

Various Sample Runs on Vect

Vect(Visual Extraction Conversion Tool) can also be used on any data that we see in everyday life. In this Tutorial, we are going to introduce how to extract data from an example file using abortion statistics.

To start, click [here](#) to download the abortion data file. Save the file on Desktop and name it "abortionData".

First, we will extract all the states out of the table. Open the abortion data by click on Open button or from File->open.

1. Define open condition

Right select the first state (Alabama) in the table, than select *New Block Open Condition*. Everything above Alabama should be in pink; Alabama itself appears to be within the green region.

Age group		<15		15-19	
State		No.	%	No.	%
Alabama		190	1.3	3,046	21.4
Arizona		83	0.7	2,245	18.8
Arkansas		72	1.3	1,339	23.3

2. Define close condition

Right click and drag over to select the last state (Wyoming) in the table, than select *New Block Close Condition*. Everything below Wyoming should appear to be in the pink selected region. Wyoming itself is in the red region.

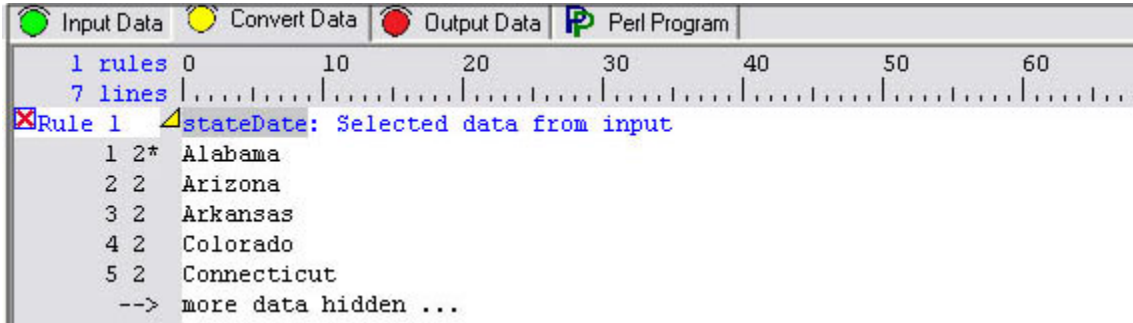
Washington	
West Virginia	
Wisconsin	
Wyoming	
Total	
Ratio	
Rate	
-- END OF FILE --	

Alabama	
Arizona	
Arkansas	
Colorado	
Connecticut	
Dist. of Columbia	
Georgia	
Hawaii	
Idaho	

3. Select Data Left click and drag over to select all of the state the state, make sure the longest state (Dist. Of Columbia) is selected.

4. Convert Data

After selecting the desired data, click on the Move button to transfer the selected data to the Convert Data panel. Name the new rule as “stateDate”. Go to the Convert Data panel. You should see the new rule is already there, it should look like the picture below.



Now you have all the states you want, select this rule and click on the Copy button to move the rule to the Output Data Panel. You can see the “<stateData>” is in the Primary Output column. Click on the Output button to see the output, or you can run the perl program under the Perl Program panel.

Now, let’s say if you want to add some data from the group of age less than 15. First, go back to Input Data panel, right click and drag over to select 190 under <15. No. column, and set it to *New Block Open Condition*.

	<15		15-19	
	No.	%	No.	%
1	190	1.3	3,046	21.4
1	83	0.7	2,245	18.8
1	83	0.7	1,888	18.8

Do the same to select 0 from the very end of the same column. Set it to *New Block Close Condition*.

51	2	Wisconsin	83	0.7
52	2	Wyoming	0	0.0
53	1	Total	5,949	0.8
54	1	Ratio	667	

Now, left click and drag over to select the whole column.

State	<15		15-19	
	No.	%	No.	%
Alabama	190	1.3	3,046	21.4
Arizona	83	0.7	2,245	18.8
Arkansas	72	1.3	1,339	23.3
Colorado	57	0.6	2,107	22.5
Connecticut	83	0.7	2,443	21.6
Dist. of Columbia &	46	0.7	1,170	18.2
Georgia	439	1.2	6,717	19.1

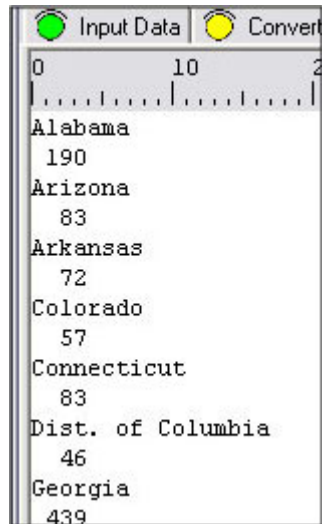
Using the Move button to transfer the data to the Convert Data Panel, name the new rule as “15LessData”. Go to Convert Data panel, copy “15LessData” to Output Data panel. There should be two rules under the Primary Output column.

```
Primary Output:
<stateData>
<15LessData>
```

The output will be very different if you arrange the two rules differently.

- If you arrange <stateData> on the first line, than <15LessData> on the second line, the output will look like this:

```
Primary Output:
<stateData>
<15LessData>
```



- If you arrange <stateData> and <15LessData> on the same line, the output will look like this:

```
Primary Output:
<stateData> |<15LessData>
```

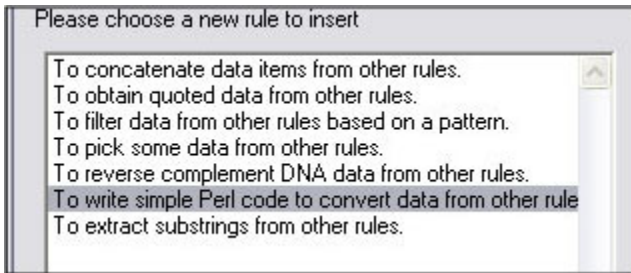
	0	10	20
Alabama			190
Arizona			83
Arkansas			72
Colorado			57
Connecticut			83
Dist. of Columbia			46
Georgia			439
Hawaii			56
Idaho			8

- If you arrange <15LessData> on the first line, <stateData> data on the second line, the output will look like this:

```
Primary Output:
<15LessData>
<stateData>
```

	0	10	20
190			
Alabama			
83			
Arizona			
72			
Arkansas			
57			
Colorado			
83			

You can also create your own rules to the data in Vect. In Convert Data panel, there is a rule under Insert Button, called “To write simple Perl code to convert data from other rule.”



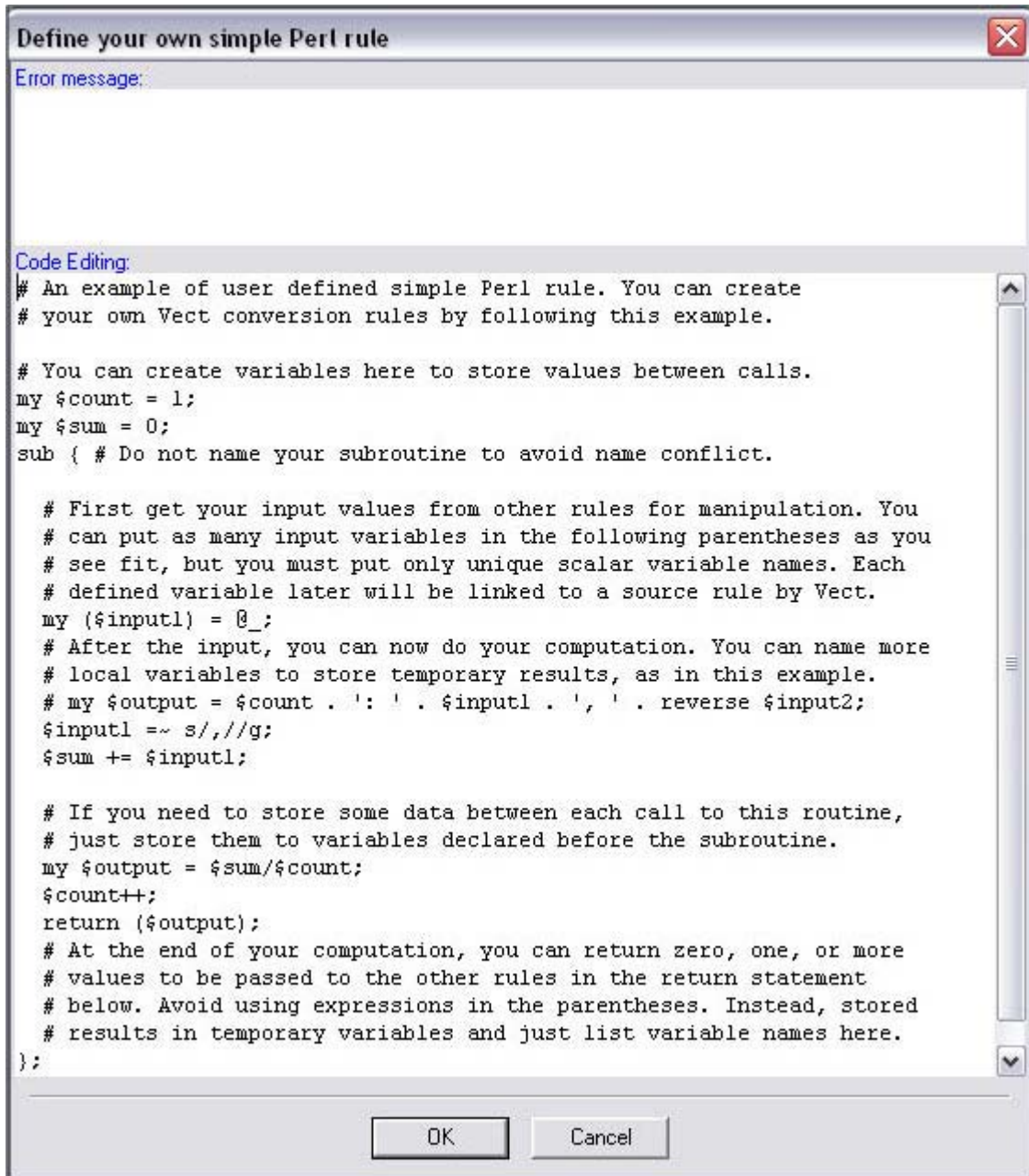
If you select the Perl code rule, and give it a name, it will bring a new rule to the Convert Data panel.

```
--> more data hidden ...
[X] Rule 3 [Y] SimpleRule: Simple user rule with input data from $input1 and $input2.
    -*- END OF DATA -*-
```

If you click on user rule, it will bring up a new window. There are already some simple perl scripts written in it. So, go ahead put in the data for input 1 and input 2. The data should look like this:

```
--> more data hidden ...
Rule 3 SimpleRule: Simple user rule with input data from stateData($input1) and
15LessData($input2).
1 2* 1: Alabama , 091
2 2 2: Arizona , 38
3 2 3: Arkansas , 27
```

You can write your own Perl script in user rule, look for other documents for more information about using perl. Let's say if you want to calculate the average of rule <15LessData>, here is an example of how to write such perl script.



The screenshot shows a dialog box titled "Define your own simple Perl rule" with a close button in the top right corner. It contains an "Error message:" field which is empty, and a "Code Editing:" area with a scroll bar. The code in the editing area is a Perl script that defines a subroutine to calculate the average of input values. The script includes comments explaining how to use input variables, perform calculations, and return results. At the bottom of the dialog are "OK" and "Cancel" buttons.

```
Define your own simple Perl rule
Error message:

Code Editing:
# An example of user defined simple Perl rule. You can create
# your own Vect conversion rules by following this example.

# You can create variables here to store values between calls.
my $count = 1;
my $sum = 0;
sub { # Do not name your subroutine to avoid name conflict.

    # First get your input values from other rules for manipulation. You
    # can put as many input variables in the following parentheses as you
    # see fit, but you must put only unique scalar variable names. Each
    # defined variable later will be linked to a source rule by Vect.
    my ($input1) = @_;
    # After the input, you can now do your computation. You can name more
    # local variables to store temporary results, as in this example.
    # my $output = $count . ': ' . $input1 . ', ' . reverse $input2;
    $input1 =~ s/,//g;
    $sum += $input1;

    # If you need to store some data between each call to this routine,
    # just store them to variables declared before the subroutine.
    my $output = $sum/$count;
    $count++;
    return ($output);
    # At the end of your computation, you can return zero, one, or more
    # values to be passed to the other rules in the return statement
    # below. Avoid using expressions in the parentheses. Instead, stored
    # results in temporary variables and just list variable names here.
};

OK Cancel
```

The results are:

```
Rule 3 SimpleRule: Simple user rule with input data from 15LessData($input1).
1 2* 190
2 2 136.5
3 2 115
4 2 100.5
5 2 97
6 2 88.5
7 2 138.571428571429
8 2 128.25
```