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

back to Fan's Intro Math for Econon, Matlab Examples, or Dynamic Asset Repositories

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
```

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
```

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
```

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
```

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. The results worked out here are embedded in the fx_summ_nd_array function of the MEconTools Package.

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_s
% disp(['xxx ' st_title ' xxxxxxxxxxxxxxxxxxxxxxxxxxx']);
disp(tb_rowcols_tab);
```

```
    group    kid    marriage    mean_z_3    mean_z_1    mean_z1    mean_z3    sd_z_3     sd_z_1     sd_z1      sd_z
    _____    ___    ________    ________    ________    _______    _______    _______    _______    _______    ____

      1       1        0         0.5442     0.41278     0.53795    0.49542    0.22935    0.22945    0.21653    0.252
      2       2        0        0.51894     0.52262     0.52544    0.45066    0.26787    0.23615    0.25833    0.311
      3       3        0        0.48248      0.5238     0.50392    0.46534    0.27009    0.26676    0.26644    0.294
      4       1        1        0.58343     0.50529     0.54361     0.5006    0.29578    0.30182    0.30952    0.273
      5       2        1        0.58408     0.45941     0.50466    0.40081    0.25026    0.34704    0.31039    0.286
      6       3        1        0.51148     0.49531     0.48963    0.47698     0.3271    0.24336    0.34498    0.340
```