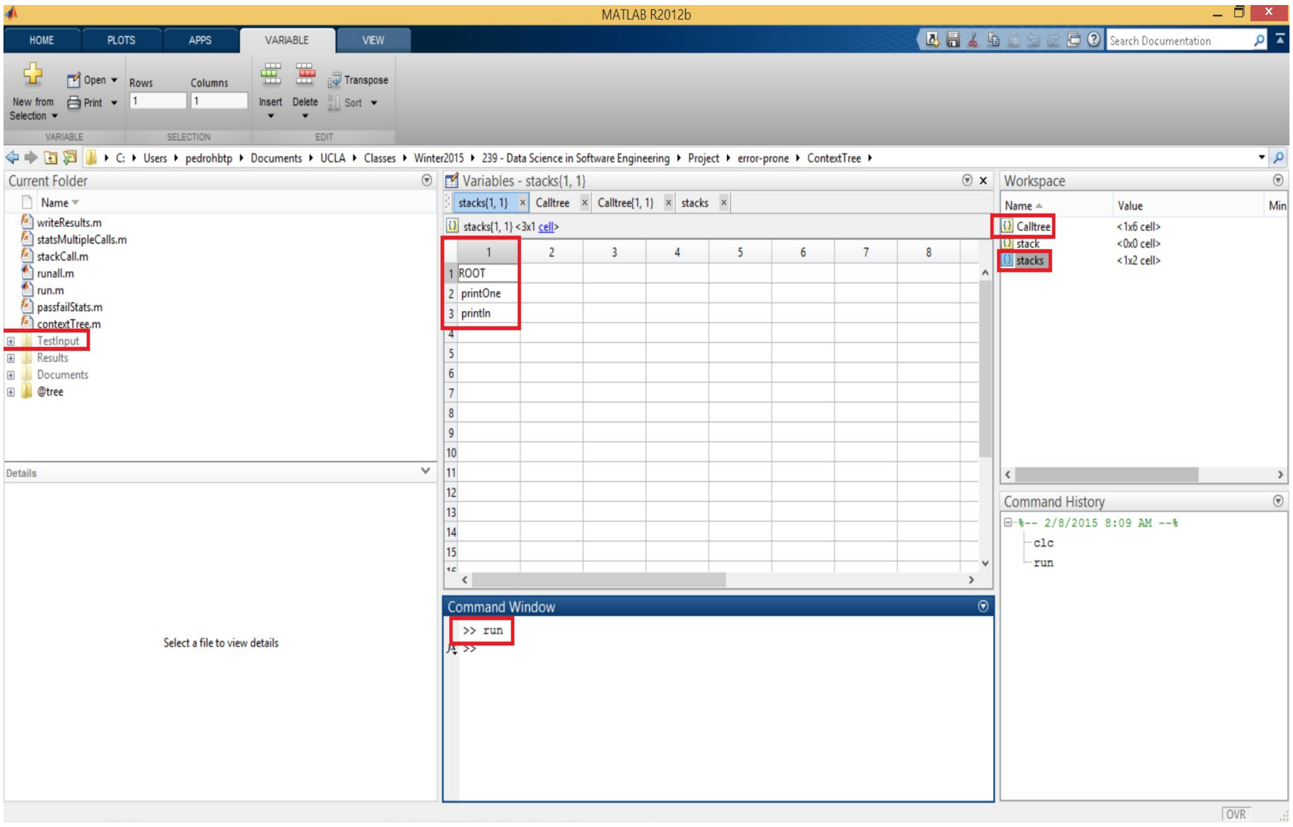
Obtaining Calling Context Trees and Corresponding Stacks using Matlab program

There program is intended to be used in two ways. One it reads a single file, builds the Calling Context tree from it and presents the stacks from the tree. The second way the program reads two sets of files. One containing logs of passed tests and the other containing logs of failed tests.

Single file

The picture below shows the execution of the construction of the tree and stacks from a single test file. This test file must be inside the folder called *TestInput.* The name of the file should be *input.txt.* On the image it is also possible to see the return values from that call.

- Calltree: Array of structures containing the information about each node. The information of each node is the following
  - Name : Name of the method that represents that node
  - Parent: Reference to the index of the parent of this onde.
  - Child: All the indexes of the children of this array
  - times_called: Number of times called in this specific path
  - index : Index of the node inside the Calltree array
  - stackbottom: A true or false value representing whether that node is the end of a stack
  - stack_shows: Only if the node is a stack bottom, it will have this field different than 0. It represents the number of times the stack was seen.
- Stacks: An array containing all the stacks. In the figure it you can see an example of a stack for the the test input data.



stacks{1, 1} <3x1 cell>

	1	2	3	4	5	6	7	8
1	ROOT							
2	printOne							
3	println							

Name	Value	Min
Calltree	<1x6 cell>	
stack	<0x0 cell>	
stacks	<1x2 cell>	

```
%-- 2/8/2015 8:09 AM --%
clc
run
```

```
>> run
>>
```

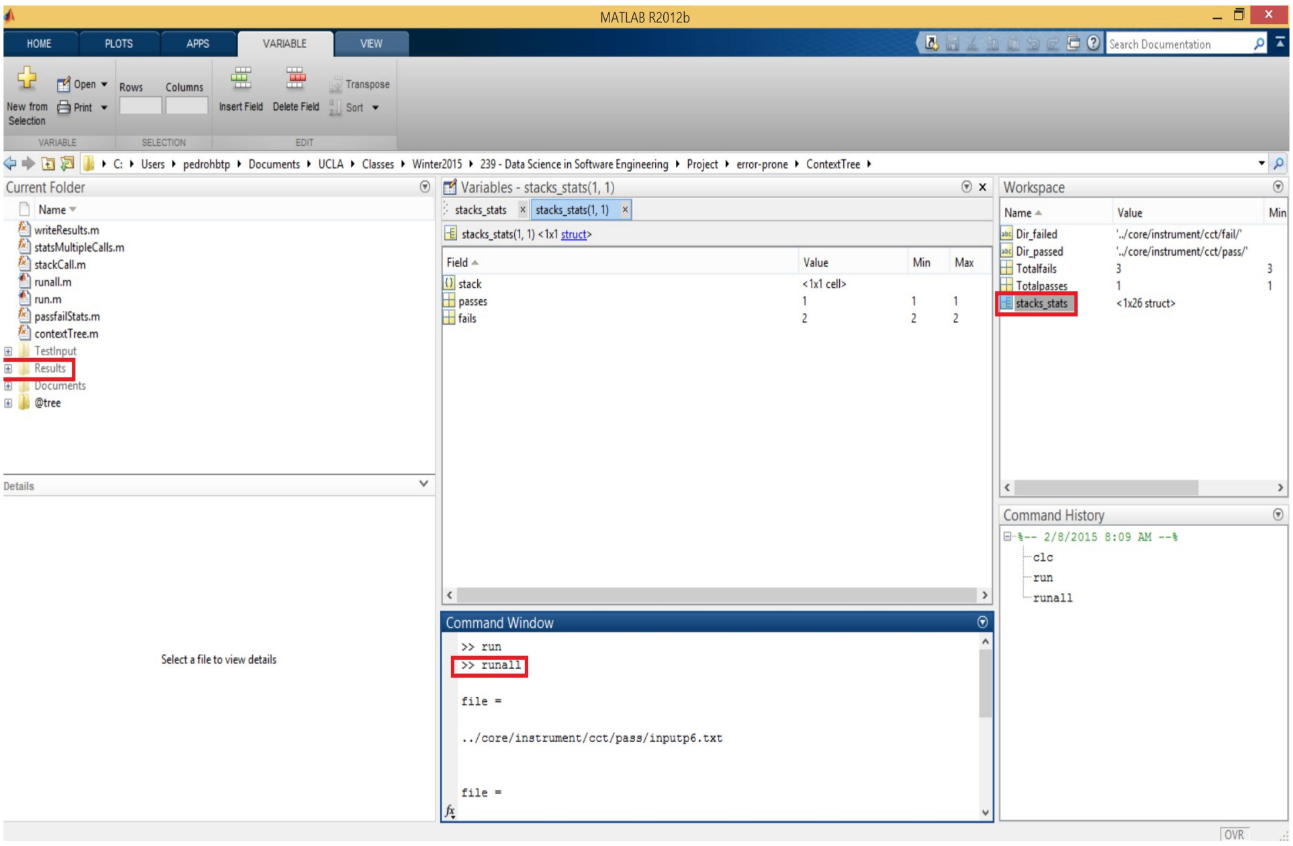
Multiple files

The figure below shows how to run the program for a set of tests. The set of log files for the failed tests should be in the directory *'../core/instrument/cct/fail/'* and the set of log files for the passed tests should be in the directory *'../core/instrument/cct/pass/'*.

To run, simply type *runall.*

The main output from that are all stacks from all the executions with annotations of how many times each were executed in a pass and in a fail tests. This information is in the *stacks_stats* object.

- **Stacks_stats:**
  - **stack:** Contains the stack obtained from the calling tree. Each element of the stack is the name of a method.
  - **Passes:** Integer representing the number of times that stack was seen on a pass test
  - **fails:** Integer representing the number of times that stack was seen on a fail test



stacks_stats(1, 1) <1x1 struct>

Field	Value	Min	Max
stack	<1x1 cell>		
passes	1	1	1
fails	2	2	2

Name	Value	Min
Dir_failed	'../core/instrument/cct/fail/'	
Dir_passed	'../core/instrument/cct/pass/'	
Totalfails	3	3
Totalpasses	1	1
stacks_stats	<1x26 struct>	

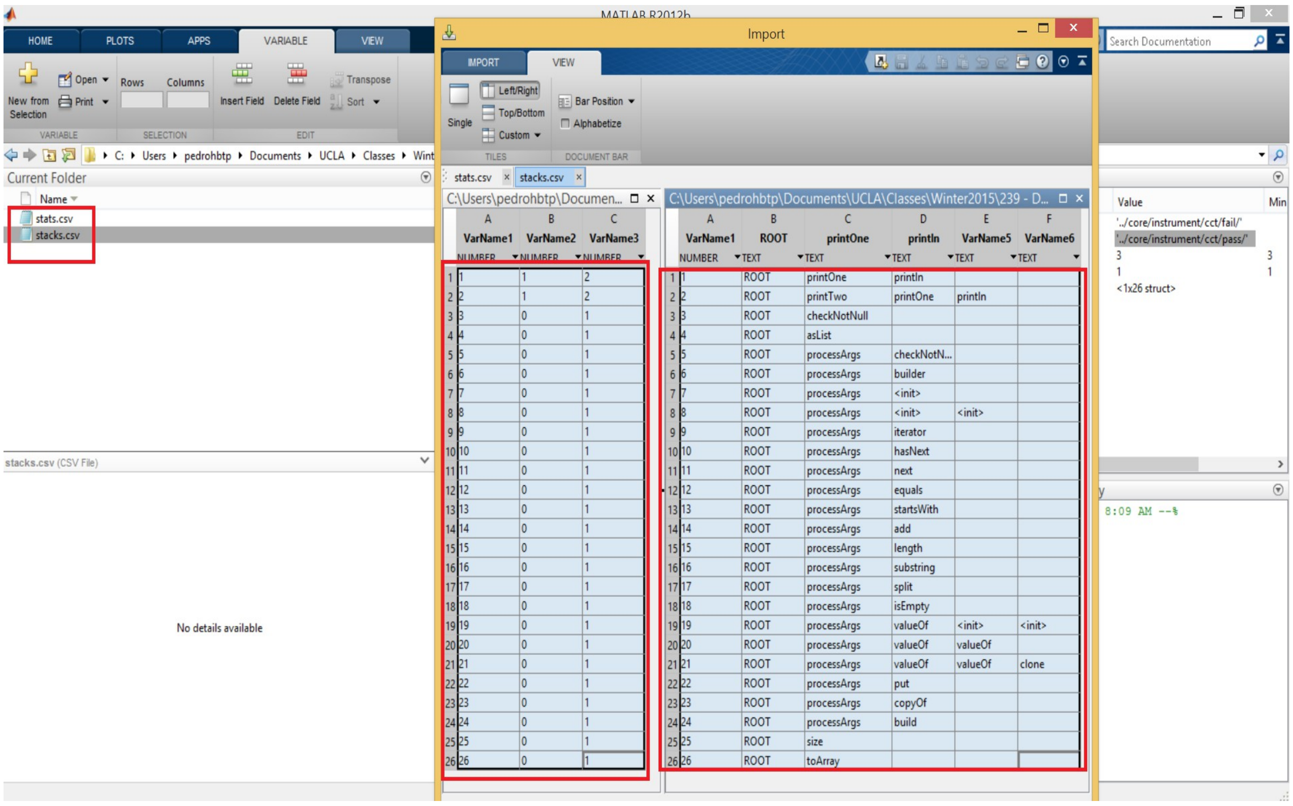
```
>> run
>> runall

file =

../core/instrument/cct/pass/inputp6.txt



file =
```

```
%-- 2/8/2015 8:09 AM --%
clc
run
runall
```

The information of the Stacks_stats is also written into two csv files in the directory *Results.*
Its format is seen in the picture below.
The format of the files is the folowing:

- stats.csv: Contains the information of number of passes and fails for each stack
  - First column: ID of the Stack
  - Second Column: Number of passes for that stack
  - Third Column: Number of fails for that stack

- stacks.csv
  - First column: ID of the stack
  - The other columns: The elements of each stack



stats.csv

A	B	C
VarName1	VarName2	VarName3
NUMBER	NUMBER	NUMBER
1	1	2
2	1	2
3	0	1
4	0	1
5	0	1
6	0	1
7	0	1
8	0	1
9	0	1
10	0	1
11	0	1
12	0	1
13	0	1
14	0	1
15	0	1
16	0	1
17	0	1
18	0	1
19	0	1
20	0	1
21	0	1
22	0	1
23	0	1
24	0	1
25	0	1
26	0	1

stacks.csv

A	B	C	D	E	F
VarName1	ROOT	printOne	println	VarName5	VarName6
NUMBER	TEXT	TEXT	TEXT	TEXT	TEXT
1	ROOT	printOne	println		
2	ROOT	printTwo	printOne	println	
3	ROOT	checkNotNull			
4	ROOT	asList			
5	ROOT	processArgs	checkNotN...		
6	ROOT	processArgs	builder		
7	ROOT	processArgs	<init>		
8	ROOT	processArgs	<init>	<init>	
9	ROOT	processArgs	iterator		
10	ROOT	processArgs	hasNext		
11	ROOT	processArgs	next		
12	ROOT	processArgs	equals		
13	ROOT	processArgs	startsWith		
14	ROOT	processArgs	add		
15	ROOT	processArgs	length		
16	ROOT	processArgs	substring		
17	ROOT	processArgs	split		
18	ROOT	processArgs	isEmpty		
19	ROOT	processArgs	valueOf	<init>	<init>
20	ROOT	processArgs	valueOf	valueOf	
21	ROOT	processArgs	valueOf	valueOf	clone
22	ROOT	processArgs	put		
23	ROOT	processArgs	copyOf		
24	ROOT	processArgs	build		
25	ROOT	size			
26	ROOT	toArray			

Value	Min
'../core/instrument/cct/fail/'	
'../core/instrument/cct/pass/'	
3	3
1	1
<1x26 struct>	