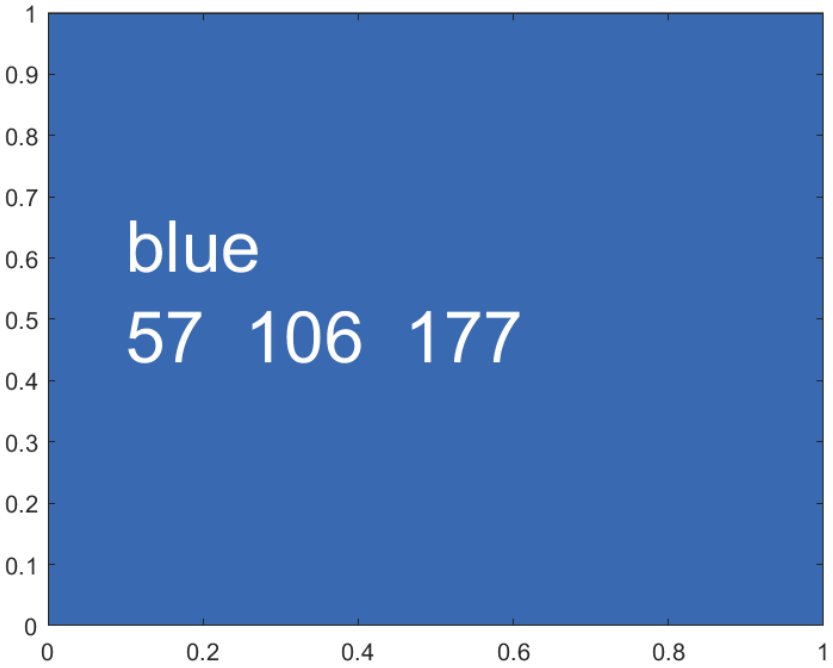
3.1.1 Matlab Graph Safe Colors for Web, Presentation and Publications Examples

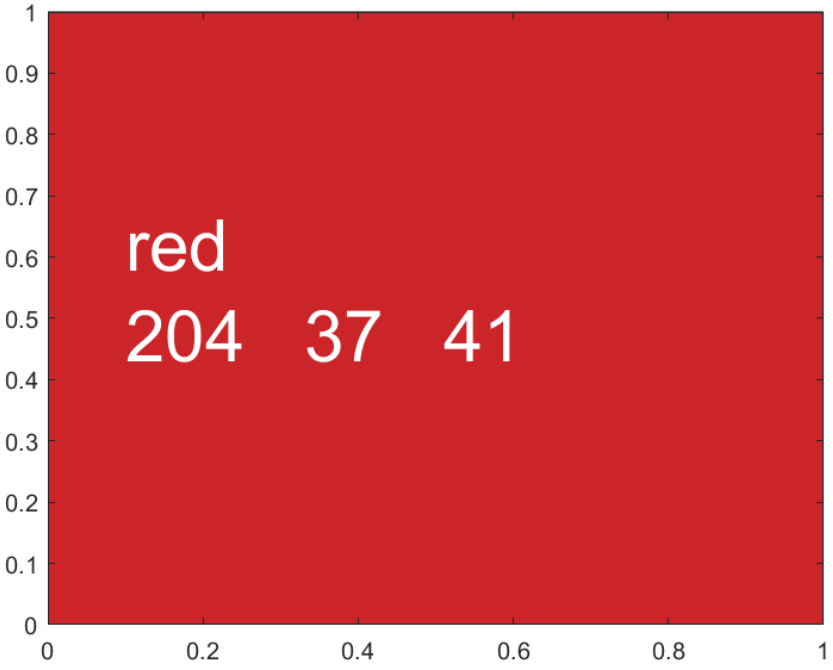
Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

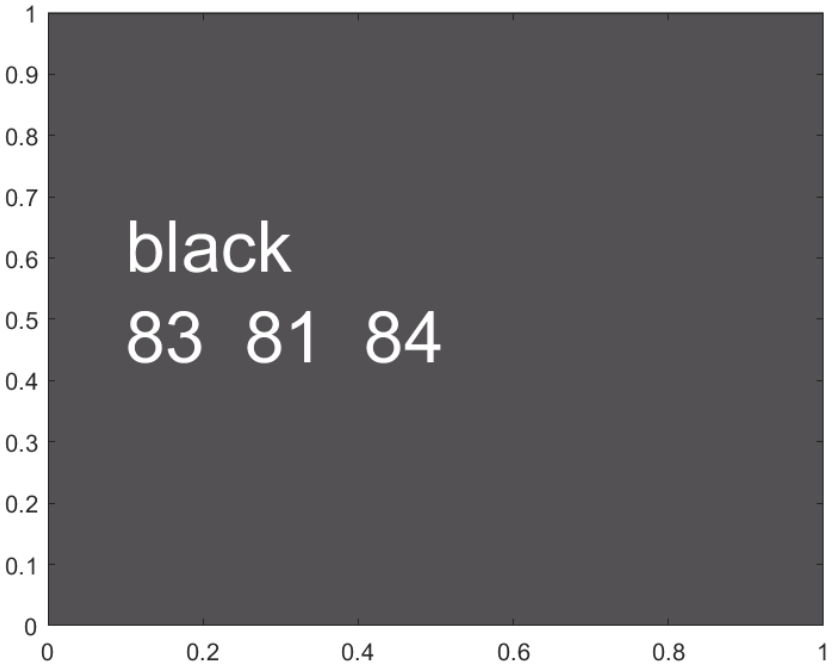
3.1.1.1 Good Colors to Use Darker

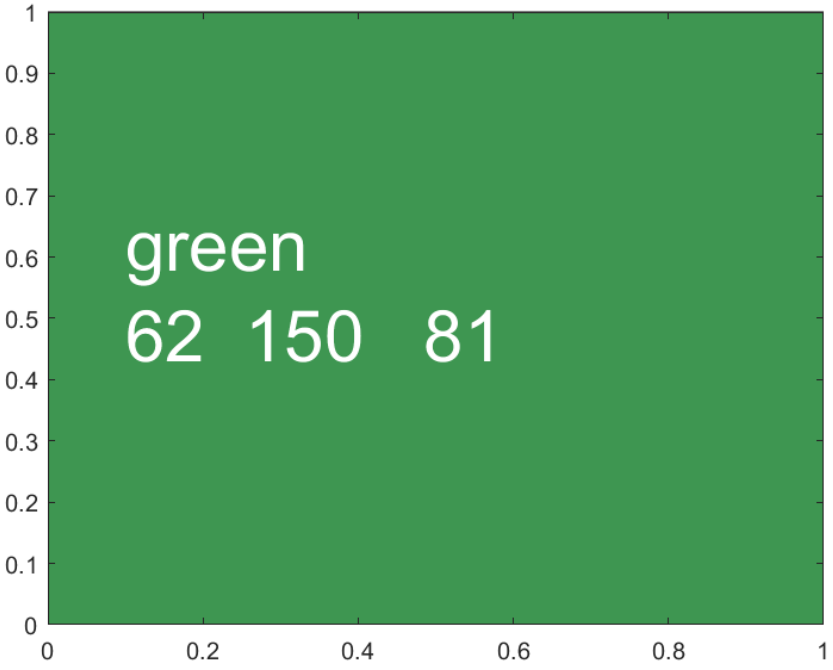
Nice darker light colors to use in matlab.

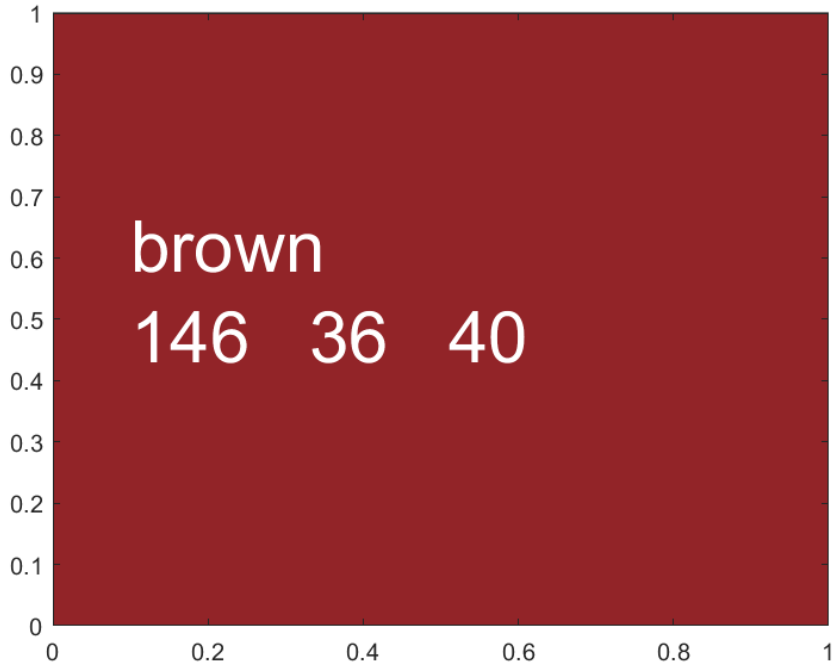
```
close all

blue = [57 106 177]./255;
red = [204 37 41]./255;
black = [83 81 84]./255;
green = [62 150 81]./255;
brown = [146 36 40]./255;
purple = [107 76 154]./255;
cl_colors = {blue, red, black, ...
             green, brown, purple};
cl_str_clr_names = ["blue", "red", "black", "green", "brown", "purple"];

for it_color=1:length(cl_colors)

    figure();
    x = [0 1 1 0];
    y = [0 0 1 1];
    fill(x, y, cl_colors{it_color});
    st_text = [cl_str_clr_names(it_color) num2str(round(cl_colors{it_color}*255))];
    hText = text(.10,.55, st_text);
    hText.Color = 'white';
    hText.FontSize = 30;
    snapnow;

end
```

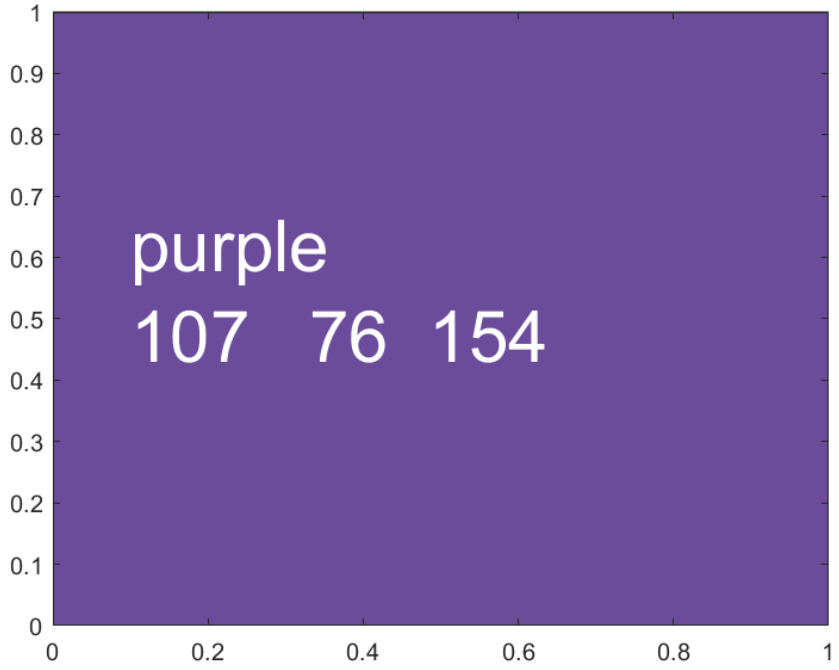

blue
57 106 177
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1


red
204 37 41
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1


black
83 81 84
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1


green
62 150 81
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1



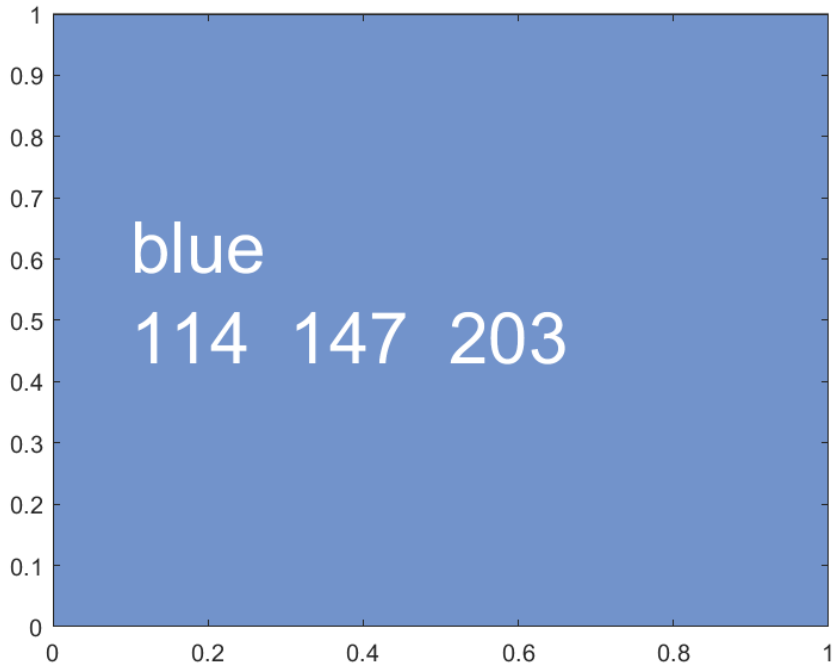


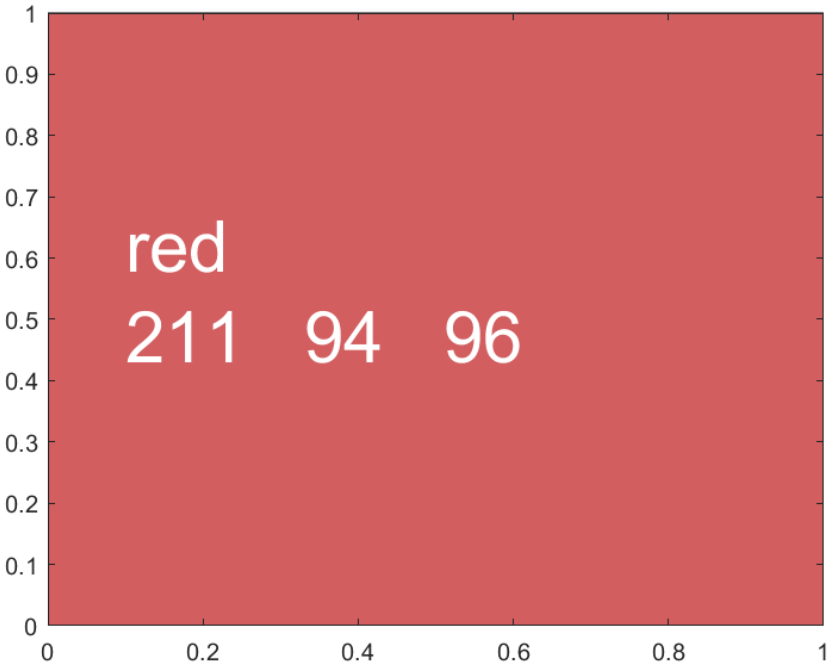
3.1.1.2 Good Colors to Use Lighter

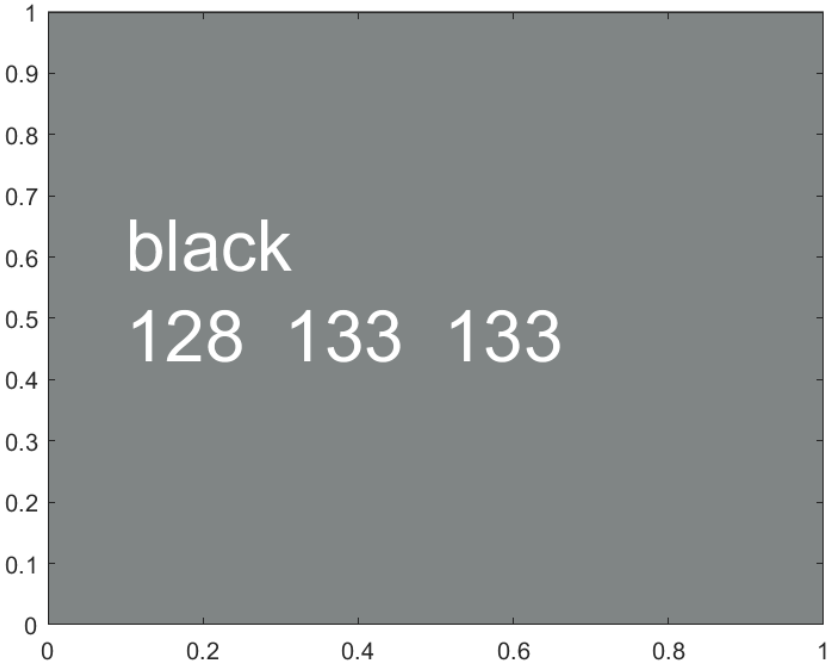
Nice ligher colors to use in matlab.

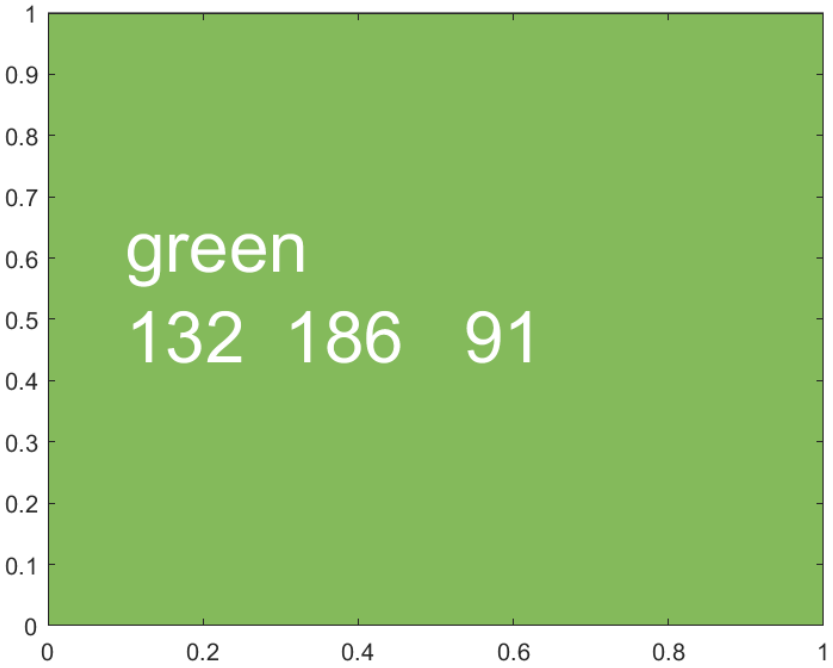
```
close all

blue = [114 147 203]./255;
red = [211 94 96]./255;
black = [128 133 133]./255;
green = [132 186 91]./255;
```

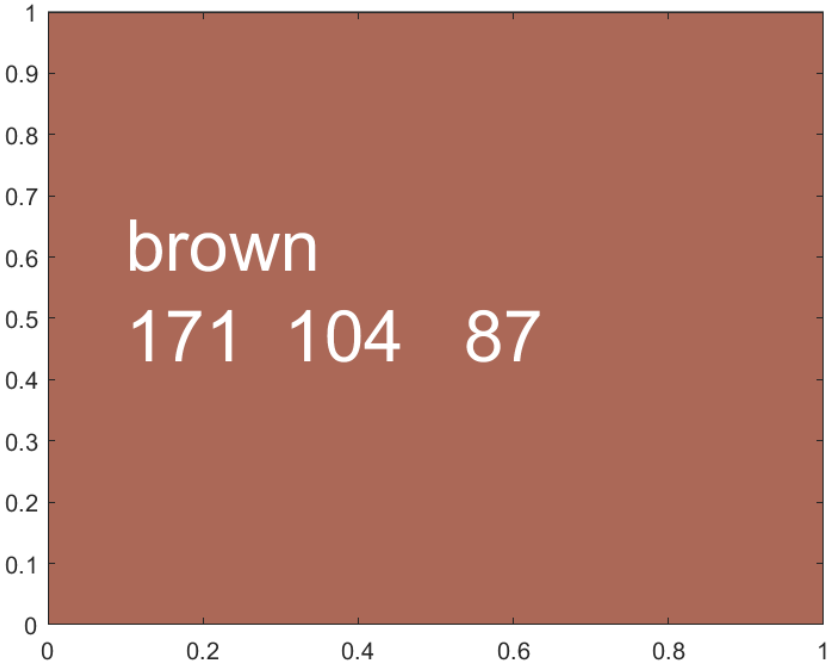
```
brown = [171 104 87]./255;
purple = [144 103 167]./255;
cl_colors = {blue, red, black, ...
             green, brown, purple};
cl_str_clr_names = ["blue", "red", "black", "green", "brown", "purple"];

for it_color=1:length(cl_colors)

    figure();
    x = [0 1 1 0];
    y = [0 0 1 1];
    fill(x, y, cl_colors{it_color});

    st_text = [cl_str_clr_names(it_color) num2str(round(cl_colors{it_color}*255))];
    hText = text(.10,.55, st_text);
    hText.Color = 'white';
    hText.FontSize = 30;
    snapnow;

end
```

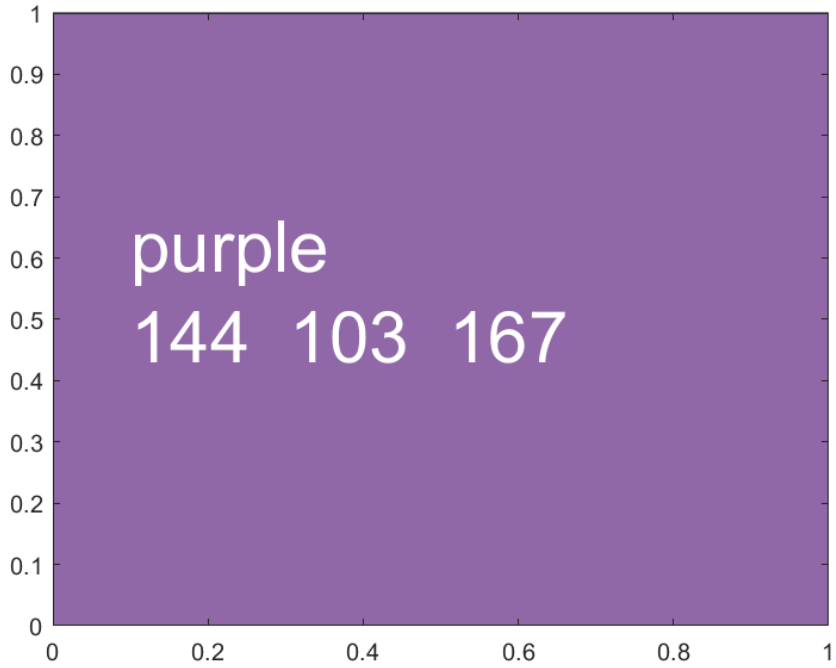



red
211 94 96
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1


black
128 133 133
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1


green
132 186 91
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.2
0.4
0.6
0.8


brown
171 104 87
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.2
0.4
0.6
0.8



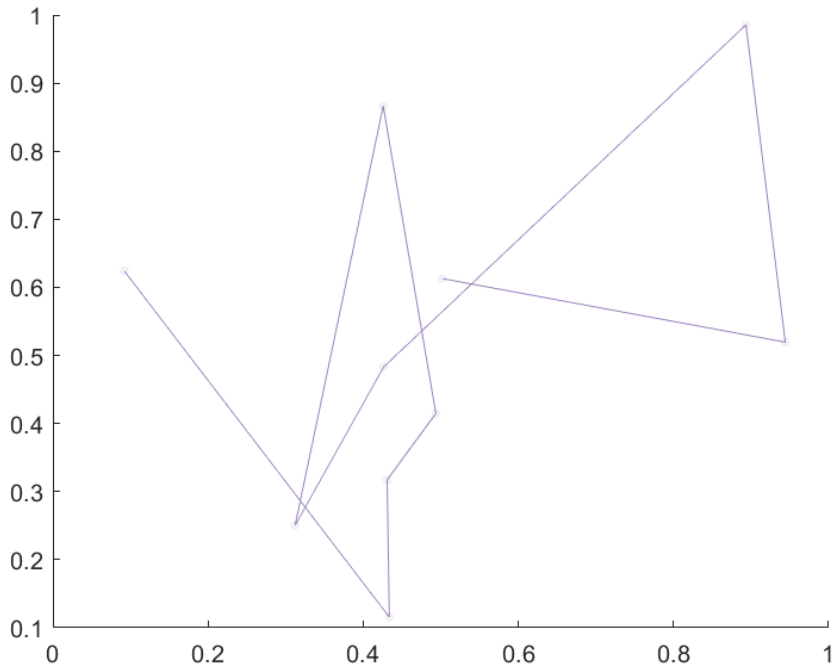
3.1.1.3 Matlab has a graphical tool for picking color

Enter uisetcolor pick color from new window and color values will appear uisetcolor

```
% Color Pickers
% uisetcolor
```

Picked Color use

```
figure();
hold on;

x = rand([10,1]);
y = rand([10,1]);

% Then can use for plot
plot(x,y,'Color',[.61 .51 .74]);

% Can use for Scatter
scatter(x, y, 10, ...
    'MarkerEdgeColor', [.61 .51 .74], 'MarkerFaceAlpha', 0.1, ...
    'MarkerFaceColor', [.61 .51 .74], 'MarkerEdgeAlpha', 0.1);
```

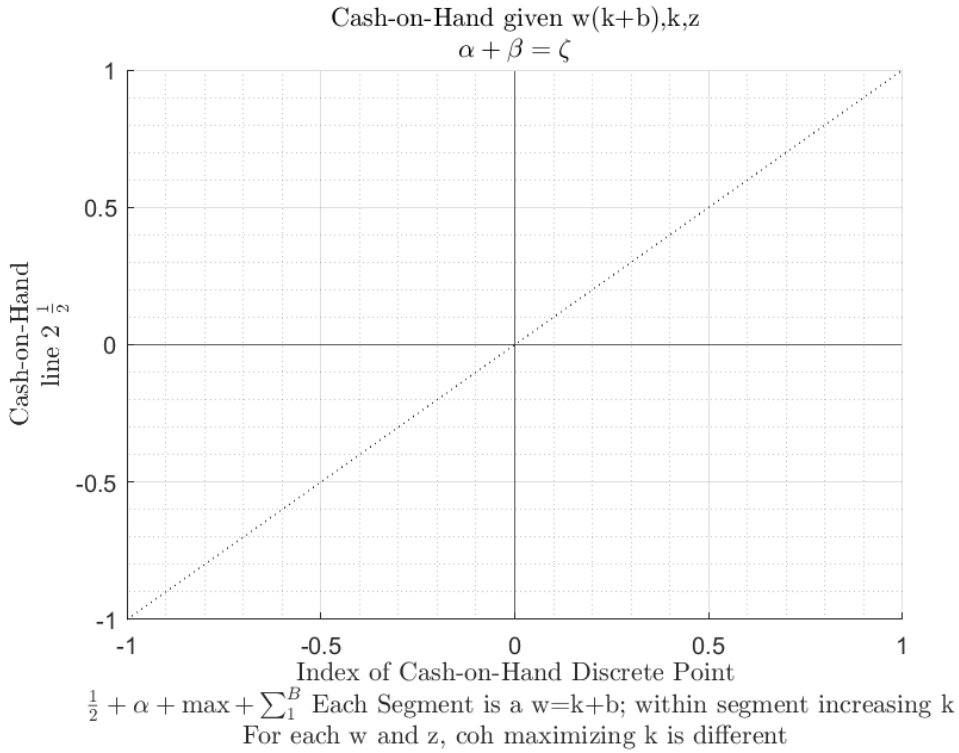


3.1.2 Matlab Graph Titling, Labels and Legends Examples

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

3.1.2.1 Draw A figure Label Title, X and Y Axises with Latex Equations

```
clear all;
close all;
figure();

% draw some lines
xline0 = xline(0);
xline0.HandleVisibility = 'off';
yline0 = yline(0);
yline0.HandleVisibility = 'off';
hline = refline([1 0]);
hline.Color = 'k';
hline.LineStyle = ':';
hline.HandleVisibility = 'off';

% Titling with multiple lines
title({'Cash-on-Hand given w(k+b),k,z' '$\alpha + \beta = \zeta$'},'Interpreter','latex');
ylabel({'Cash-on-Hand' 'line 2 $\frac{1}{2}$'},'Interpreter','latex');
xlabel({'Index of Cash-on-Hand Discrete Point'...
        ' $\frac{1}{2} + \alpha + \max + \sum_1^{B}$ Each Segment is a w=k+b; within segment increas
        'For each w and z, coh maximizing k is different'},'Interpreter','latex');
grid on;
grid minor;
```

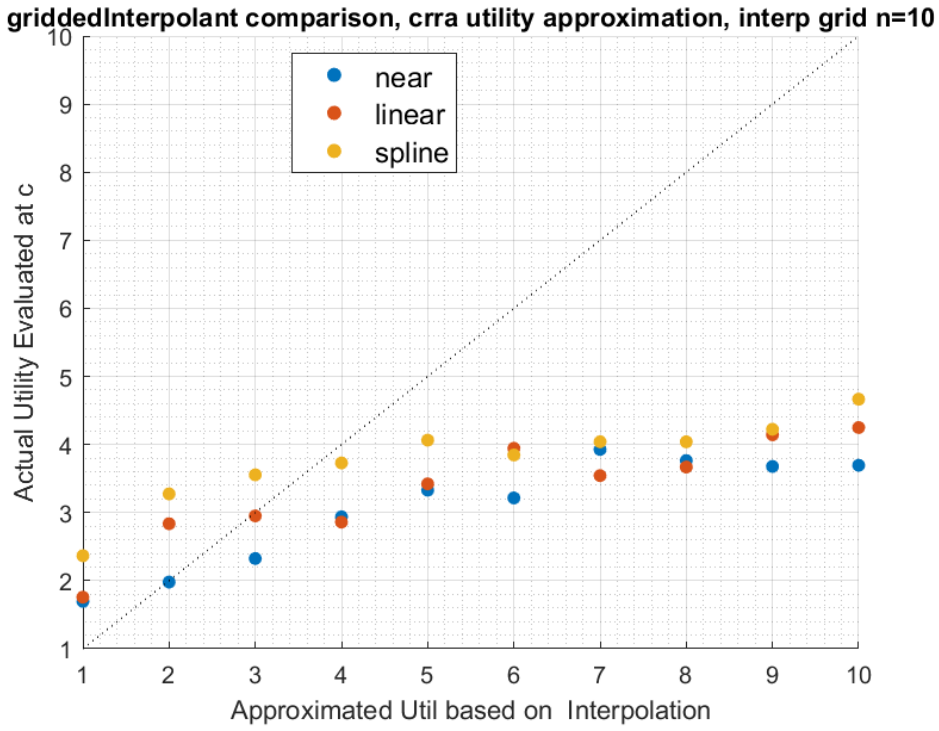


3.1.2.2 Matlab Graph Specify Legends Manually

Specify labels manually, note we can use HandleVisibility to control what part of figure show up in legends.

```
% Generate Random Data
rng(123);
it_x_n = 10;
it_x_groups_n = 3;
mat_y = rand([it_x_n, it_x_groups_n]);
mat_y = mat_y + sqrt(1:it_x_groups_n);
mat_y = mat_y + log(1:it_x_n)';
ar_x = 1:1:it_x_n;

% Start Figure
figure('PaperPosition', [0 0 10 10]);

hold on;

g1 = scatter(ar_x,  mat_y(:,1), 30, 'filled');
g2 = scatter(ar_x,  mat_y(:,2), 30, 'filled');
g3 = scatter(ar_x,  mat_y(:,3), 30, 'filled');

legend([g1, g2, g3], {'near','linear','spline'}, 'Location','best',...
        'NumColumns',1,'FontSize',12,'TextColor','black');

% PLot this line, but this line will not show up in legend
hline = refline([1 0]);
hline.Color = 'k';
hline.LineStyle = ':';
% not to show up in legend
hline.HandleVisibility = 'off';
grid on;
grid minor;
```

```
title(sprintf('griddedInterpolant comparison, crra utility approximation, interp grid n=%d', it_x_n)
ylabel('Actual Utility Evaluated at c')
xlabel('Approximated Util based on  Interpolation')
```

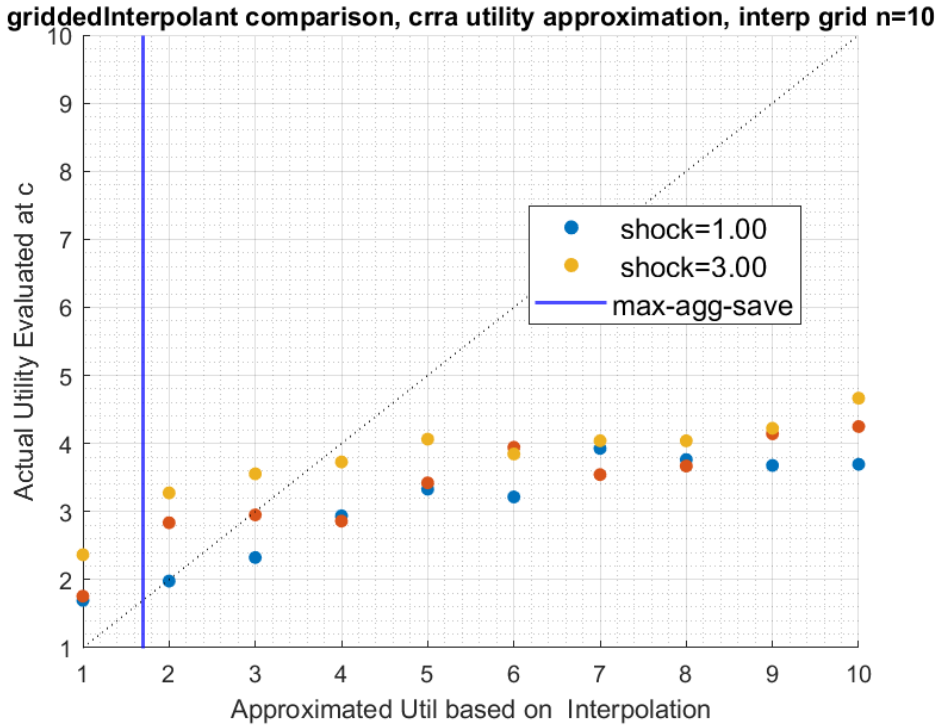



```
snapnow;
```

3.1.2.3 Given Graph, Graph Subset of Lines and Add Extra Line with Legend

Same plot as before, except we plot only 2 of the three lines and add another line with associated legend entry.

```
legendCell = cellstr(num2str(ar_x', 'shock=%3.2f'));
xlinemax = xline(min(mat_y, [], 'all'));
xlinemax.Color = 'b';
xlinemax.LineWidth = 1.5;

legendCell{length(legendCell) + 1} = 'max-agg-save';
legend([g1, g3, xlinemax], legendCell([1,3,length(legendCell)]), 'Location', 'best');
```




```
snapnow;
```

3.1.3 Matlab Graph Matrix with Jet Spectrum Color, Label a Subset Examples

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

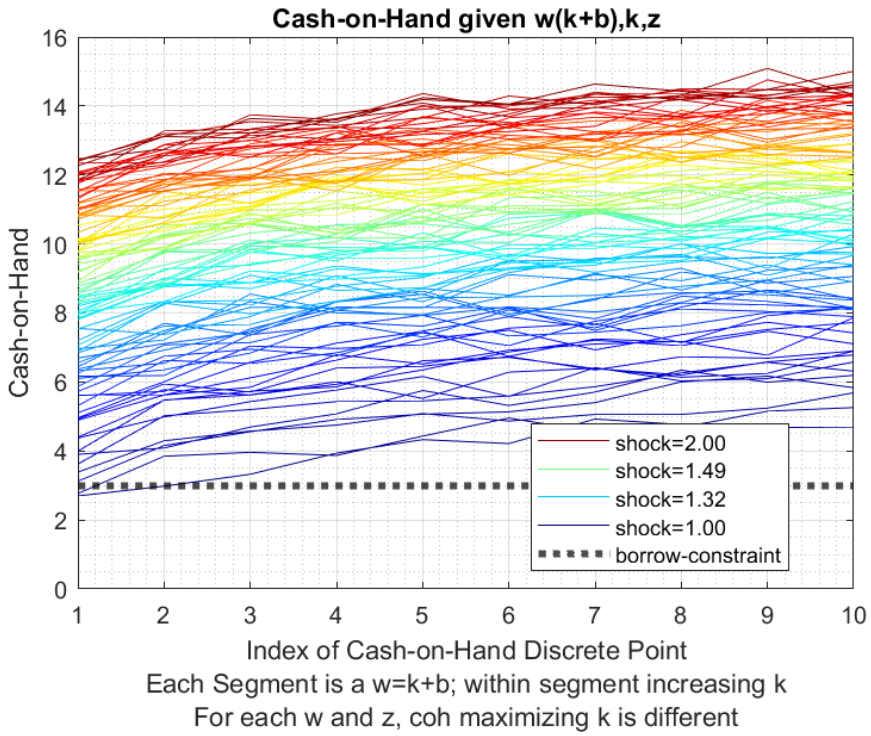
3.1.3.1 Plot a Subset of Data Matrix with Appropriate Legends

Sometimes we solve a model across many states, but we can only plot at a subset of states, or perhaps we plot at all states, but only show legends/labels for a subset.

In the example below, many lines are plotted, however, only a subset of lines are labeled in the legend.

```
clear all;
close all;

% Generate Random Data
rng(123);
it_x_n = 10;
it_y_groups_n = 100;
ar_y = linspace(1,2,it_y_groups_n);
mat_y = rand([it_x_n, it_y_groups_n]);
mat_y = mat_y + sqrt(1:it_y_groups_n);
mat_y = mat_y + log(1:it_x_n)' + ar_y;
ar_x = 1:1:it_x_n;

% Jet color Graph All
figure('PaperPosition', [0 0 7 4]);
chart = plot(mat_y);
clr = jet(numel(chart));
for m = 1:numel(chart)
    set(chart(m),'Color',clr(m,:))
end
```

```
% zero lines
xline(0);
yline(0);

% invalid points separating lines
yline_borrbound = yline(3);
yline_borrbound.HandleVisibility = 'on';
yline_borrbound.LineStyle = ':';
yline_borrbound.Color = 'black';
yline_borrbound.LineWidth = 3;

% Titling
title('Cash-on-Hand given w(k+b),k,z');
ylabel('Cash-on-Hand');
xlabel({'Index of Cash-on-Hand Discrete Point'...
    'Each Segment is a w=k+b; within segment increasing k'...
    'For each w and z, coh maximizing k is different'});

% Xlim controls
xlim([min(ar_x), max(ar_x)]);

% Grid ons
grid on;
grid minor;
% Legends
legend2plot = fliplr([1 round(numel(chart)/3) round((2*numel(chart))/4)  numel(chart)]);
legendCell = cellstr(num2str(ar_y', 'shock=%3.2f'));

legendCell{length(legendCell) + 1} = 'borrow-constraint';
chart(length(chart)+1) = yline_borrbound;
legend(chart([legend2plot length(legendCell)]), ...
       legendCell([legend2plot length(legendCell)]), ...
       'Location', 'best');
```



3.2 Basic Figure Types

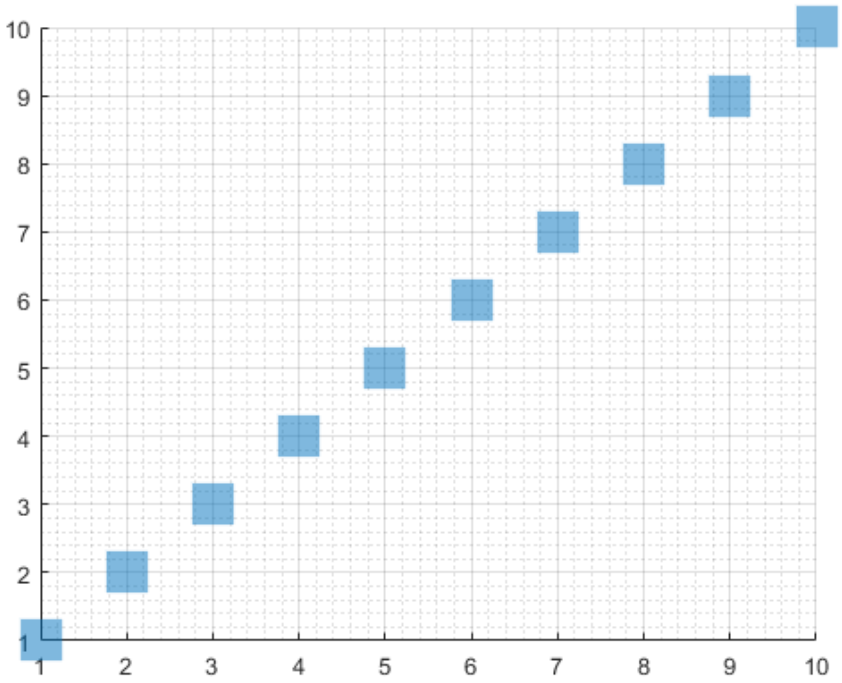
3.2.1 Matlab Graph Scatter Plot Examples

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

3.2.1.1 Scatter Plot Example

The plot below as square scatter points, each one with think border. Can set transparency of border/edge and inside separately.

```
close all;
figure();
size = 100;
s = scatter(1:10,1:10,size);

s.Marker = 's';
% color picked by using: uisetcolor
s.MarkerEdgeColor = [0    0.4471    0.7412];
s.MarkerEdgeAlpha = 0.5;
s.MarkerFaceColor = [.61 .51 .74];
s.MarkerFaceAlpha = 1.0;
s.LineWidth = 10;
grid on;
grid minor;
```



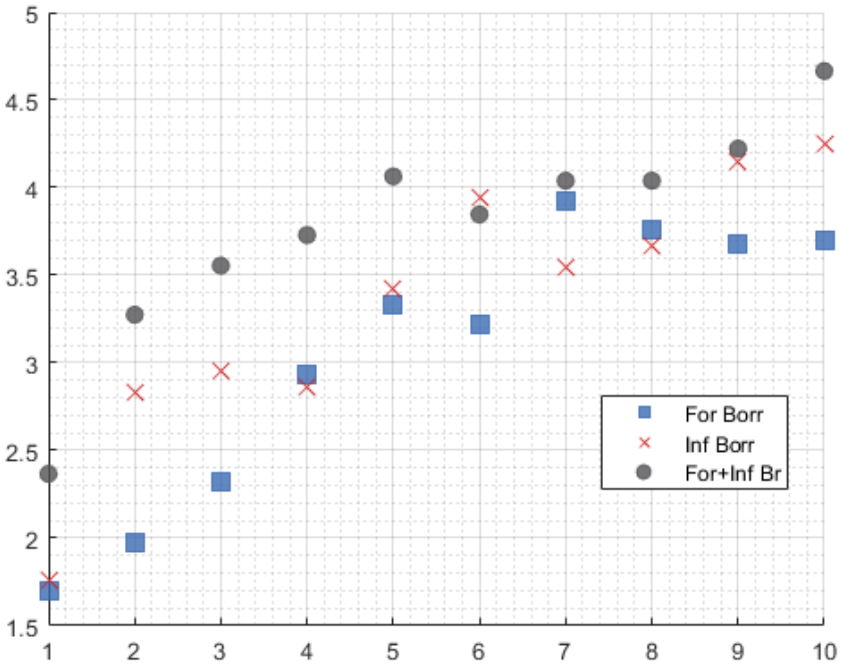

```
% 'o'   Circle
% '+'   Plus sign
% '*'   Asterisk
% '.'   Point
% 'x'   Cross
% 'square' or 's'   Square
% 'diamond' or 'd'  Diamond
% '^'   Upward-pointing triangle
% 'v'   Downward-pointing triangle
```

```
% '>'   Right-pointing triangle
% '<'   Left-pointing triangle
% 'pentagram' or 'p'    Five-pointed star (pentagram)
% 'hexagram' or 'h' Six-pointed star (hexagram)
% 'none'    No markers
```

3.2.1.2 Scatter with Edge and Face Color and Transparency

Here is another way to Set Scatter Edge and Fac Colors and Transparencies.

```
% Generate Data
rng(123);
it_x_n = 10;
it_x_groups_n = 3;
mat_y = rand([it_x_n, it_x_groups_n]);
mat_y = mat_y + sqrt(1:it_x_groups_n);
mat_y = mat_y + log(1:it_x_n)';
ar_x = 1:1:it_x_n;

% Colors
blue = [57 106 177]./255;
red = [204 37 41]./255;
black = [83 81 84]./255;
green = [62 150 81]./255;
brown = [146 36 40]./255;
purple = [107 76 154]./255;
cl_colors = {blue, red, black, ...
             green, brown, purple};

% Scatter Shapes
cl_scatter_shapes = {'s','x','o','d','p','*'};
% Scatter Sizes
cl_scatter_sizes = {100,100,50,50,50,50};
% Legend Keys
cl_legend = {'For Borr', 'Inf Borr', 'For+Inf Br'};

% Plot
figure();
hold on;
for it_m = 1:it_x_groups_n
    scatter(ar_x, mat_y(:,it_m), cl_scatter_sizes{it_m}, ...
        'Marker', cl_scatter_shapes{it_m}, ...
        'MarkerEdgeColor', cl_colors{it_m}, 'MarkerFaceAlpha', 0.8, ...
        'MarkerFaceColor', cl_colors{it_m}, 'MarkerEdgeAlpha', 0.8);
    cl_legend{it_m} = cl_legend{it_m};
end
legend(cl_legend, 'Location', 'best');
grid on;
grid minor;
```



3.2.2 Matlab Line and Scatter Plot with Multiple Lines and Axis Lines

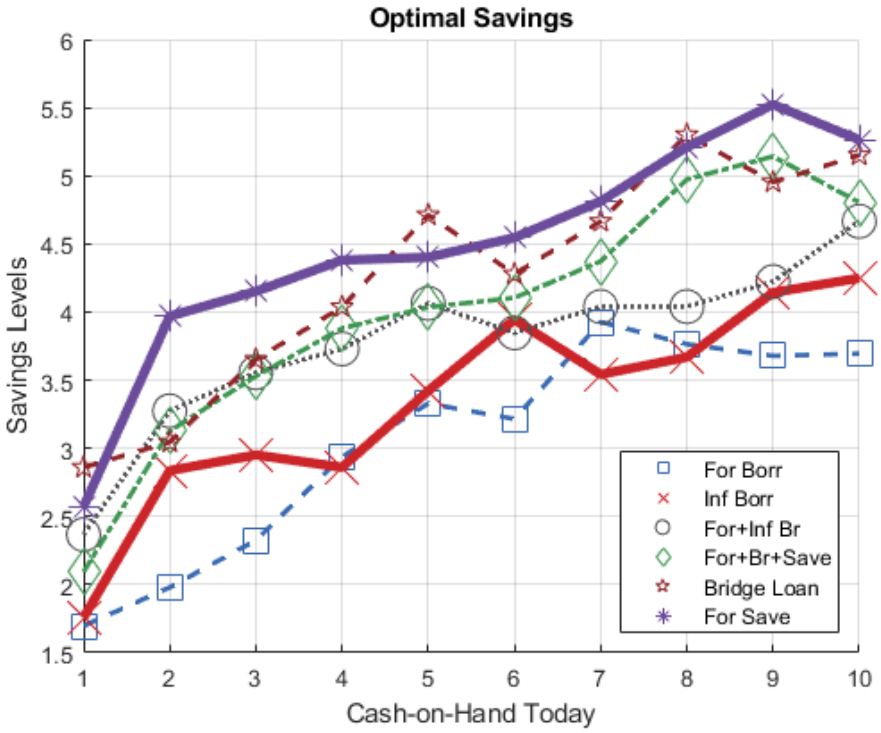
Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

3.2.2.1 Six lines Plot

Colors from optimal colors. Generate A line plot with multiple lines using safe colors, with differening shapes. Figures include lines as well as scatter overlayed jointly.

```
close all
figure();
hold on;

blue = [57 106 177]./255;
red = [204 37 41]./255;
black = [83 81 84]./255;
green = [62 150 81]./255;
brown = [146 36 40]./255;
purple = [107 76 154]./255;
cl_colors = {blue, red, black, ...
             green, brown, purple};
cl_legend = {'For Borr', 'Inf Borr', 'For+Inf Br', 'For+Br+Save', 'Bridge Loan', 'For Save'};
cl_scatter_shapes = {'s','x','o','d','p','*'};
cl_linestyle = {'--','-',':','-.','--','-'};
it_sca_bs = 20;
cl_scatter_csizes = {10*it_sca_bs, 20*it_sca_bs, 10*it_sca_bs, 10*it_sca_bs, 5*it_sca_bs, 8*it_sca_b
it_line_bs = 2;
cl_line_csizes = {1*it_line_bs, 2*it_line_bs, 1*it_line_bs, 1*it_line_bs, 1*it_line_bs, 2*it_line_bs

it_x_groups_n = length(cl_scatter_csizes);
it_x_n = 10;

% Generate Random Data
```

```
rng(123);
mat_y = rand([it_x_n, it_x_groups_n]);
mat_y = mat_y + sqrt(1:it_x_groups_n);
mat_y = mat_y + log(1:it_x_n)';
ar_x = 1:1:it_x_n;

ar_it_graphs_run = 1:6;
it_graph_counter = 0;
ls_chart = [];
for it_fig = ar_it_graphs_run

    % Counter
    it_graph_counter = it_graph_counter + 1;

    % Y Outcome
    ar_y = mat_y(:, it_fig)';

    % Color and Size etc
    it_csize = cl_scatter_csizes{it_fig};
    ar_color = cl_colors{it_fig};
    st_shape = cl_scatter_shapes{it_fig};
    st_lnsty = cl_linestyle{it_fig};
    st_lnwth = cl_line_csizes{it_fig};

    % plot scatter and include in legend
    ls_chart(it_graph_counter) = scatter(ar_x, ar_y, it_csize, ar_color, st_shape);

    % plot line do not include in legend
    line = plot(ar_x, ar_y);
    line.HandleVisibility = 'off';
    line.Color = ar_color;
    line.LineStyle = st_lnsty;
    line.HandleVisibility = 'off';
    line.LineWidth = st_lnwth;

    % Legend to include
    cl_legend{it_graph_counter} = cl_legend{it_fig};
end

% Legend
legend(ls_chart, cl_legend, 'Location', 'southeast');

% labeling
title('Optimal Savings');
ylabel('Savings Levels');
xlabel('Cash-on-Hand Today');
grid on;
```

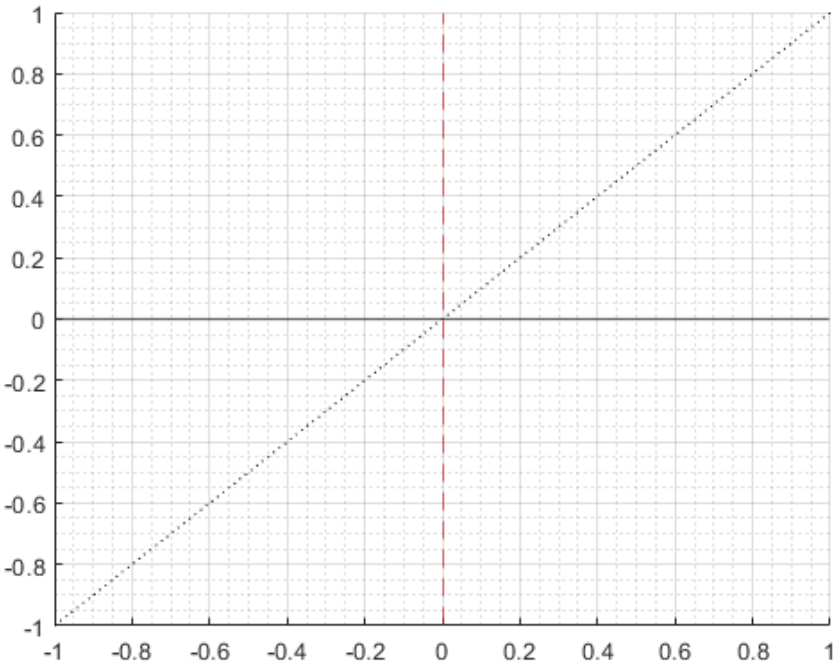



```
snapnow;
```

3.2.2.2 Horizontal and Vertical Lines and 45 Degree

Draw x and y axis, and draw a 45 degree line.

```
figure();

xline0 = xline(0);
xline0.HandleVisibility = 'off';
xline0.Color = red;
xline0.LineStyle = '--';
yline0 = yline(0);
yline0.HandleVisibility = 'off';
yline0.LineWidth = 1;

hline = refline([1 0]);
hline.Color = 'k';
hline.LineStyle = ':';
hline.HandleVisibility = 'off';

snapnow;
grid on;
grid minor;
```



3.2.3 Matlab Graph Scatter and Line Spectrum with Three Variables

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

Generate k + b = w, color for each w, vectors of k and b such that k + b = w for each w

There are two N by M matrix, A anb B.

Values in Matrix A correspond to the x-axis, values in Matrix B correspond to the y-axis.

The rows and columns in matrix A and B have some other meanings. In this case, we will give color to the columns.

The columns is represented by vector C, which is another variable.

1. Each line a different color representing variable 3
2. Legend labeling a subset of colors
3. X and Y could be asset choices, color could be utility, consumption etc.

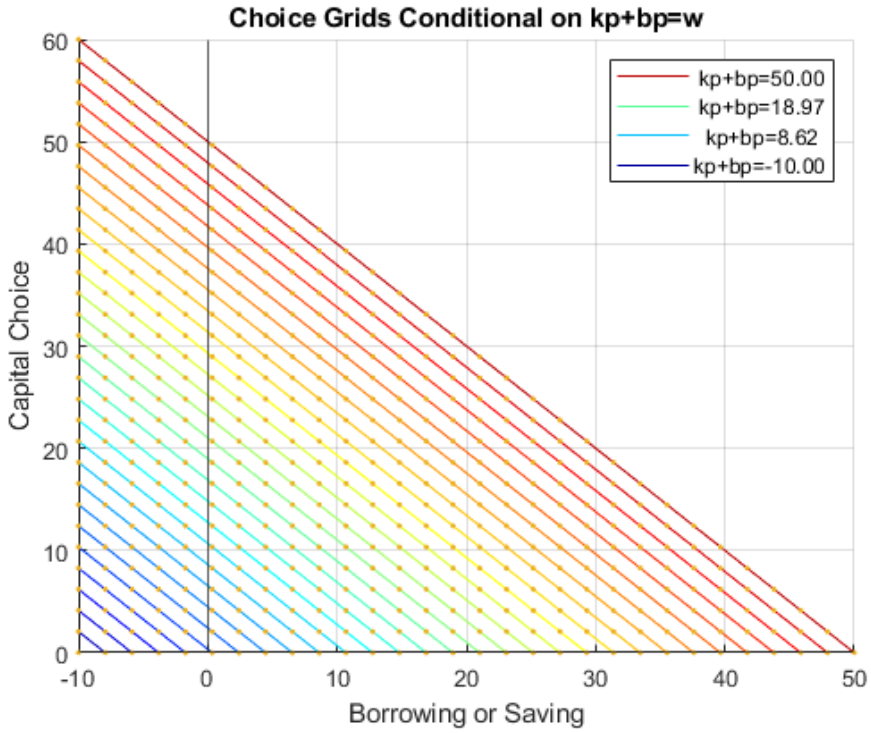
3.2.3.1 Setting Up Data

```
close all
clear all

% Bounds
fl_b_bd = -10;
% Max and Mins
fl_w_max = 50;
fl_w_min = fl_b_bd;
fl_kp_max = fl_w_max - fl_b_bd;
fl_kp_min = 0;

% Grid Point Counts
it_w_i = 30;
it_kb_j = 30;
```

```
% Grids
ar_w = linspace(fl_w_min, fl_w_max, it_w_i);
ar_kp = linspace(fl_kp_min, fl_kp_max, it_kb_j);
mt_bp = ar_w - ar_kp';
mt_kp = ar_w - mt_bp;
mt_bl_constrained = (mt_bp < fl_b_bd);
mt_bp_wth_na = mt_bp;
mt_kp_wth_na = mt_kp;
mt_bp_wth_na(mt_bl_constrained) = nan;
mt_kp_wth_na(mt_bl_constrained) = nan;

% Flatten
ar_bp_mw_wth_na = mt_bp_wth_na(:);
ar_kp_mw_wth_na = mt_kp_wth_na(:);
ar_bp_mw = ar_bp_mw_wth_na(~isnan(ar_bp_mw_wth_na));
ar_kp_mw = ar_kp_mw_wth_na(~isnan(ar_kp_mw_wth_na));
```

3.2.3.2 Graphing

```
figure('PaperPosition', [0 0 7 4]);
hold on;

chart = plot(mt_bp_wth_na, mt_kp_wth_na, 'blue');

clr = jet(numel(chart));
for m = 1:numel(chart)
   set(chart(m),'Color',clr(m,:))
end
if (length(ar_w) <= 50)
    scatter(ar_bp_mw, ar_kp_mw, 5, 'filled');
end
xline(0);
yline(0);

title('Choice Grids Conditional on kp+bp=w')
ylabel('Capital Choice')
xlabel({'Borrowing or Saving'})
legend2plot = fliplr([1 round(numel(chart)/3) round((2*numel(chart))/4)  numel(chart)]);
legendCell = cellstr(num2str(ar_w', 'kp+bp=%3.2f'));
legend(chart(legend2plot), legendCell(legend2plot), 'Location','northeast');

grid on;
```



3.3 Write and Read Plots

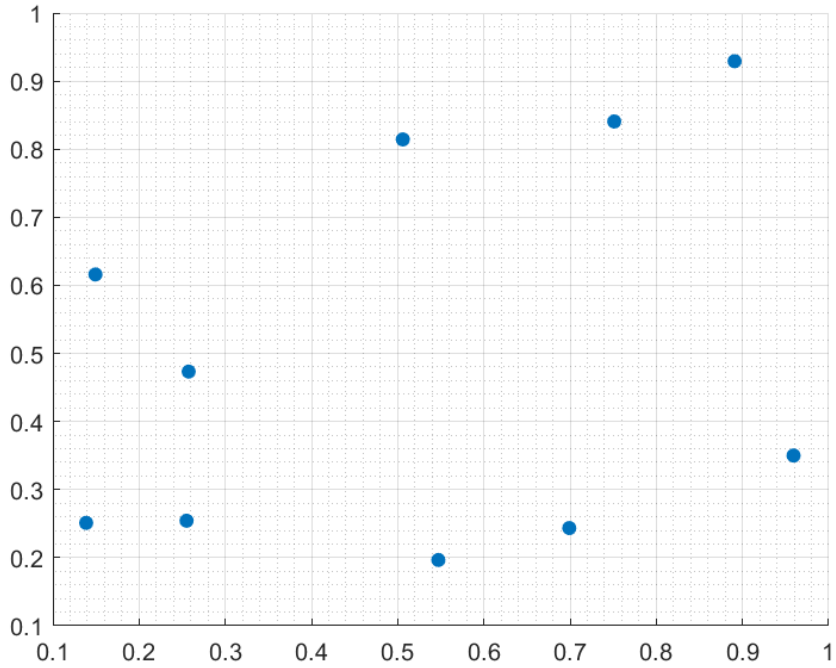
3.3.1 Matlab Graph Generate EPS postscript figures in matlab

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

3.3.1.1 Properly Save EPS with Scatter and Other Graphing Methods: Renderer = Painters

scatter plot saving as eps seems to only work when Renderer is set to Painters

```
fl_fig_wdt = 3;
fl_fig_hgt = 2.65;

figure('PaperPosition', [0 0 fl_fig_wdt fl_fig_hgt], 'Renderer', 'Painters');
x = rand([10,1]);
y = rand([10,1]);
scatter(x, y, 'filled');
grid on;
grid minor;

st_img_path = 'C:/Users/fan/M4Econ/graph/export/_img/';
st_file_name = 'fs_eps_scatter_test';

% eps figure save with tiff preview
print(strcat(st_img_path, st_file_name), '-depsc', '-tiff');
```


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Chapter 4

Tables

4.1 Basic Table Generation

4.1.1 Named Tables with Random Data

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

4.1.1.1 Generate A Table with M Variables

Generate a numeric table with random varlues and a string column

```
% Numeric Matrix
it_num_cols = 4;
it_num_rows = 5;
mt_data = rand([it_num_rows, it_num_cols]);

% Generate Table
tb_test = array2table(mt_data);

% Generate Row and Column Names
cl_col_names = strcat('col_', string((1:it_num_cols)));
cl_row_names = strcat('row_', string((1:it_num_rows)));

tb_test.Properties.VariableNames = matlab.lang.makeValidName(cl_col_names);
tb_test.Properties.RowNames = matlab.lang.makeValidName(cl_row_names);

% Generate two string variable
rng(456);
cl_st_var1 = strcat('data=', string(rand([it_num_rows,1])));
cl_st_var2 = strcat('data=', string(rand([it_num_rows,1])));
tb_test = addvars(tb_test, cl_st_var1, cl_st_var2);

% Display Table
disp(tb_test);
```

	col_1	col_2	col_3	col_4	cl_st_var1	cl_st_var2
row_1	0.43568	0.4688	0.18092	0.14604	"data=0.24876"	"data=0.60411"
row_2	0.38527	0.57	0.11816	0.54272	"data=0.16307"	"data=0.8857"
row_3	0.57571	0.6457	0.24273	0.8571	"data=0.78364"	"data=0.75912"
row_4	0.14609	0.72334	0.0081834	0.20021	"data=0.80852"	"data=0.18111"
row_5	0.68659	0.68067	0.36007	0.13463	"data=0.62563"	"data=0.15017"

4.1.2 Tables Order Columns and Sort

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

4.1.2.1 Given Table, Show Some Columns First

```
% Generate Table
it_num_cols = 4;
it_num_rows = 5;
mt_data = rand([it_num_rows, it_num_cols]);
tb_test = array2table(mt_data);
cl_col_names = strcat('col_', string((1:it_num_cols)));
cl_row_names = strcat('row_', string((1:it_num_rows)));
tb_test.Properties.VariableNames = matlab.lang.makeValidName(cl_col_names);
tb_test.Properties.RowNames = matlab.lang.makeValidName(cl_row_names);

rng(123);
mean = strcat('data=', string(rand([it_num_rows,1])));
sd = strcat('data=', string(rand([it_num_rows,1])));
tb_test_ori = addvars(tb_test, mean, sd);

% Move Variable
tb_test_varmove = movevars(tb_test_ori, {'mean', 'sd'}, 'Before', 'col_1');

% Display
disp(tb_test_ori);
```

```
              col_1       col_2      col_3      col_4          mean               sd
             --------    -------    -------    -------    --------------    --------------

    row_1     0.34318      0.738     0.6344    0.32296    "data=0.69647"    "data=0.42311"
    row_2     0.72905    0.18249    0.84943    0.36179    "data=0.28614"    "data=0.98076"
    row_3     0.43857    0.17545    0.72446    0.22826    "data=0.22685"    "data=0.68483"
    row_4    0.059678    0.53155    0.61102    0.29371    "data=0.55131"    "data=0.48093"
    row_5     0.39804    0.53183    0.72244    0.63098    "data=0.71947"    "data=0.39212"
```

```
disp(tb_test_varmove);
```

```
                mean               sd            col_1       col_2      col_3      col_4
             --------------    --------------    --------    -------    -------    -------

    row_1    "data=0.69647"    "data=0.42311"     0.34318      0.738     0.6344    0.32296
    row_2    "data=0.28614"    "data=0.98076"     0.72905    0.18249    0.84943    0.36179
    row_3    "data=0.22685"    "data=0.68483"     0.43857    0.17545    0.72446    0.22826
    row_4    "data=0.55131"    "data=0.48093"    0.059678    0.53155    0.61102    0.29371
    row_5    "data=0.71947"    "data=0.39212"     0.39804    0.53183    0.72244    0.63098
```

4.1.3 Row and Column Names for Table based on Arrays

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

4.1.3.1 Generate Table with Row and Column Names based on Multiple Numeric Array

Two numeric arrays describe the column names, combine numeric arrays together to form string array which becomes table variable/column names.

```
close all;
```

```
% Generate Table 1
ar_fl_abc1 = [0.4 0.1 0.25 0.3 0.4];
ar_fl_abc2 = [0.4 0.1 0.2 0.3 0.4];
number1 = '123';
number2 = '456';
mt_data_a = [ar_fl_abc1' ar_fl_abc2'];

tb_test_a = array2table(mt_data_a);
cl_col_names_a = {['col' num2str(number1)], ['col' num2str(number2)]};
cl_row_names_a = strcat('rowA=', string((1:size(mt_data_a,1))));

tb_test_a.Properties.VariableNames = cl_col_names_a;
tb_test_a.Properties.RowNames = cl_row_names_a;
disp(tb_test_a);
```

	col123	col456
rowA=1	0.4	0.4
rowA=2	0.1	0.1
rowA=3	0.25	0.2
rowA=4	0.3	0.3
rowA=5	0.4	0.4

4.1.3.2 Include Row Names as a a String Cell Variable

```
% a and b must have the same row names
cl_st_varrownames = tb_test_a.Properties.RowNames;
tb_test_a = addvars(tb_test_a, cl_st_varrownames, 'Before', 1);

disp(tb_test_a);
```

	cl_st_varrownames	col123	col456
rowA=1	{'rowA=1'}	0.4	0.4
rowA=2	{'rowA=2'}	0.1	0.1
rowA=3	{'rowA=3'}	0.25	0.2
rowA=4	{'rowA=4'}	0.3	0.3
rowA=5	{'rowA=5'}	0.4	0.4

4.1.3.3 Include Row Names as a String Variable

```
% a and b must have the same row names
st_varrownames = string(cl_st_varrownames);
tb_test_a = addvars(tb_test_a, st_varrownames, 'Before', 1);

disp(tb_test_a);
```

	st_varrownames	cl_st_varrownames	col123	col456
rowA=1	"rowA=1"	{'rowA=1'}	0.4	0.4
rowA=2	"rowA=2"	{'rowA=2'}	0.1	0.1
rowA=3	"rowA=3"	{'rowA=3'}	0.25	0.2
rowA=4	"rowA=4"	{'rowA=4'}	0.3	0.3
rowA=5	"rowA=5"	{'rowA=5'}	0.4	0.4

4.2 Table Joining

4.2.1 Row and Column Combine Stack Tables and Matrices

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

4.2.1.1 Generate Some Tables and Matrixes for Combination

```
close all;

% Generate Table 1
ar_fl_abc1 = [0.4 0.1 0.25 0.3 0.4];
ar_fl_abc2 = [0.4 0.1 0.2 0.3 0.4];
number1 = '123';
number2 = '456';
mt_data_a = [ar_fl_abc1' ar_fl_abc2'];

tb_test_a = array2table(mt_data_a);
cl_col_names_a = {['col' num2str(number1)], ['col' num2str(number2)]};
cl_row_names_a = strcat('rowA=', string((1:size(mt_data_a,1))));

tb_test_a.Properties.VariableNames = cl_col_names_a;
tb_test_a.Properties.RowNames = cl_row_names_a;
disp(tb_test_a);
```

```
              col123    col456
              ------    ------

    rowA=1      0.4       0.4
    rowA=2      0.1       0.1
    rowA=3     0.25       0.2
    rowA=4      0.3       0.3
    rowA=5      0.4       0.4
```

```
% Generate Table 2
rng(123);
ar_fl_abc3 = rand(size(ar_fl_abc1));
ar_fl_abc4 = rand(size(ar_fl_abc1));
ar_fl_abc5 = rand(size(ar_fl_abc1));

mt_data_b = [ar_fl_abc3' ar_fl_abc4' ar_fl_abc5'];

tb_test_b = array2table(mt_data_b);
cl_col_names_b = {['col' num2str(33)], ['col' num2str(44)], ['col' num2str(55)]};
cl_row_names_b = strcat('rowB=', string((1:size(mt_data_a,1))));

tb_test_b.Properties.VariableNames = cl_col_names_b;
tb_test_b.Properties.RowNames = cl_row_names_b;
disp(tb_test_b);
```

```
               col33      col44      col55
              -------    -------    --------

    rowB=1    0.69647    0.42311     0.34318
    rowB=2    0.28614    0.98076     0.72905
    rowB=3    0.22685    0.68483     0.43857
    rowB=4    0.55131    0.48093    0.059678
    rowB=5    0.71947    0.39212     0.39804
```

4.2.1.2 Combine Tables Together Stack Columns

Tables with the same number of rows, add more columns with named variables

```
% a and b must have the same row names
tb_test_b_withArownames = tb_test_b;
tb_test_b_withArownames.Properties.RowNames = tb_test_a.Properties.RowNames;
tb_ab_col_stacked = [tb_test_a tb_test_b_withArownames];
disp(tb_ab_col_stacked);
```

```
              col123    col456     col33      col44      col55
              ------    ------    -------    -------    --------

    rowA=1      0.4       0.4     0.69647    0.42311     0.34318
    rowA=2      0.1       0.1     0.28614    0.98076     0.72905
    rowA=3     0.25       0.2     0.22685    0.68483     0.43857
    rowA=4      0.3       0.3     0.55131    0.48093    0.059678
    rowA=5      0.4       0.4     0.71947    0.39212     0.39804
```

4.2.1.3 Combine Tables Together Stack Rows

Tables with the same number of columns, dd more rows variables

```
% Select only 2 columns to match table a column count
tb_test_b_subset = tb_test_b(:,1:2);

% Make Column Names consistent
tb_test_b_subset.Properties.VariableNames = cl_col_names_a;

% Reset Row Names, can not have identical row names
tb_test_a.Properties.RowNames = strcat('row=', string((1:size(mt_data_a,1))));
tb_test_b_subset.Properties.RowNames = ...
    strcat('row=', string(((size(mt_data_a,1)+1):(size(mt_data_a,1)+size(tb_test_b_subset,1)))));
% tb_test_b_subset.Properties.RowNames =

% Stack Rows
tb_ab_row_stacked = [tb_test_a; tb_test_b_subset];
disp(tb_ab_row_stacked);
```

```
              col123     col456
              -------    -------

    row=1        0.4        0.4
    row=2        0.1        0.1
    row=3       0.25        0.2
    row=4        0.3        0.3
    row=5        0.4        0.4
    row=6    0.69647    0.42311
    row=7    0.28614    0.98076
    row=8    0.22685    0.68483
    row=9    0.55131    0.48093
    row=10   0.71947    0.39212
```

4.2.1.4 ND Dimensional Parameter Arrays, Simulate Model and Stack Output Tables

Now we will first column combine matrixes, model parameters and model outcomes, and then row combine matrixes from different simulations.

A model takes a N parameters, solve the model over M sets of parameters. Each time when the model is solved, a P by Q table of results is generated. Each column is a different statistics (mean, std, etc.), and each row is a different outcome variable (consumption, asset choices, etc.). Stack these P by Q Tables

together, and add in information about the N parameters, each of the tables been stacked initially had the same column and row names.

The resulting table should have P times M rows, for M sets of model simulations each with P rows of results. And there should be N + Q columns, storing the N parameters as well as the Q columns of different outcomes.

```
rng(123);
% Generate A P by Q matrix of random parameter Values
it_param_groups_m = 5;
it_params_n = 2;
it_outcomes_p = 3;
it_stats_q = 3;

% Parameter Matrix and Names
ar_param_names = strcat('param_', string(1:it_params_n));
mt_param_m_by_n = round(rand([it_param_groups_m, it_params_n])*5, 2);

% Loop over the parameters
for it_cur_param_group=1:1:it_param_groups_m

    % Current Parameters
    ar_param = mt_param_m_by_n(it_cur_param_group,:);

    % Some Model is simulated
    mt_model_simu = normrnd(mean(ar_param), std(ar_param), [it_outcomes_p, it_stats_q]);

    % Model Results are Saved As Table With Column and Row Information
    tb_model_simu = array2table(mt_model_simu);
    cl_col_names = strcat('stats_', string((1:size(mt_model_simu,2))));
    cl_row_names = strcat('outvar_', string((1:size(mt_model_simu,1))));
    tb_model_simu.Properties.VariableNames = cl_col_names;
    tb_model_simu.Properties.RowNames = cl_row_names;

    % Convert Row Variable Names to a Column String
    outvar = string(tb_model_simu.Properties.RowNames);
    tb_model_simu = addvars(tb_model_simu, outvar, 'Before', 1);

    % Parameter Information Table that Shares Row Names as Simu Results
    mt_param_info = zeros([it_outcomes_p,it_params_n]) + ar_param;
    tb_param_info = array2table(mt_param_info);
    tb_param_info.Properties.VariableNames = ar_param_names;
    tb_param_info.Properties.RowNames = cl_row_names;

    % Combine Parameter Information and Simulation Contents
    tb_model_simu_w_info = [tb_param_info tb_model_simu];
    % Update Row Names based on total row available
    ar_rows_allsimu = (1:it_stats_q)' + (it_cur_param_group-1)*it_stats_q;
    tb_model_simu_w_info.Properties.RowNames = strcat('row=', string(ar_rows_allsimu));

    % Show One Example Table before Stacking
    if (it_cur_param_group == round(it_param_groups_m/2))
        disp(tb_model_simu);
        disp(tb_param_info);
        disp(tb_model_simu_w_info);
    end

    % Stack all results
    if(it_cur_param_group == 1)
```

```
        tb_model_allsimu_w_info = tb_model_simu_w_info;
    else
        tb_model_allsimu_w_info = [tb_model_allsimu_w_info; tb_model_simu_w_info];
    end

end
```

```
                outvar      stats_1     stats_2    stats_3
              ----------    --------    -------    -------

    outvar_1    "outvar_1"    0.056853    2.1703     2.1098
    outvar_2    "outvar_2"      3.1545    2.0634     0.7798
    outvar_3    "outvar_3"    -0.49033    2.2566     1.7896
              param_1    param_2
              -------    -------

    outvar_1     1.13       3.42
    outvar_2     1.13       3.42
    outvar_3     1.13       3.42
           param_1    param_2      outvar      stats_1     stats_2    stats_3
           -------    -------    ----------    --------    -------    -------

    row=7     1.13       3.42      "outvar_1"    0.056853    2.1703     2.1098
    row=8     1.13       3.42      "outvar_2"      3.1545    2.0634     0.7798
    row=9     1.13       3.42      "outvar_3"    -0.49033    2.2566     1.7896
```

Show all Simulation Joint Table Outputs:

```
disp(tb_model_allsimu_w_info);
```

```
              param_1    param_2      outvar      stats_1     stats_2    stats_3
              -------    -------    ----------    --------    -------    -------

    row=1      3.48       2.12      "outvar_1"      2.2665    1.1885      1.924
    row=2      3.48       2.12      "outvar_2"      3.3427    2.4647     2.3548
    row=3      3.48       2.12      "outvar_3"      2.6714    3.6132      2.918
    row=4      1.43        4.9      "outvar_1"      3.3859    5.3759     1.5816
    row=5      1.43        4.9      "outvar_2"      3.9499    3.8698     2.2693
    row=6      1.43        4.9      "outvar_3"      5.7745    4.6871     1.7334
    row=7      1.13       3.42      "outvar_1"    0.056853    2.1703     2.1098
    row=8      1.13       3.42      "outvar_2"      3.1545    2.0634     0.7798
    row=9      1.13       3.42      "outvar_3"    -0.49033    2.2566     1.7896
    row=10     2.76        2.4      "outvar_1"      2.9611    2.6847     2.4986
    row=11     2.76        2.4      "outvar_2"      2.9333    2.3457     3.0629
    row=12     2.76        2.4      "outvar_3"      2.5814    2.4372     2.4806
    row=13      3.6       1.96      "outvar_1"      2.7199    3.3129     3.0577
    row=14      3.6       1.96      "outvar_2"      3.9804    1.4529     2.9285
    row=15      3.6       1.96      "outvar_3"      2.8445    4.4117     2.6576
```

Chapter 5

Panel

5.1 Time Series

5.1.1 Simulate AR(1) Autoregressive Processes

Go back to fan's CodeDynaAsset Package, Matlab Code Examples Repository (bookdown site), or Math for Econ with Matlab Repository (bookdown site).

5.1.1.1 Mean and Standard Deviation for AR(1) Autoregressive Process

A first-order autoregressive process can be written as:

- AR1: $X_t = \text{constant} + \text{persistence} \cdot x_{t-1} + \epsilon$
- AR1: $X_t = C + \rho \cdot x_{t-1} + \epsilon$

Assume that ϵ is mean zero

Note that, we know the mean of X:

- $\mu_X = C + \rho \cdot \mu_X + 0$
- $\mu_x = \frac{C}{1-\rho}$

Note that, we also know the standard deviation of X:

- $\text{var}(X) = \rho^2 \cdot \text{var}(X) + \text{var}(\epsilon)$
- $\sigma_x = \sqrt{\frac{\sigma_\epsilon^2}{1-\rho^2}}$

We will let the initial point of the time series follow the stationary distribution of the AR(1) process, then we simulate the time series over 100 periods, in the example below, we use a highly persistent shock process with $\rho = 0.98$, $\sigma_\epsilon = 0.02$, $C = 0.02$. Note that for this process:

- $\mu_x^{\rho=0.98,\sigma_\epsilon=0.02,C=0.02} = \frac{0.02}{1-0.98} = 1$
- $\sigma_x^{\rho=0.98,\sigma_\epsilon=0.02,C=0.02} = \sqrt{\frac{0.02^2}{1-0.98^2}} \approx 0.10$

5.1.1.2 Simulated one First-Order Autoregressive Time-Series

In the Example below, we simulate an individual for 1000 periods, given $\rho = 0.98$, $\sigma_\epsilon = 0.02$, $C = 0.02$. Given that the process is highly persistent, the individual stays rich or poor for dozens of periods at a time. If each period is a year, look at the results below, and suppose the simulated time series is income, what is the process saying about this person's income rise and fall. Note that we have the same person through all 1000 periods, but if you only look at 50 periods (years), you might this this person during

one span is really successful, anothe segment of 50 years, doing really bad, but actually there is nothing changing in the person's type, all that is changing is the person's luck.

First Set Parameters:

```
% Number of Time Periods
it_T = 1000;
% Mean and SD of the Shock Process
fl_constant = 0.02;
fl_normal_sd = 0.02;
% Persistence
fl_persistence = 0.98;
% Bounds on Shocks
fl_shk_bnds = 3;
% Initialize with exo fed point or not, if false initialize at Random Point
% from the stationary distribution
bl_init = true;
fl_init = fl_constant/(1 - fl_persistence);
```

Second, generate a vector of normal shocks:

```
% Generate a normal shock vector (the first draw will be ignored)
it_draws = it_T;
rng(789);
ar_fl_shocks = normrnd(0, fl_normal_sd, 1, it_draws);
disp(ar_fl_shocks(1:20));

  Columns 1 through 13

   -0.0060   -0.0047    0.0168    0.0118    0.0380    0.0062   -0.0616   -0.0485   -0.0192    0.0023

  Columns 14 through 20

   -0.0089    0.0160    0.0099   -0.0200   -0.0206   -0.0090   -0.0069
```

Third, replace any values exceeding bounds:

```
% out of bounds indicators
fl_shk_bds_lower = 0 - fl_normal_sd*fl_shk_bnds;
fl_shk_bds_upper = 0 + fl_normal_sd*fl_shk_bnds;
ar_bl_outofbounds = (ar_fl_shocks <= fl_shk_bds_lower | ar_fl_shocks >= fl_shk_bds_upper);
% count out of bounds
disp(strcat('lower:', num2str(fl_shk_bds_lower), ', upper:', num2str(fl_shk_bds_upper)));

lower:-0.06, upper:0.06

disp(sum(ar_bl_outofbounds));

     4

ar_fl_shocks(ar_fl_shocks <= fl_shk_bds_lower) = fl_shk_bds_lower;
ar_fl_shocks(ar_fl_shocks >= fl_shk_bds_upper) = fl_shk_bds_upper;
```

Fourth, generate the AR(1) time series:

```
% Initialize Output Array
ar_fl_time_series = zeros(size(ar_fl_shocks));
% Loop over time
for it_t=1:1:length(ar_fl_shocks)
    if (it_t == 1)
        % initialize using the ean of the process
        ar_fl_time_series(1) = fl_constant/(1 - fl_persistence);
```

```
        if (bl_init)
            ar_fl_time_series(1) = fl_init;
        end
    else
        fl_ts_t = fl_constant + ar_fl_time_series(it_t-1)*fl_persistence + ar_fl_shocks(it_t);
        ar_fl_time_series(it_t) = fl_ts_t;
    end
end
```

Fifth, show the mean and sd of the process (these are very close to the analytical results):

```
disp(mean(ar_fl_time_series));
```

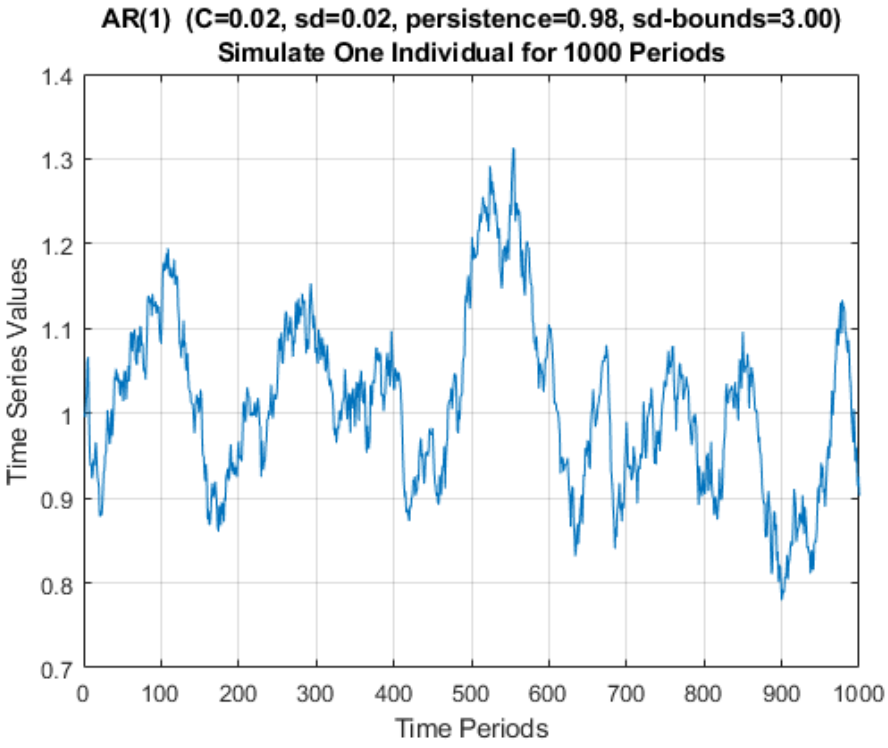
```
    1.0104
```

```
disp(std(ar_fl_time_series));
```

```
    0.1000
```

Sixth, plot the results:

```
figure();
% x-axis
ar_it_time = 1:1:length(ar_fl_shocks);
% plot
plot(ar_it_time, ar_fl_time_series);
% Generate Title
ar_fl_params_values = [fl_constant, fl_normal_sd, fl_persistence, fl_shk_bnds];
ar_st_parms_names = ["C", "sd", "persistence", "sd-bounds"];
st_rounding = '.2f';
st_title_main = "AR(1) ";
ar_st_params = strcat(ar_st_parms_names, compose(strcat("=%", st_rounding), ar_fl_params_values));
st_param_pasted = strjoin(ar_st_params, ', ');
st_title_wth_params = strcat(st_title_main, ' (', st_param_pasted, ')');
title({st_title_wth_params, 'Simulate One Individual for 1000 Periods'});
% X and Y labels
ylabel({'Time Series Values'});
xlabel('Time Periods');
grid on;
```


AR(1) (C=0.02, sd=0.02, persistence=0.98, sd-bounds=3.00)
Simulate One Individual for 1000 Periods
Time Series Values
Time Periods
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0
100
200
300
400
500
600
700
800
900
1000

Appendix A

Index and Code Links

A.1 Data Structures links

A.1.1 Section 1.1 Matrices and Arrays links

1. Array Reshape, Repeat and Expand: **mlx** | **m** | **pdf** | **html**
   - Reshape and flatten arrays.
   - **m**: *reshape()*
2. Array Index Slicing and Subsetting to Replace and Expand: **mlx** | **m** | **pdf** | **html**
   - Index based column and row expansions.
   - Anonymous function to slice array subsets.
   - **m**: *sub2ind() + @(it_subset_n, it_ar_n) unique(round(((0:1:(it_subset_n-1))/(it_subset_n-1)) times (it_ar_n-1)+1))*
3. 3D, 4D, N4 Arrays Reshape and Summarize: **mlx** | **m** | **pdf** | **html**
   - Slice 2D matrixes out of ND matrixes. The 2D matrix is contiguous, but can be intermediate dimensions.
   - Summarize a nd dimensional matrix along one or two dimensions group by various other dimensions.
   - **m**: *permute(mn, [3,1,2,4]) + squeeze(num2cell(mn, [1,2])) + celldisp() + ndgrid()*
4. Array Broadcasting Examples: **mlx** | **m** | **pdf** | **html**
   - broadcast means: array + array' + matrix = matrix.
5. Grid States, Choices and Optimal Choices Example: **mlx** | **m** | **pdf** | **html**
   - States, choices, and find max.
6. Accumarray Examples: **mlx** | **m** | **pdf** | **html**
   - Accumarray to sum up probabilities/values for discrete elements of arrays.
   - **m**: *unique() + reshape() + accumarray()*
7. Array Random Draws and Permutation: **mlx** | **m** | **pdf** | **html**
   - Draw randomly from array, permutate arrays.
   - **m**: *ndgrid() + cell2mat(cellfun(@(m) m(:), cl_mt_all, 'uni', 0))*
8. Imaginary Elements of Array: **mlx** | **m** | **pdf** | **html**
   - Find imaginary elements of array.
   - **m**: *imag()*

A.1.2 Section 1.2 Cells links

1. List Comprehension with Cells: **mlx** | **m** | **pdf** | **html**
   - Cell2mat, cellfun, anonymous function list comprehension over cells.
   - Find min and max of all arrays in cells.
   - Find length of all arrays in cells; find index of elements of one array in another cell array.
   - **m**: *cell2mat() + cellfun() + strcmp() + find() + cell2mat(cellfun(@(m) find(strcmp(ls_st_param_key, m)), cl_st_param_keys, 'UniformOutput', false))*
2. Permutate Cells: **mlx** | **m** | **pdf** | **html**
   - Generate all possible combinations of various arrays contained in cell array.

- **m**: *ndgrid() + cell2mat() + array2table() + cell2mat(cellfun(@(m) m(:), cl_mt_all, 'uni', 0))*

3. Combine Cells: **mlx** | **m** | **pdf** | **html**
   - Combine string cell arrays and string.
   - **m**: *[{st_param}, ls_st_param_key, cl_st_param_keys]*
4. Nested Cells: **mlx** | **m** | **pdf** | **html**
   - Cell of cells with inner cell having multiple types.
   - **m**: *linspace() + cell([4,1]) + clns_parm_tstar{1} = {'fl_crra', 'CRRA', linspace(1, 2, it_simu_vec_len)} + disp(clns_parm_tstar(1)) + disp(clns_parm_tstar{1}{1})*

A.1.3 Section 1.3 Characters and Strings links

1. String Basics: **mlx** | **m** | **pdf** | **html**
   - Compose string and rounded numeric array.
   - Cut string suffix and append new suffix.
   - **m**: *compose() + strjoin() + str_sub = split(string, ".") + strcat(str_sub{1}, '_m.m')*
2. String Arrays Operations: **mlx** | **m** | **pdf** | **html**
   - String arrays and cell strings.
   - Duplicate strings, concatenate string, and paste strings jointly with separator.
   - Find string element positions, replace substrings.
   - **m**: *repmat() + num2str() + strcat() + strjoin() + fprintf() + strcmp() + strrep() + cel2mat(cellfun(@(m) find(strcmp()))))*
3. String and Numeric Array Concatenations: **mlx** | **m** | **pdf** | **html**
   - Generate rounded string array matrix with leading zero, leading space, decimal round from numeric matrix.
   - Create a title string by joining rounded parameter and parameter names.
   - Concatenate multiple numeric arrays together with strings and format.
   - **m**: *compose() + cellstr() + strcat() + strjoin() + %.2f*

A.1.4 Section 1.4 Map Containers links

1. Container Map Basics: **mlx** | **m** | **pdf** | **html**
   - Numeric container map, dynamically filled container map.
   - **m**: *isKey() + strjoin() + containers.Map('KeyType', 'char', 'ValueType', 'any')*
2. Display Container Map Keys, Values and Subseting: **mlx** | **m** | **pdf** | **html**
   - Loop over map, display keys and values.
   - Select Container map subset by keys.
   - **m**: *strjoin() + keys(map) + values(map) + containers.Map(keys, values)*
3. Container Map Varied Value Type: **mlx** | **m** | **pdf** | **html**
   - Numeric scalar, string, matrix as values for map container.
   - Get values for multiple keys in map.
   - **m**: *map.keys() + map.values() + values(param_map, {'share_unbanked_j', 'equi_r_j'})*
4. Cell Override: **mlx** | **m** | **pdf** | **html**
   - Override default map with externally fed map, update existing and add new keys.
   - **m**: *param_map_updated = [param_map_old; param_map_updates_new]*

A.2 Functions links

A.2.1 Section 2.1 varargin Default Parameters links

1. Use varargin as a Function Parameter: **mlx** | **m** | **pdf** | **html**
   - Default parameters allow for maintaining code testability.
   - Use varargin for functions with limited parameters.
   - **m**: *varargin + cell2mat() + function [out_put] = func_name(varargin)*
2. Use varargin as a Function Parameter: **mlx** | **m** | **pdf** | **html**
   - The varargin structure could lead to excessive code lines. Container Map works well with large parameter structure.

- Core model functions with potentially many parameters, possibly override default generation to save time.
- **m**: *varargin + function [out_put] = func_name(varargin) + cm_defaults = {cm_a, cm_b} + [cm_defaults{1:optional_params_len}] = varargin{:} + cm_c = [cm_a;cm_b]*

A.3 Graphs links

A.3.1 Section 3.1 Figure Components links

1. Image Pick Safe Colors: **mlx** | **m** | **pdf** | **html**
   - Display safe colors.
   - **m**: *blue = [57 106 177]./255 + fill(x, y, cl_colors{it_color})*
2. Figure Titling and Legend: **mlx** | **m** | **pdf** | **html**
   - Multi-line titles, add legend lines.
   - Add to legend, select legend to show.
   - **m**: *title({'Cash-on-Hand' 'α + β = ζ'},'Interpreter','latex') + legend([g1, g2, g3], {'near','linear','spline'}, 'Location', 'best', 'NumColumns', 1, 'FontSize', 12, 'TextColor', 'black');*
3. Graph Many Lines Legend for Subset: **mlx** | **m** | **pdf** | **html**
   - State-space plots with color spectrum: can not show all states in legend, show subset, add additional line to plot and legend.
   - **m**: *jet() + numel() + fliplr() + jet(numel(chart)), set(chart(m), 'Color', clr(m,:))*

A.3.2 Section 3.2 Basic Figure Types links

1. Scatter Plot Examples: **mlx** | **m** | **pdf** | **html**
   - Scatter multiple lines different colors, shapes and sizes.
   - **m**: *scatter(x, y, size) + Marker + MarkerEdgeColor + MarkerEdgeAlpha + MarkerFaceColor + MarkerFaceAlpha*
2. Scatter Plot Examples: **mlx** | **m** | **pdf** | **html**
   - Scatter and lines multiple lines different colors, shapes and sizes.
   - X axis, Y axis, and 45 degree line.
   - **m**: *xline(0) + yline(0) + refline([1 0]) + plot(x,y) + HandleVisibility + Color + LineStyle + LineWidth*
3. Three variables Scatter and Lines with Color Spectrum: **mlx** | **m** | **pdf** | **html**
   - Two dimensional matrix for x and y, a third variable with color spectrum set via loop.
   - **m**: *plot(2d, 2d) + jet + set(chart(m), 'Color', clr)*

A.3.3 Section 3.3 Write and Read Plots links

1. Graph Generate EPS Postscript Figures: **mlx** | **m** | **pdf** | **html**
   - EPS vector graphics, avoid bitmap (jpg, png), use vector graphics.
   - **m**: *figure('Renderer', 'Painters')*

A.4 Tables links

A.4.1 Section 4.1 Basic Table Generation links

1. Named Tables with Random Data: **mlx** | **m** | **pdf** | **html**
   - Convert a random matrix to a table with column and row names defined with arrays.
   - **m**: *array2table() + strcat() + addvars() + matlab.lang.makeValidName()*
2. Order and Sort Columns: **mlx** | **m** | **pdf** | **html**
   - Convert a matrix to table with mean and sd columns. Rearrange columns.
   - **m**: *array2table() + rng() + movevars() + matlab.lang.makeValidName() + tb.Properties.VariableNames + tb.Properties.RowNames*
3. Array Based Row and Column Names: **mlx** | **m** | **pdf** | **html**
   - Generate a column and row named table. Convert row names to a column as strings.

- **m**: *array2table() + string() + strcat('rowA=', string((1:size(mt, 1)))) + tb_test_a.Properties.VariableNames + tb_test_a.Properties.RowNames + addvars(tb, rownames, 'Before', 1)*

A.4.2 Section 4.2 Table Joining links

1. Stack Matlab Tables: **mlx** | **m** | **pdf** | **html**
   - Append columns to existing table. Stack tables vertically and horizontally.
   - Simulate a model, column combine simulation parameters with multi-row simulation results. Then row stack results from multiple simulations together.
   - **m**: *array2table() + [tb_a tb_b] + [tb_a; tb_b] + tb.Properties.VariableNames + tb.Properties.RowNames*

A.5 Panel links

A.5.1 Section 5.1 Time Series links

1. Autoregressive Process AR(1): **mlx** | **m** | **pdf** | **html**
   - The Mean and standard deviation of an AR(1) process.
   - Simulate and graph an AR(1) persistent process.
   - **m**: *normrnd() + for it_t=1:1:length(ar_shk) + plot(ar_t, ar_y)*

Bibliography

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

Wang, F. (2020). *MEconTools: Tools for Analyzing Matlab Data Structures and Dynamic Programming*. M package version 0.0.0.9000.

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