

Stock analysis Language (SLang)

Ray Siu

rls2121 (at) Columbia (dot) edu

Table of Contents

Stock analysis Language (SLang).....	3
Introduction.....	3
SLang	3
Simple	3
Powerful.....	3
Platform independent	4
Example	4

Stock analysis Language (SLang)

Overview

Introduction

There are many languages that allow developers to make and test custom formulas for stock analysis; however these languages are either too broad or hyper proprietary. Languages like MatLab and Java can be used for stock analysis purposes however they were not designed specifically to do so and as such go overboard with many unused and unnecessary features. Other languages like AFL are very good but are tied in with specific services like AmiBroker which can get quite expensive. SLang is a simple alternative to currently available tools by providing only a bare bone and necessary functionality.

SLang

SLang is a language with a simple set of math related functions and a data structure set designed specifically for stock analysis. It allows the developer to use the simple default functions of SLang to build more complex custom functions.

Simple

SLang offers simplicity in two ways: simplicity of architecture and ease of use. Languages like AFL use an expensive service to pull data into the code. SLang is free of such dependencies by allowing the user to read in any text file with stock data in a specific format. This allows the user to obtain stock data from any source they want or even create their own hypothetical price histories. SLang also offers a streamlined syntax that is easy to understand with a set of functionalities that is simple but powerful.

Powerful

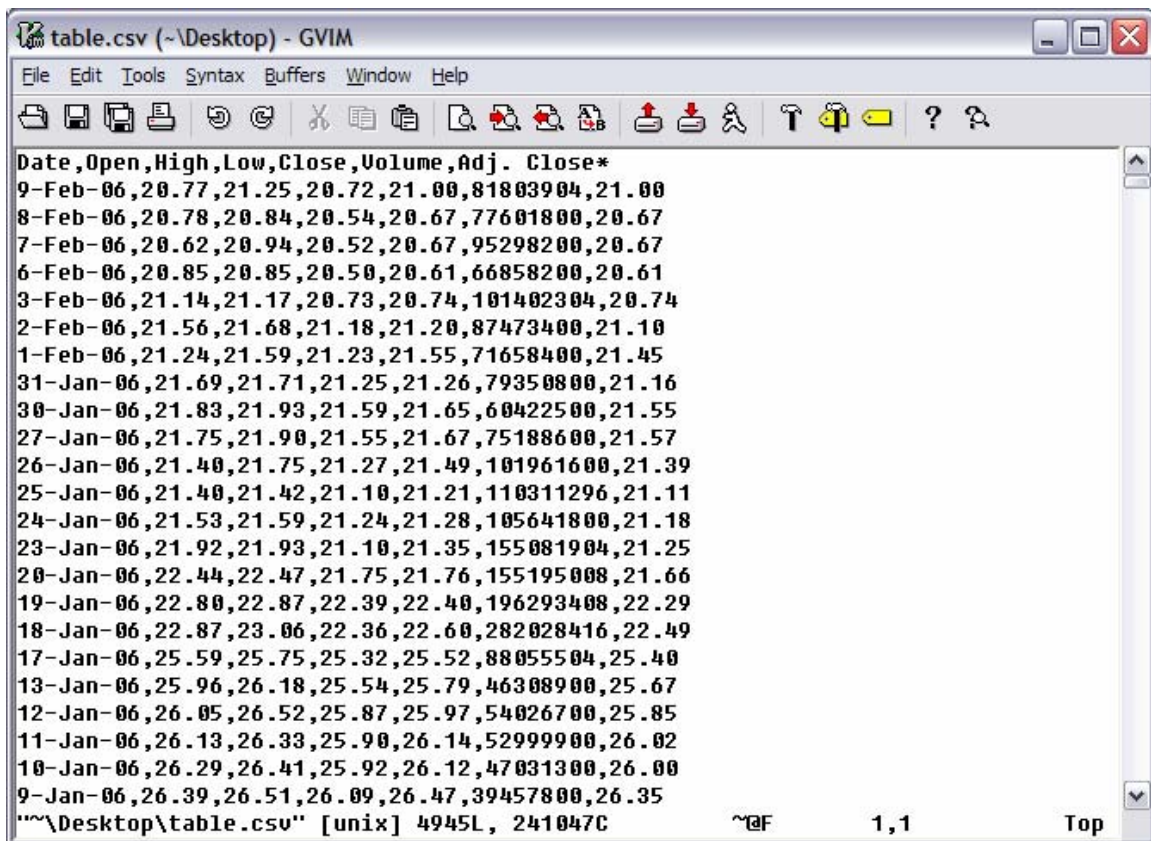
SLang has a custom data structure designed specifically to hold stock information as well as an easy interface whereby data from a file can be populated. Additionally SLang defines a set of commonly used functionalities and simple data types that will allow the user to compose new functionality. SLang will also allow the user to print results to STDOUT where it can be manipulated by other programs or visualized by other graphing applications like GNU plot.

Platform independent

The SLang compiler will be JAVA based and so will be highly portable between systems. Also, all file I/O will be distinguished via URIs which will free the language from being tied down to any particular OS or file system implementation.

Example

Before we get into the specifics of SLang we must first specify the data format that the language expects. Since Yahoo's Y! Finance service seems to be the more popular source for free stock price back data, we choose to use a simple CSV format based on Yahoo's format.



```
table.csv (~\Desktop) - GVIM
File Edit Tools Syntax Buffers Window Help
Date,Open,High,Low,Close,Volume,Adj. Close*
9-Feb-06,20.77,21.25,20.72,21.00,81803904,21.00
8-Feb-06,20.78,20.84,20.54,20.67,77601800,20.67
7-Feb-06,20.62,20.94,20.52,20.67,95298200,20.67
6-Feb-06,20.85,20.85,20.50,20.61,66858200,20.61
3-Feb-06,21.14,21.17,20.73,20.74,101402304,20.74
2-Feb-06,21.56,21.68,21.18,21.20,87473400,21.10
1-Feb-06,21.24,21.59,21.23,21.55,71658400,21.45
31-Jan-06,21.69,21.71,21.25,21.26,79350800,21.16
30-Jan-06,21.83,21.93,21.59,21.65,60422500,21.55
27-Jan-06,21.75,21.90,21.55,21.67,75188600,21.57
26-Jan-06,21.40,21.75,21.27,21.49,101961600,21.39
25-Jan-06,21.40,21.42,21.10,21.21,110311296,21.11
24-Jan-06,21.53,21.59,21.24,21.28,105641800,21.18
23-Jan-06,21.92,21.93,21.10,21.35,155081904,21.25
20-Jan-06,22.44,22.47,21.75,21.76,155195008,21.66
19-Jan-06,22.80,22.87,22.39,22.40,196293408,22.29
18-Jan-06,22.87,23.06,22.36,22.60,282028416,22.49
17-Jan-06,25.59,25.75,25.32,25.52,88055504,25.40
13-Jan-06,25.96,26.18,25.54,25.79,46308900,25.67
12-Jan-06,26.05,26.52,25.87,25.97,54026700,25.85
11-Jan-06,26.13,26.33,25.90,26.14,52999900,26.02
10-Jan-06,26.29,26.41,25.92,26.12,47031300,26.00
9-Jan-06,26.39,26.51,26.09,26.47,39457800,26.35
""\Desktop\table.csv" [unix] 4945L, 241047C ~ 1,1 Top
```

Each new line is a day and each day has the following values separated by commas: date, opening stock price, day's high, day's low, closing price, volume, and adjusted closing price.

We would provide some simple interface to grab the file by specifying a URI for a local or network resource with this CSV:

```
SDATA intel = OPEN(file:///c:/local/data/intc.csv);
```

We could then provide simple interfaces into this stock data.

```
INT size := intel::SIZE;
INT index := 0;

DOUBLE open := 0.0;

FOR(index := 0; index < size; index := index + 1)
{
    IF(open < intel::OPEN(index))
    {
        open := intel::OPEN(index);
    }
}

PRINT(open);
```

The above program will get the largest opening price for Intel for the period reported by “intc.csv”.