String Array Manipulations, Join, Find, Replace and the Alphabet

back to Fan's Intro Math for Econ, Matlab Examples, or MEconTools Repositories

Generate a String Array from Strings

Empty String Array and fill with values.

```
ar_st_titles = strings([3,1]);
ar_st_titles(1) = 'Title1';
ar_st_titles(2) = 'Title2';
ar_st_titles(3) = 'Title3';
disp(ar_st_titles);
```

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

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
```

Given some previously defined chars or strings with single or double quotes, not sure which. To safely generate a string array, wrap in brackets and then conver with string function. This generates a string array whether the original inputs were single or double quoted:

```
st_a = 'a';
st_b = 'b';
st_c = 'c';
st_a_dq = "a";
st_b_dq = "b";
st_c_dq = "c";
ar_st_singlequotes = string({st_a, st_b, st_c});
ar_st_doublequotes = string({st_a_dq, st_b_dq, st_c_dq});
disp(["st_singlequotes" ar_st_singlequotes]);
```

```
    "st_singlequotes"    "a"    "b"    "c"
```

```
disp(["ar_st_doublequotes" ar_st_doublequotes]);
```

```
    "ar_st_doublequotes"    "a"    "b"    "c"
```

Convert the string array to a cell string array

```
disp(cellstr(ar_st_doublequotes));
```

```
    {'a'}    {'b'}    {'c'}
```

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'}
```

Joint String Cell Array with Suffix

```
ar_st_titles = {'Title1','Title2','Title3'};
disp(strcat(ar_st_titles, '_init'));
```

```
    {'Title1_init'}    {'Title2_init'}    {'Title3_init'}
```

Duplicate String

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

```
  Columns 1 through 9

    {'String'}    {'String'}    {'String'}    {'String'}    {'String'}    {'String'}    {'String'}    {'String'}

  Column 10
```

```
{'String'}
```

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 = 1×2 string
"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'
```

String Join dash (Paste)

This is similar to R's paste function:

```
st_var_name = "abc";
st_var_name = [st_var_name, 'efg', 'mqo'];
disp(strjoin(st_var_name, "_"));
```

```
abc_efg_mqo
```

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

```
abc,efg,mqo
```

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'
```

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'}
```

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
```

Find the positions of String Cells in Full String Cells

Find the positions of fl_w, fl_beta, and it_z_n in ls_st_param_key. Then just find the position of fl_crra. When looking for the position of something that does not exist, generate an find outcome array of length 0.

```
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 = 1×3
     3     2     6
```

```
find(strcmp(ls_st_param_key, 'fl_crra'))
```

```
ans = 1
```

```
length(find(strcmp(ls_st_param_key, 'fl_crra_not_exist')))
```

```
ans = 0
```

```
~sum(strcmp(ls_st_param_key, 'fl_crra_not_exist'))
```

```
ans = logical
   1
```

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
```

Generate Alphebetical String Array from A to Z

Generate a single string that is A to Z, then generate this as a string array.

```
% a to z single string
st_a2z = 'a':'z';
% a to z array of letters
ar_st_a2z = string(('A':'Z')');
% Display
disp(st_a2z);
```

```
abcdefghijklmnopqrstuvwxyz
```

```
disp(ar_st_a2z);
```

```
  Columns 1 through 18

    "A"    "B"    "C"    "D"    "E"    "F"    "G"    "H"    "I"    "J"    "K"    "L"    "M"    "N"    "O"    "P"

  Columns 19 through 26

    "S"    "T"    "U"    "V"    "W"    "X"    "Y"    "Z"
```