Draft - Language Reference Manual
Equity Portfolio Statistical Analysis Language (EPSAL)

May 3rd, 2009 v 1.0

Charles Neher
cgn2103@columbia.edu

COMS W4115
Programming Languages and Translators
Professor Stephen Edwards
Spring 2009 Semester

Table of Contents[1]

[1] This LRM has referenced the framework of The C Programming Language by Brian Kernigan and Dennis Ritchie pages – LTM 191 -231 – Prentice Hall , 1988

1. Language Introduction

This manual describes the EPSAL (Equity Portfolio Statistical Analysis Language) which is a Statistical computing language similar to simple version of the R [2] with built in function specific to Equity time series analysis. The most important aspect of the language is for the user to have to the ability to express algorithms in the language to write their own statistical functions with a simple standard library and set of operators that can be combined implement an algorithm in an optimal fashion.

2. Document Conventions

This manual will have 2 distinct syntax notations when describing language notations. Syntactic categories are indicated in Ariel *italic* type and literal words and characters will be in bold **Lucidia Sans Typewriter 12 point.**

3. Lexical Conventions

3.1. Tokens

There are 5 classes of tokens in the EPSAL language: Keywords, identifiers, constants, string literals, and operators. All blanks, newlines, vertical tabs and comments as described in section 3.2 are considered whitespace and ignored during tokenization.

3.2. Comments

The Characters /* introduces a comment and terminates with */ and will be ignored by the complier. There are no multiline comments, or nesting, and they should not occur within a string or character literals

3.3. Identifiers

An identifier is used for variable names and consists of a series of letters and digits. First character must be a letter and may be a combination of Upper and lower case letters as well as digits. The language is not case sensative.

[2] http://wiki.r-project.org/rwiki/doku.php

3.4. Keywords

The following identifiers are reserved for keywords and may not be used in any other context except a componenet of a combination of letters or digits.

Keyword	purpose
DIST	Distribution
TS	time series
PRINT	Printing
ERROR	Error handling
IO	Input / output
ARRAY	Array
LIST	List
CHAR	Character
INT	Integer
FLOAT	Floating Point
IF	Program control
THEN	“ “
ELSE	“ “
RETURN	“ “
GOTO	“ “
BREAK	“ “
CONTINUE	“ “

3.5. Constants

There are 4 types of constants – Integer, Character, Floating and String Literals

3.6. Integer Constants

An integer constant is a series of digits and may not begin with 0

3.7. Character Constants

A character constant is a sequence of one or more characters enclosed by single quotes ‘A’.

3.8. Floating Constants

A floating constant consist of an integer part, decimal part, a fraction part, an e or E, and an optionally signed integer exponent. A digit string may be of any length and the location of the radix point by a dot. If the radix point is omitted it is assumed to lie to the right end of the string.

Either the integer part of the fraction part (but not both) may be missing.

3.9. String Literals

A String literal or constant is a sequence of characters surrounded by double quotes “ “. A string has an array of characters and is initialized with the given characters. The compiler puts a null byte \0 which is appended to single string or concatenated string so that he program can find its end.

4. Expressions

In EPSAL an expression is a sequence of operators and applicable operands whose highest order precedence relates the organization sub sections in chapter 4 of this LRM.

4.1. Primary Expressions

Primary expressions are identifiers, numbers, strings or expressions in parenthesis.

Primary-expression:

Identifier

Constant

String

(expression)

An identifier is a primary expression provided that is has been declared with accepted type of int, float or string.

Arrays

A postfix expression followed by an expression in square brackets is a postfix expression denoting a subscribed array reference. The index of the array is specified by integer constants.

Example: identifier = expression [integer constant]

4.2. Functions

A function call is primary postfix expression followed by the parentheses with a list of comma separated list of assignment expressions which represent the input parameters of the function. The function itself is a sequence of statements that result in a return value or call another function in relation to a recursive algorithm.

Example : example Func int identifier(assignment expression 1, assignment expression 2, etc):

4.3. Multiplicative Operators

The multiplicative operators *, / , % group left to right. If one of the operands is 0 in the division operation the result is undifined. If both operands are non-negative, the remainder is also non-negative and smaller than the divisor.

Multiplicative-expression:

=expression * expression
= expression / expression

% operator is the remainder of the first operand by the second

4.4. Additive Operators

The additive operators + and – group left to right. The rest of the + operator is the sum of the operands and the - operand is the difference. The syntax is as follows:

Additive-expression:
Additive-expression + Additive-expression
Additive-expression - additive-expression

4.5. Relational Operators

The relational operators group left-to-right and evaluates to either 0 or 1

The operators <(less), > (greater), <= (less or equal), >= (greater of equal), == (equal to) all yield 0 if false and 1 if true.

relational-expression:

relational_expression < relational_expression
relational_expression>relational_expression
relationa_expression<=relational_expression
relational_expression>=relational_expression
relational_expression==relational_expression

4.6. Logical AND Operator

Operators of AND logical type are Boolean and equal to 1if both operands are not equal to 0 and 0 for all other cases. The operands must be on same type and are valuated from left to right.

logical-expression AND logical-expression

4.7. Logical OR Operator

Operators of OR logical type are Boolean and equal to 1if either operands are not equal to 0 and 0 for all other cases. The operands must be on same type and are valuated from left to right.

logical-expression OR logical-expression

4.8. Assignment Operator

In Epsal the '=' operator is used, evaluated from left to right and the operand on far left is reuired to an identifier and not a function. All the types must also be the same and the value stored in the operand is the value of expression to right of the '=' sign.

assignment-expression:
Identifier = operator assignment-expression

4.9. Comma Operator

The comma operator is indicated by the comma symbol (,) and evaluated left to right and separates expressions, identifiers and operators as well as data point delineations when reading in or writing to file calculation results.

Expression:
Assignment-expression
Expression, assignmenet-expression

5. Declarations

Declarations specify the interpretation given to each identifier by the language

5.1. Type specified

The type specifiers are:

Int
Float
Char
Array
String

6. Statements

Statements are executed in sequence are fall into 4 specific types:

Statement:
Expression –statement
Selection-statement
Iteration-statements
Jump-statement

6.1. Expression Statement

A majority of statement in EPSAL will be expression statements which have the form

Expression-statement:
Expression;

6.2. Selection Statement

Selection statements have 2 flows of control

Selection-statement

If (expression) statement

If (expression) statement **else** statement

If first condition in statement is true the statement is executed. In second expression the statement is executed should the first expression is false.

6.3. Iteration Statements

Iteration statements specify looping operation

Iteration-statement

While (expression) statement

Do statement **while** (expression) ;

In the while statement the statements is executed while the expression is true. In the Do-while statement the expression is executed while the statement is true.

6.4. Jump Statement

Jump-statement

Goto identifier;

Break;

In a goto statement transfers to the labled identifier. In the break statement in applicable to stooping the operation of an interation loop.

7. Appendix [3]

Financial functions

Holding Period Return – HPR (RT, PT, PT1, DT)

The holding return period formula is used for time-weighted return measurement.

$R_t = [(P_t – P_{t-1} + D_t)/ P_{t-1}]$

RT -*R_t = holding period return for time period (t)*

PT -P_t = price of asset at end of time period t

PT1 -P_{t-1} = price of asset at end of time period (t – 1),

DT -D_t = cash distributions received during time t

Weighted MEAN – WMEAN (AC1, AC1W, AC2, AC2W, AC3, AC3W)

Weighted Average or Mean – for 3 asset classes max

The weighted mean takes the mean return of each asset class and weights it by the allocation of each class. The weighted mean is calculated by weighting the return on each class and summing:

Weighted mean is frequently seen in portfolio problems in which various assets classes are weighted within the portfolio – for example, if stocks comprise 50% of a portfolio, then 0.5 is the weight. A weighted mean is computed by multiplying the mean of each weight by the weight, and then summing the products.

Example

Stocks - AC1 = .6

Stocks weight – ACW1 = .1

Cash – AC2 = .3

Cash Weight = ACW2 = .06

Bonds – AC3 = .1

Bonds AC3W = .02

WMEAN = 8%[4]

[3] http://www.investopedia.com/exam-guide/cfa-level-1/quantitative-methods

[4] http://www.investopedia.com/exam-guide/cfa-level-1/quantitative-methods/statistical-return-calculations.asp

Sharpe Ratio - SR (MR,RFR, SDR)

Is a measure of the risk-reward tradeoff of an investment security or portfolio. The Sharpe ratio is calculated by dividing the ratio of excess return, to the standard deviation of return.

Sharpe ratio = [(mean return) – (risk-free return)] / standard deviation of return

MR – mean return

RFR – Risk free return

SDR – Standard Deviation of Return