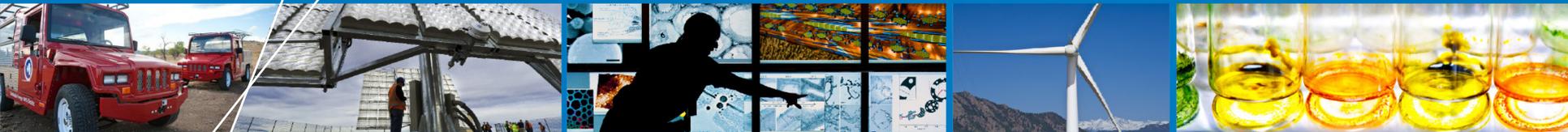


Scripting in SAM with LK



SAM Webinar

Aron Dobos

March 19, 2015

SAM Webinar Schedule for 2015



System Advisor Model

- **Introduction to New SAM!**
 - Nov 20, 2014: Janine Freeman
- **Sizing Photovoltaic Systems in SAM**
 - Dec 18, 2014: Janine Freeman
- **Parametric and Statistical Analysis in SAM**
 - Jan 22, 2015: Paul Gilman
- **Modeling Power Towers in SAM**
 - Feb 19, 2015: Mike Wagner
- **Scripting in SAM with LK**
 - Mar 19, 2015: Aron Dobos
- **Modeling Wind Systems in SAM**
 - Apr 16, 2015: Janine Freeman
- **SAM's Residential Third Party Financial Model**
 - Aug 20, 2015: Nate Blair
- **Battery Storage for PV Systems in SAM**
 - Sep 17, 2015: Aron Dobos

Details

- All sessions last one hour and begin at 1 p.m. Mountain Time
- You must register to participate
- Registration is free, but space is limited
- More details and registration information on Learning page of SAM website

<https://sam.nrel.gov/content/resources-learning-sam>

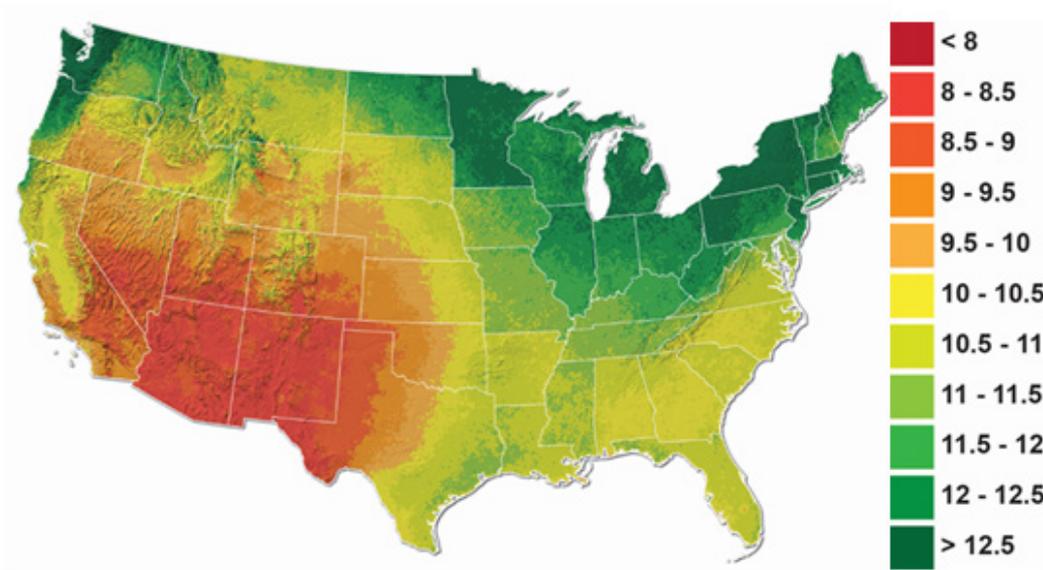


- **What are SAM LK scripts?**
- **What are some examples?**
- **How can I get started with scripting?**
- **How are Macros different from scripts?**
- **What are some useful features of LK?**
- **Demonstration**
- **Q & A**

Why scripting?



- Scripts let you control SAM automatically without clicking in the user interface or typing commands
- You can automate a large number of simulations



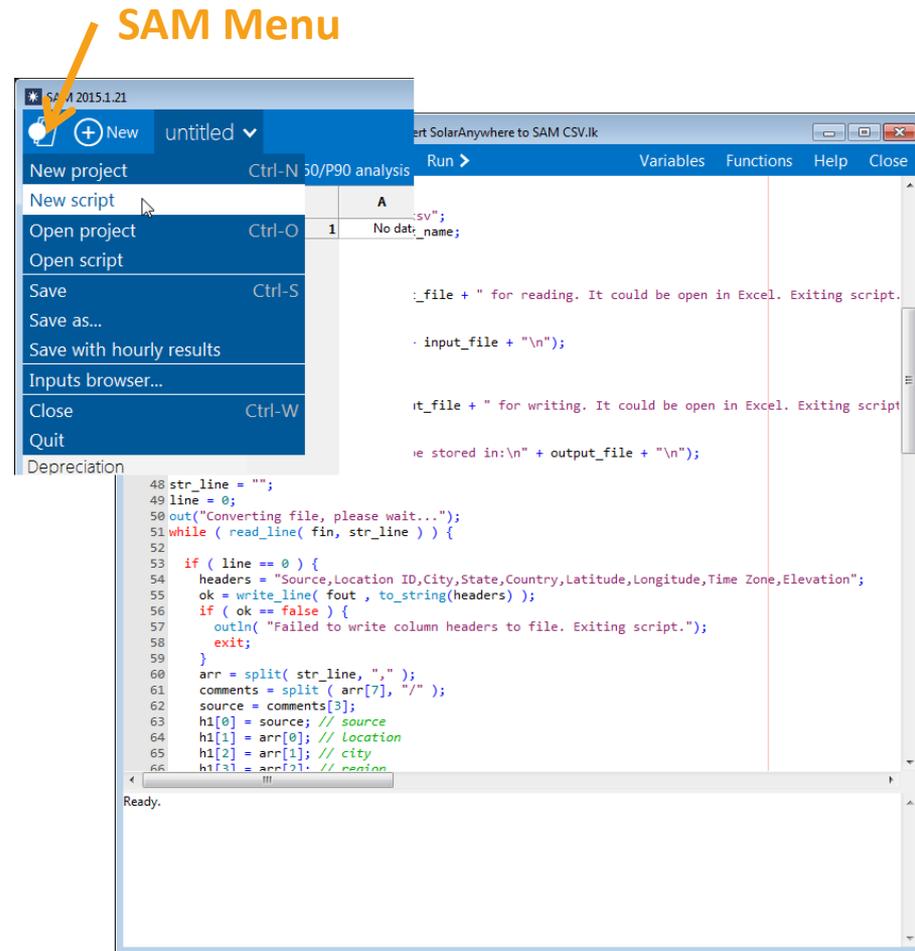
2014 modeled U.S. LCOE (\$ cent per kWh) map for a typical 50 MW (fixed-tilt) utility-scale PV system (assuming 25-year system life, 10% IRR target, 10.9% nominal discount rate, 50% debt fraction at 7% interest for 20 years, 30% federal ITC, and 5-year federal and state MACRS depreciation)

- You can customize output files, or read in inputs from another data source
- You can do custom post-processing of results and make your own calculations

What is LK?



- **SAM's built-in scripting language is called "LK"**
 - Analogous to Visual Basic for Applications (VBA) in Microsoft Excel
 - Easy-to-use but powerful language based around a few simple concepts
 - Very tightly integrated with SAM
 - Programs work the same way on Windows and OSX
- **Most flexible way to control and run simulations, but requires basic knowledge of computer programming**



Different from Software Development Kit, which allows you to run SAM algorithms from applications you write in C or other languages

Basic User Interface

The screenshot shows a script editor window titled "F:\SkyDrive\SAM\LK\Sample scraps from Aron\lk_features_test.lk *". The window has a menu bar with "New", "Open", "Save", "Save as", "Find", "Run", "Variables", "Functions", "Help", and "Close". The "Run" button is highlighted with a callout that says "Run the script". The code in the editor is as follows:

```
1 // This script tests some basic LK operations
2
3 outln("Number operators:");
4 a = 1;
5 outln(a);
6 a += 2;
7 outln(a);
8 a *= 2.5;
9 outln(a);
10 a -= 1;
11 outln(a);
12 a /= 3;
13 outln(a);
14 outln("");
15
16 outln("Concatenate strings:");
17 s = 'abc';
18 s += 'def';
```

Below the code is a console window showing the output of the script:

```
Number operators:
1
3
7.5
6.5
2.16667

Concatenate strings:
abcdefghi
```

There are three callouts pointing to the interface:

- "List available input and output variables." points to the "Variables" menu item.
- "List LK functions." points to the "Functions" menu item.
- "The console displays script output." points to the console window.

Quick Example

The screenshot illustrates a workflow in SAM 2015.2.12. The main window shows the 'Residential PV' project. The 'System Parameters' section is visible, with 'System nameplate size' set to 4 kWdc and 'Module type' set to Standard. The 'Array type' is set to 'Fixed open rack', with 'Tilt' at 21.5 degrees, 'Azimuth' at 180 degrees, and 'Age ratio' at 0.4. Other parameters like 'Shading' (3%), 'Snow' (0%), and 'Mismatch' (2%) are also visible.

A code editor window titled 'untitled *' is open, showing the following script:

```
1 active_case( 'Residential PV' );
2 set( 'tilt', 21.5 );
3 simulate();
4 energy = get('annual_energy');
5 outln( 'annual energy in MWh: ' + (energy / 1000 ) );
```

The 'Run' button in the code editor is marked with a red star. The output window shows the results of the simulation:

```
annual energy in MWh: 6.86476
Elapsed time: 0.4 seconds.
```

Annotations in the image include:

- 1: Points to the 'System Design' tab in the SAM interface.
- 2: Points to the 'Run' button in the code editor.
- 3: Points to the 'Simulate' button in the SAM interface.

Working with SAM Variables



The screenshot shows the SAM 2015.2.12 interface. The main window displays the 'System Parameters' section for a 'Residential PV' system. A code editor window titled 'untitled *' is open, showing the following code:

```
1 active_case( 'Residential PV' );  
2 set( 'tilt', 21.5 );  
3 set( '|', 190 );  
4 simulate();  
5 energy = get('ann  
6 outln( 'annual es
```

The 'Browse Variables' dialog box is open, showing the 'Available Variables' section for the 'PVWatts, Residential' case. The search field contains 'azi' and the text 'Start typing here...' is displayed. The 'Azimuth' variable is checked and listed as 'Azimuth {'azimuth'} (degrees) [number]'. Two orange arrows point to the 'Azimuth' variable, with the labels 'Label' and 'Variable name' below them. A red star is placed over the 'Variables' menu item in the code editor, and another red star is placed over the 'OK' button in the dialog box. An orange arrow labeled 'Case' points to the 'PVWatts, Residential' dropdown menu in the dialog box.



- **Data: numbers, text strings**
- **Conditional statements (if-then-else)**
- **Loops (while, for)**
- **Single and multi-dimensional arrays**
- **Tables (hash tables/dictionaries)**
- **User-written functions**
- **Built-in library of functions**
 - Input and output (text console and graphical)
 - Math functions
 - Functions for working with text strings
 - Reading and writing files, scanning folders, etc.
 - Reading and writing CSV data files
 - Exchanging data with Microsoft Excel
 - Plotting functions for visualizing data
 - Functions for working with web services and downloading
 - Monte Carlo sampling (LHS) and stepwise regressions
 - SAM-specific functions for working with cases, variables, and results

Working with Built-in Functions



Scripting Reference

- active_case** ([string:case name]):variant
Sets the currently active case, or returns its name.
- varinfo** (string:var name):table
Gets meta data about an input or output variable. Returns null if the variable does not exist.
- select_inputs** (array:checked>, [string:title]):boolean
Shows the input variable selection dialog.
- set** (string:name, variant:value):none
Set an input variable's value. Issues a script error if the variable doesn't exist or there is data type error.
- get** (string:name):variant
Get an output or input variable's value.
- simulate** ([string:messages], [boolean:update UI]):boolean
Run the base case simulation for the currently active case. Errors and warnings are optionally returned in the first parameter. The results in the user interface are not updated by default, but can be via the second parameter.
- show_base** (string:base name):boolean

```
1 active_case( 'Residential PV' );
2 set( 'tilt', 21.5 );
3
4 sim
5 ene
6 out
```

annual energy in MWh: 6.86476
Elapsed time: 0.4 seconds.

Shading	3 %
Snow	0 %
Mismatch	2 %



- SAM help system includes an overview
- Full reference manual is included (click “Scripting Reference” in Help window)
- LK Cookbook of Examples (in progress)
<https://sam.nrel.gov/sample>
- Search “LK” in online forum

```
degraded_output = degraded_output - degraded_output * 0.1;
// use parentheses to subtract before multiplication
cash_amount = total_cost * ( 1 - debt_fraction/100.0 );
```

2.4 Simple Input and Output

You can use the built-in `out` and `outln` functions to write textual data to the console window. The difference is that `outln` automatically appends a newline character to the output. To output multiple text strings or variables, use the `+` operator, or separate them with a comma. Note that text strings may be enclosed by either single or double quotes - both have the exact same meaning.

```
array_power = 4.3k;
array_eff = 0.11;
outln("Array power is " + array_power + " Watts.");
outln("It is " + (array_eff*100) + " percent efficient.");
outln("It is ", array_eff*100, " percent efficient."); // same as above
```

The console output generated is:

```
Array power is 4300 Watts.
It is 11 percent efficient.
```

Use the `in` function to read input from the user. You can optionally pass a message to `in` to display to the user when the input popup appears. The `in` function always returns a string, and you may need to convert to a different type to perform mathematical operations on the result.

```
cost_per_watt = to_real(in("Enter cost per watt:")); // Show a message. in() also is fine.
notice("Total cost is: " + cost_per_watt * 4k + " dollars"); // 4kW system
```

The notice function works like `out`, except that it displays a pop up message box on the computer screen.

2.5 Data Types and Conversion

LK supports two basic types of data: numbers and text. Numbers can be integers or real numbers. Text strings are stored as a sequence of individual characters, and there is no specific limit on the length of a string. LK does not try to automatically convert between data types in most cases, and will issue an error if you try to multiply a number by a string, even if the string contains the textual representation of a valid number. To convert between data types, LK has several functions in the standard library for this purpose.

Boolean (`true/false`) data is also stored as a number - there is no separate boolean data type. All non-zero numbers are interpreted as `true`, while zero is `false`. This convention follows the C programming language.

7

Macros vs Scripts



Macros are LK scripts designed to let people run them who don't have scripting knowledge

- You can package up a script as a SAM macro, and others can run your code without being intimidated
- Simple user interface and documentation is possible
- SAM includes some Macros to add custom calculations or automate some tasks

The screenshot shows the SAM 2014.10.1 interface. The 'Simulate' menu is open, highlighting the 'Macros' option. A list of macros is displayed, including 'Auxiliary Gas Heater', 'Download Electric Load', 'Create a Tornado Chart', 'Sizing Considerations', 'Weather File Checker', and 'Weather File Converter'. The 'Auxiliary Gas Heater' macro is selected, showing its description and input fields. The 'Run macro' button is highlighted. The console at the bottom shows simulation results.

1. Click **Macros to show a list of macros available for the case.**

2. Click a macro's name to see its description and inputs.

3. Review and change the macro's inputs.

4. Click **Run macro to start it.**

5. Check the console for messages and results. Some macros display results in a separate window.

You can always see the code behind a macro

Energy from gas saved with solar = 6,274,487 kWh/year
Energy from gas used with no solar = 4,118,094 kWh/year
Annual solar fraction = 1.61
Number of hours with solar fraction of one = 2

Case study: Batch Calculation of PV Economics



Input: spreadsheet of 30 buildings with PV

- Street address
- PV system size
- Cost
- Utility Rate

Site Name	Address	Estimated Roof Size Sq Ft	Actual Roof Size	PV Size (kW) Est by DOE	Actual or Estimated Size by Solicitation Process
Little Rock Old Post Office Courthouse	300 West Second Street Little Rock AR 721	18,502		172	
Joseph P. Addabbo Federal Building	15902 Jamaica Ave Jamaica NY	76,933		715	
Richard H Chambers U.S. Courthouse	125 South Grand Avenue Pasadena CA 91	13,533		127	
William R. Cotter Federal Building	135 High Street Hartford CT 06103	38,289		356	
Elijah Barrett Prettyman Building	333 Constitution Avenue NW Washington D	104,623		972	
Howard T. Markey National Courthouse	717 Madison Place NW Washington DC 2	21,787		202	
Robert C. Weaver Building	451 Seventh Street SW Washington DC 2C	104,651		972	
Theodore Roosevelt	1900 E ST NW WASHINGTON DC 20415-(78,130		726	
Winder Building	600 17th Street NW, Washington DC 2050	16,186		150	
James A McClure Federal Building Courthouse	550 West Fort St Boise ID 83724-0042	27,592		256	
Senator Paul Simon Federal Building	250 West Cherry Street Carbondale Illinois	17,889		166	300kw from ARRA Sht
Major General Emmett J. Bean (Phase I - PV and	8899 East 50th Street Indianapolis IN 462	415,060		3856	
Thomas P. O'Neill Jr. Federal Building	10 Causeway Street Boston MA 02222-101	65,297		607	
	7500 Security Boulevard Baltimore MD 212	66,392		617	
	7500 Security Boulevard Baltimore MD 212	55,587		516	
	7500 Security Boulevard Baltimore MD 212	14,481		135	CMS Complex combined
	7500 Security Boulevard Baltimore MD 212	57,622		535	
CMS Headquarters Complex	310 New Bern Avenue Raleigh NC 27601	42,603		396	
Edward Zorinsky Federal Building	1600 Capitol Ave Omaha NE 68102	36,006		335	
Robert Merriage Courthouse	RICHMOND VA 23219-1833	61,250	#####	476	

Result: In less than 100 lines of script code, the whole simulation process was automated for each building. Weather data was automatically downloaded for each address from the NREL NSRDB, and the simulation results were written to a CSV file.

Output: spreadsheet with results for each system

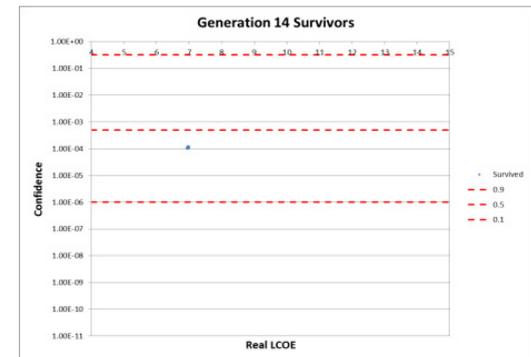
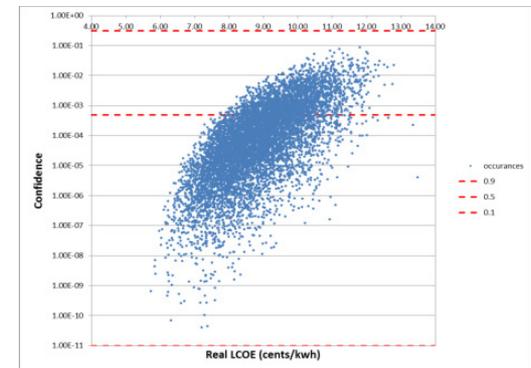
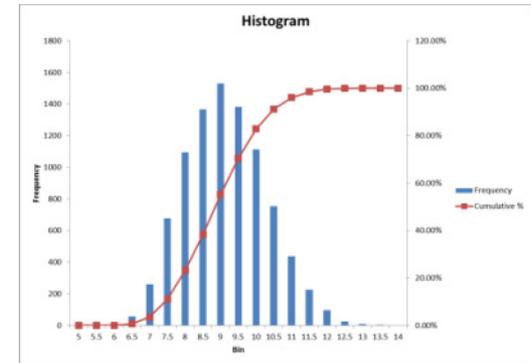
- LCOE real & nominal
- Annual system output
- Payback period

```
C:\Users\adobos\Desktop\building_batch.lk *
New Open Save Save as Find Run Variables Functions Help Close
1 workdir = homedir() + "/My Documents/federal_buildings_samu1";
2 if (! dir_exists(workdir))
3   mkdir(workdir);
4
5 inputfile = workdir + "/input.csv";
6 outputfile = workdir + "/output.csv";
7
8 fin = open(inputfile, "r");
9 if (!fin)
10 {
11   outln("Could not open input data file.");
12   exit;
13 }
14
15 fout = open(outputfile, "w");
16 if (!fout) {
17   outln("Could not open output file for writing.");
18   close(fin);
19   exit;
20 }
21
22 write(fout, "Num,Building,Address,WeatherFile,Latitude,Longitude,DCRate,");
23 write(fout, "ModuleCost,TotalInstalled,$perKw,UtilityRateC/kwh,");
24 write(fout, "LCOEnom,LCOEreal,ENet,Payback,LCOEnomFlat,LCOErealFlat,ENetFlat,PaybackFlat\n");
25
26 active_case("Federal Building PV");
27
28 buffer = "";
29 read_line(fin, buffer); // skip the header line
30 while ( read_line( fin, buffer ) )
31 {
32   cols = split(buffer, ",");
33
34   building = cols[1];
35   address = cols[2];
36
37 }
Ready.
```

Case study: CSP System LCOE Optimization



- Programmed a basic Genetic Algorithm in script to optimize a CSP power tower design to minimize the Levelized Cost of Energy (LCOE)
- Use sampling functions to generate configurations within ranges of selected parameters for the system
- Generate sufficient number of configurations to have a representative number of ≈ 6 cent results (e.g. 10,000).
- Calculate “risk” associated with each configuration.
- Write Genetic Algorithm routine to minimize LCOE subject to risk and variable constraints
- Write results to files for analysis

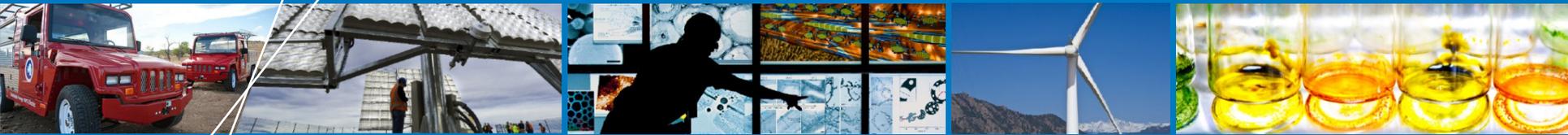




Question: What is the best DC/AC ratio to use for a system in Portland, OR?

Steps:

- 1. Create a case that represents your system**
- 2. Load an input file created in Excel**
- 3. Run all of the simulations**
- 4. Save the results to a CSV file**
- 5. Plot the results and save an image**



Thank you! Questions?