.

Formula Primer

Introduction

Equis has been supporting MetaStock for some years now and has frequently had questions from customers on how to write their own indicators, system tests, or explorations. The commands are printed in the manual, but most inquiries stemmed from people who had no idea how to begin. The MetaStock formula language involves some basic programming concepts and this seemed to have frightened many users into not even trying.

This text explains, in small incremental steps, what the MetaStock formula language is and how it works. Soon you'll know how to write a MetaStock function. Confidence will increase and you'll bravely combine two functions into a single indicator. Amazed at how easily you did this, you'll start using more until you suddenly realize, you are writing your own system tests and explorations.

If you don't believe this, then start reading chapter one. Be patient and follow the exercises. Please type in the formulas as they are shown and try them in MetaStock. You will get the most from this text if you actually do the exercises while reading the relevant sections. The more you use a program, the more comfortable you will be with it.

Conventions used in this Document

Throughout this text, you will receive instructions on using MetaStock and creating sample formulas. Below is a list of the symbols and special text used to distinguish these instructions and alert you to possible points of interest.

Bold Bolded text refers to menu commands. Each step will be separated by the | symbol. For example **File | Open** means click on File and then click Open

`Font` Text typed in this font denotes commands for you to type in your computer.

Italics Italics are used to denote new terms. A definition will always follow the word or term. For your convenience a complete list of all these terms is duplicated in a glossary at the back of this text.

💡 This symbol is used to point out a section of text you should pay extra attention to.

During the exercises, you will occasionally open charts and perform other basic MetaStock tasks. No step-by-step instructions will accompany these tasks as a basic knowledge of MetaStock is highly recommended before trying to learn the MetaStock formula language. Charts of securities that include open, high, low, close, and volume data are recommended to perform the exercises in this document.

Chapter 1
The Basics

All About Power Tools

MetaStock has several advanced features, based on the MetaStock formula language, collectively referred to as power tools. These features allow you to customize your technical analysis. They are accessed through the Tools menu and include the Expert Advisor, Indicator Builder, System Tester, and The Explorer.

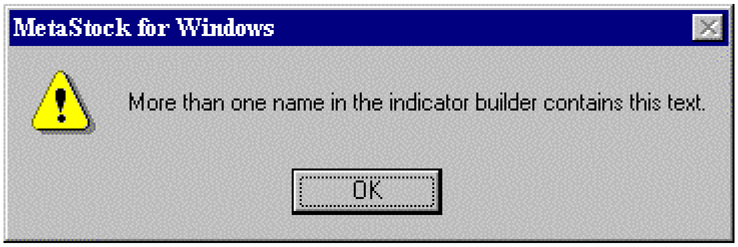
The power tools are extremely powerful tools that greatly enhance the capabilities of MetaStock. They come with several predefined examples, but to really get the most out of them, you need to be able to write your own. That's where this text comes in.

The MetaStock formula language is not a massive collection of symbols you have to remember. Instead, it is a small group of mathematical functions that the MetaStock program uses to calculate a numerical value. We will create many formulas throughout this text. Each one will show, by example, a new function or concept.

To do this, we will use the Indicator builder. This power tool is the easiest to use and is the basis on which the other three are built. In case you've never looked at it before: in MetaStock, click on **Tools | Indicator Builder**.

The window in front of you contains a listing of all the existing custom indicators in your copy of MetaStock. The standard indicators are preset and you cannot alter them. A custom *indicator* is a user-defined formula that can be plotted on a chart. When one of these custom indicators is plotted, MetaStock calculates the value of the formula for every point of data in the chart and draws a line connecting the values. Your first custom indicators will be very simple and progress from there.

The custom indicators you currently have listed will not affect your ability to use this text. The only possible problem that might occur is if this text suggests a name for an indicator and you already have one by that name. When you try to create a formula with a name that has already been used, you will get a message like the following:



More than one name in the indicator builder contains this text.

If you receive this message, just click OK and type in a different name. It doesn't matter what name you choose. What you name it will in no way affect the formula or the calculation of the indicator. However, when naming an indicator, you may find it useful for future reference to have the name reflect the indicator's formula or its intended function.

What's in a Formula

Now that you know where custom indicators are created, it's time to discuss the actual *formulas*. A MetaStock formula is a logical arrangement of one or more functions, operators, and data arrays. Don't be intimidated by these terms. They are actually very simple concepts.

A *function* is a command to perform a specific mathematical calculation. Cosine and moving averages are examples of functions. Others are more complex but they all take one or more values and perform a set calculation.

Operators are basically a subset of functions that can be expressed by a single symbol. They are separate because of a few operations that don't have matching functions. For example, Addition is a standard function but it is usually written with the single + symbol instead of the longer function command.

Data array is simply a fancy term for an ordered grouping of numbers. The closing prices of IBM are a data array. The volume of the NYSE is another data array. Each of these examples is arranged in chronological order by dates in a MetaStock data file. Any such series of numbers can be a Data array.

Now that the terms are defined, examine the formula definition again. A formula is a logical arrangement of one or more functions, operators, and data arrays. Restated, this means the data arrays, operators, and functions express a mathematical calculation for MetaStock to plot on a chart.

Your first Indicator

An indicator can be written using just a data array, for example the closing prices. Formula 1 demonstrates this. Create an indicator with the formula below and name it Closing Prices.

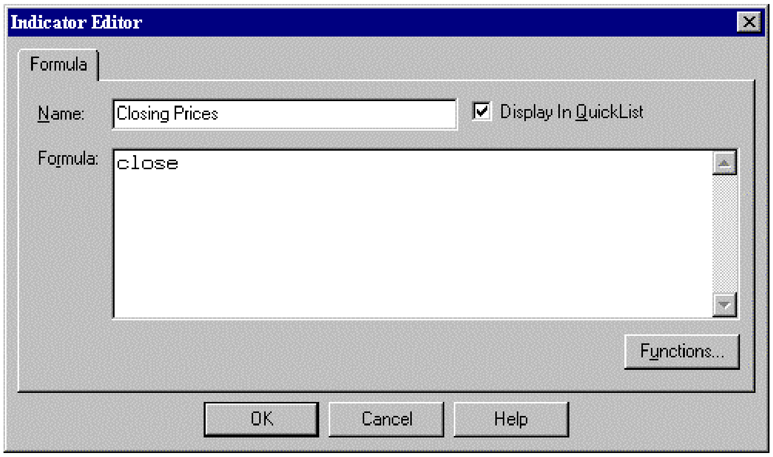
Formula 1:

```
close
```

If you have never created an indicator before, just follow these steps.

1) Choose Indicator Builder from the Tools menu.
2) Click the New button.
3) Type the name of the indicator, in this case: Closing Prices.
4) Click in the large white box titled Formula
5) Type the formula: close.

Your screen should now look like the picture below:



Name: Closing Prices

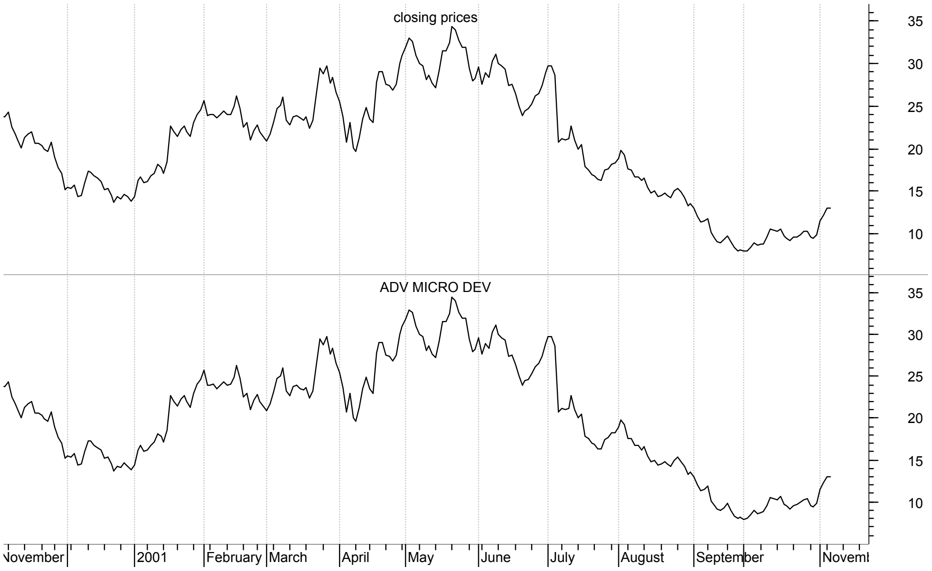
☑ Display In QuickList

Formula:

```
close
```

6) Click the OK button.
7) If you get an error, check the spelling in the formula and make any corrections needed.
8) Close the Indicator Builder by clicking the Close button.

Congratulations, you have just written your first indicator. It will be listed in the Indicator Quicklist after the built-in indicators. This is the place where all the custom indicators are alphabetized. Now plot it on a chart and see what happens. You should get a line that exactly matches the closing prices of the stock you plotted it on.



This indicator is no big deal because you could get the same thing by changing the price style to a line chart. However, you have created an indicator. The volume displayed at the bottom of most charts is an indicator with the following formula:

Formula 2:

```
volume
```

If you wish, create the above formula and plot in a chart with the volume indicator. The only difference is the volume is displayed as a histogram instead of a line. The built-in Open Interest indicator has the formula:

Formula 3:

```
Open Interest
```

If this formula is created and plotted with the built-in Open Interest indicator, the values will exactly match. So you see, your first formula is not useless, just simplistic.

How to do a Rewrite

Occasionally, your formula will not do what you want it to do. If you suspect the formula is incorrect or you want to modify it, then you need to edit the formula. When editing a formula, you can change any part of it, give it a new name, or even completely rewrite it.

To edit your first formula:

1) Open the Indicator Builder
2) Click on the indicator named Closing Price
3) Click the Edit button

You should see the same window you saw when you first created this formula. Notice that the word "close" is capitalized even though you probably typed it in lower case, as shown in the example. This is because MetaStock automatically capitalizes the names of all functions and data arrays. You do not need to worry about capitalization. As long as the spelling is correct, MetaStock will take care of most of the rest. You still need to provide the logic, however.

Here is a shortcut for entering the most common data arrays. The price fields of securities can have their names abbreviated. You could have replaced the word "close" in Formula 1 with the letter "c". Below is a table of the different price data arrays and their abbreviations:

Open	O
High	H
Low	L
Close	C
Volume	V
Open Interest	OI

Either the letter or the full price field name can be used interchangeably. In future formulas, for simplicity, this text will use just the letter. If at any time you are confused by an abbreviation, check the table to see what value is being used.

These shortcuts work because the letters are known by MetaStock as keywords. A *keyword* is a letter or word stored in an internal dictionary by MetaStock. Everything typed in the formula window has to be a number, an operator, or a keyword.

EXCEPTION: Anything typed between curly brackets { }, will be ignored by MetaStock. Such text is called a comment. Comments are used to give instructions and leave notes about the formula. Comments are not used in the formulas written in this text. They are mentioned so you will understand them if seen in other formulas.

Adding all this together

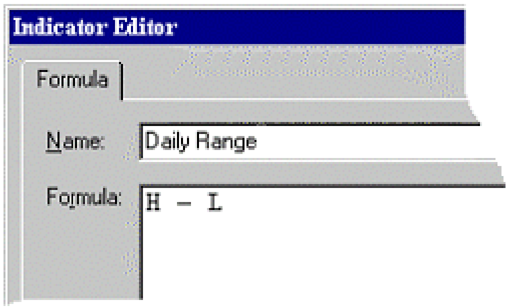
Obviously, if you want your custom indicators to do more than display the data of a price field you need to do something with that data. Perhaps the easiest solution is to ignore it, but then you would not have bought MetaStock to begin with. The second choice is basic math, similar to what you learned in grade school.

Don't worry if math is not your strong suit. This time you get to be the teacher and your student, MetaStock, knows the answers to all of your questions. Your only problem is how to phrase the question.

Exercise 1:

Suppose you wanted to know a range of prices and wanted that plotted as an indicator. Here, the range is defined as the difference between the highest price and lowest price for each period. How would you do it? Write a formula to do this and name it Daily Range. Look at Formula 4 for the answer.

Formula 4:



Name: Daily Range

Formula: H - L

At first, many formulas may seem cryptic. Therefore, all the formulas created in these exercises will include an English "translation." Here is the translation for formula 4.

Formula 4 in English

The value in the high data field minus the value in the low data field.

The MetaStock formula language accepts all the basic arithmetic operations. You can add, subtract, multiply, and divide the data anyway you like. Just write out the equation and MetaStock will solve it for you when you plot the indicator.

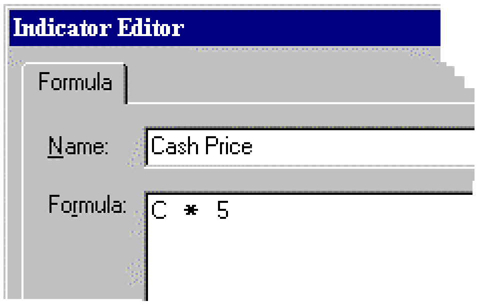
Your keyboard has the plus and minus signs (+ , -), but where are the division and multiplication symbols? The x you used in school gives some error about an unrecognized function and the "divided by" sign just does not exist.

Welcome to the world of computer math. Because computers do not have the desired and familiar symbols on their keyboards, other symbols are used in their place. The multiplication symbol is replaced by an asterisk (*) and the division symbol is replaced by a slash (/).

Exercise 2:

Here's a real challenge for you. You want to see the dollar value of a future contract and your data provider only gives the security's prices in points. Short of getting a new data vendor, what can you do? Well, if you know that one point is equal to five dollars, you can create an indicator to display the equivalent cash price. Write a formula to do this and name it Cash Price. Look at Formula 5 for the answer.

Formula 5:



Name: Cash Price

Formula: C * 5

Formula 5 in English

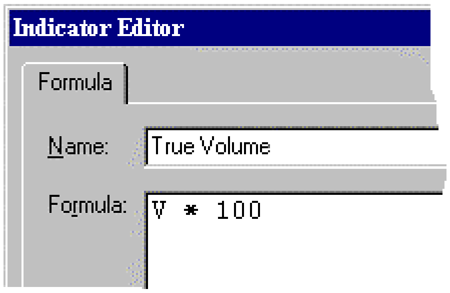
The closing price multiplied by 5

If you are not familiar with the futures market, the last example may not have made much sense to you. In that case, please look at the following.

Exercise 3:

The volume in MetaStock's data files is usually divided by 100 before the value is stored. How can you see the actual volume of a security? Write a formula to do this and name it True Volume. Look at formula 6 for the answer.

Formula 6:



Name: True Volume

Formula: V * 100

Formula 6 in English

Volume multiplied by 100

Who's done first

Now that the math symbols are sorted out, you probably just can’t wait to use them. Well, before you do, you are going to be tested. No studying is required. Just solve the following equation in you head.

Formula 6:

```
1+ 2 * 3 - 4
```

If you got 3, you’re right on top of things. If you didn't, read on. Your computer, and consequently MetaStock, always calculates formulas in a specific way. This procedure is called an “order of precedence.” It's a fancy way of saying “this is how things are done.” In baseball, the runner can't skip some bases or run them backwards. The same is done in computer math. The exact order is Multiplication, Division, Addition, and Subtraction. An easy way to remember this is to think of My Dear Aunt Sally.

Using the order of precedence, examine the above equation again.

```
1+ 2 * 3 - 4
```

The multiplication is done first which changes it to:

```
1+ 6 - 4
```

Since there is no division to be done, the computer adds next giving the result of:

```
7 - 4
```

Subtraction is now done yielding the final result of 3. Knowing the order of precedence will save you from pulling out your hair in frustration over bizarre formula results. If you have several operations of the same type, they are executed in order from left to right. Thus:

```
8 / 2 / 2
```

is the same as:

```
4 / 2
```

which equals 2.

How Parentheses Work

Please examine the test question one more time.

```
1+ 2 * 3 - 4
```

Suppose you wanted to add before you multiplied. This can be done by putting the addition part of the formula in parentheses as follows:

```
( 1+ 2 ) * 3 - 4
```

Parentheses tell the computer to “do this part of the formula first.” This allows you to bypass part of the order of precedence. Inside the parentheses, the order of precedence is still followed exactly. So:

```
( 1+ 2 * 3 ) - 4
```

equals 3 while :

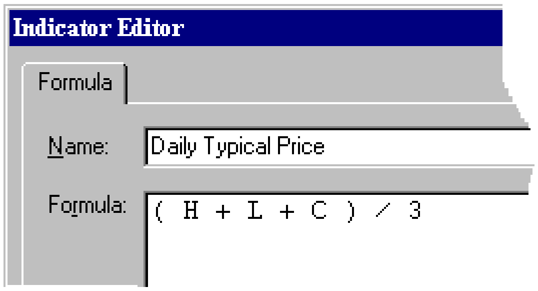
```
( 1+ 2 ) * 3 - 4
```

equals 5.

Exercise 4:

Now put this to some practical use. Suppose you wanted to calculate the typical price, which is normally defined as the average of the High, Low and Closing prices. Write a formula to do this and name it Daily Typical Price. Look at Formula 7 for the answer.

Formula 7:



Name: Daily Typical Price

Formula:
```
( H + L + C ) / 3
```

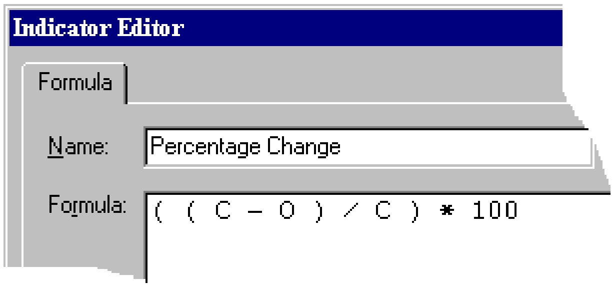
Formula 7 in English

The total of the highest price, the lowest price, and the closing price all divided by 3

Exercise 5:

Suppose you wanted to know the percentage change between the open and close of a stock. Write a formula to do this and name it Percentage Change. Limit the formula to just what has been discussed so far, the different price fields, and mathematical operators. Look at Formula 8 for the answer

Formula 8:



Name: Percentage Change

Formula:
```
( ( C - O ) / C ) * 100
```

Formula 8 in English

The result of the closing price minus the opening price divided by the closing price, all multiplied by 100.

As you can see parentheses can be put inside parentheses. This is called nesting and will be covered in more detail later (See chapter 3).

Adding Functionality

You've read the definition of functions but have not really seen how they are used. All of MetaStock's built-in custom indicators have a function to calculate the same value. This is a shortcut to save you the trouble of writing a long formula MetaStock already knows how to calculate. For example, to write a formula for the MACD, you could type the full mathematical calculation, or you could enter in the following:

Formula 9:

```
MACD( )
```

All function names are usually the name of the indicator they calculate, or an abbreviation thereof. Additionally, all functions have the open and closing parentheses after the name. On some functions, these parentheses are empty, as with the MACD(). Others require additional information, called *parameters*.

Whenever a function is discussed to in this document the parentheses will be empty. If the function is used in a formula, all the parameters will be filled in. This is for ease of reading and space consideration

Formula 10:

```
MOV( C, 14, S )
```

The above formula is for a 14-period simple moving average of the close. Notice the text in the parentheses. This is the additional parameters required for the Mov() function. If you reference the Mov() function in your MetaStock manual or the online help, you will see the following information:

Moving Average

SYNTAX mov(DATA ARRAY, PERIODS, METHOD)

FUNCTION Calculates a PERIODS moving average of DATA ARRAY using METHOD calculation method.

Valid methods are EXPONENTIAL, SIMPLE, TIMESERIES, TRIANGULAR, WEIGHTED, VARIABLE, AND VOLUMEADJUSTED (these can be abbreviated as E, S, T, TRI, W, VAR, and VOL).

If you want to use the Median Price or Typical Price for the data array, simply use the functions (i.e., mp() or typical()).

EXAMPLE The formula "mov(CLOSE, 25, EXPONENTIAL)" returns the value of a 25-period exponential moving average of the closing prices.

This is the standard format for all MetaStock functions. The first line tells you the name of the function. The Syntax section shows the function as it is written and what additional parameters are required by MetaStock. The Function area defines what this function does. It also lists the options for parameters which require specific values. The Example section is just that. It shows the function with the parameters filled in and what the function would plot.

The Syntax section does not explain all the required parameters. Certain values it assumes you already understand. Since this may be your first exposure to these functions, here is a more detailed explanation. The DATA ARRAY has already been discussed. PERIODS is simply asking how many time periods are used in the calculation (i.e., a 14-period moving average uses the number 14 here). METHOD is asking what kind moving average you wish to calculate.

MetaStock also includes functions for various mathematical operations. This includes things like the square root of a number or the cosine of an angle. Other functions are designed to recognize a candlestick pattern or return specific information.

Time Management

When MetaStock is calculating any formula, it moves sequentially through the data and applies the formula for each period individually, (i.e., day, week, month, etc.) It does not look at any other period's data unless specifically told to do so. This brings us to the Reference function. The Ref() function allows you to reference data from periods other than the one currently being examined. Its full description follows:

Reference

SYNTAX ref(DATA ARRAY, PERIODS)

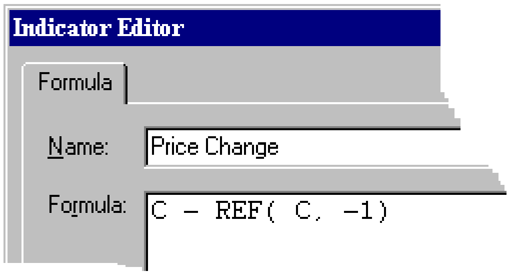
FUNCTION References a previous or subsequent element in a DATA ARRAY. A positive PERIOD references "n" periods in the future; a negative PERIOD references "n" periods ago.

EXAMPLE The formula "ref(CLOSE, -12)" returns the closing price 12 periods ago.

Exercise 6:

Using this function, you can write a formula to calculate the difference in the closing price between one period and the next. It only requires the closing price data array and one Ref() function. Write a formula to do this and name it Price Change. Look at Formula 11 for the answer.

Formula 11:



Name: Price Change

Formula:
```
C - REF( C, -1)
```

Formula 11 in English

The closing price minus the previous period's closing price

Exercise 7:

This formula can also be written using the Rate of Change function, listed below. Examine the function information below and rewrite the formula Price Change. Name the new indicator Changes. Look at Formula 12 for the answer.

Rate of Change

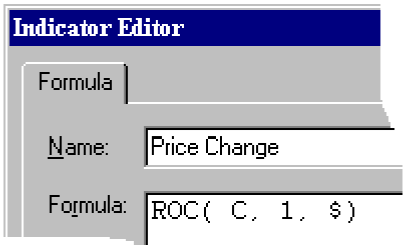
SYNTAX roc(DATA ARRAY, PERIODS, DIFF_METHOD)

FUNCTION Calculates the PERIODS rate-of-change of DATA ARRAY expressed as DIFF_METHOD.

Valid DIFF_METHODs are PERCENT and POINTS (these can be abbreviated as % and $).

EXAMPLE The formula "roc(CLOSE, 12, PERCENT)" returns the 12-period percent rate-of-change of the closing prices.

Formula 12:



Name: Price Change

Formula: ROC(C, 1, $)

Formula 12 in English

The point difference between one period's closing price and the preceding period's closing price

If you want a percentage change, you can replace the dollar sign ($) with the percent sign (%). You can also change the 1 to a different number to see the rate of change over longer periods of time. Try experimenting with what has been covered so far. Also, please take the time to try and answer the review questions. When you feel you are ready, go on to the next chapter.

Review Questions

1. Pick any indicator you like to use in MetaStock. Plot the indicator in a chart. Look up it corresponding function in the manual. Make a custom indicator to duplicate the one you just used and plot it in the same window. The two lines should match exactly.

The above question is one of the best methods to learn the formula language. Some of the indicators may require advanced techniques that have not been covered yet. Come back to those later and keep trying to rewrite them in the formula language. The effort used here will be rewarded with greater confidence and familiarity with the formula language.

2. What does My Dear Aunt Sally stand for?

3. Put parentheses in the following formula to make it equal 3.

```
5 - 4 * 3 / 2 - 1
```

4. When plotting a moving average, you receive several prompts about what type of moving average to plot. One of those prompts is for a vertical shift percentage. This will move the moving average upwards or downwards by the given percentage. A custom indicator can do the same thing by multiplying the Mov() function by a value. Write an indicator that plots a 21-period simple moving average with 20 percent vertical shift upwards.

5. Below are three different formulas. If they were typed as custom indicators, what would be the difference between them (ie.: how would the custom indicators be different when plotted ?)

A	CLOSE
B	C
C	close

Chapter 2
Being Logical

Logic means many things to different people. To avoid confusion, this document will define logic in MetaStock's formulas language as a reasoned decision between two options base on a mathematical comparison. A reasoned decision means that MetaStock is forced to make a choice. The choice is always limited to two options and the decision is based upon comparing something to something else using math.

This means that a logical decision in MetaStock can always be broken down to three parts. The first part is the mathematical comparison. This is followed by the option to use if the comparison is true. The last item is the option to use if the comparison is false. To write this in pseudo-logic:

```
IF( Condition, True result, False result )
```

Does all this sound artificial to you? How are you supposed to relate to this? Here’s a secret: You do this exact same thing every day of your life. Consider the following examples:

You are in your car:
If the gas tank is near empty then buy gas, else wait till later

You are shopping for some item:
If Brand A is cheaper then buy Brand A, else buy Brand B

You are driving and approach an intersection:
If the light is red then stop, else keep going

Logic can be easy if you reduce everything down to a condition and a choice of two results.

Logic in Formulas

In the MetaStock formula language, logic is applied with an If() function. This consists of the same three parts discussed earlier. First define a mathematical condition. Then list the action to take if the condition is true and lastly, the action for a false condition.

If

SYNTAX	if(DATA ARRAY > >= < <= <> = DATA ARRAY, THEN DATA ARRAY, ELSE DATA ARRAY)
FUNCTION	A conditional function that returns the second parameter (THEN) if the conditional expression defined by the first parameter is true; otherwise, the third parameter is returned (ELSE).
EXAMPLE	The formula "if(1<2,3,4)" will always return the value three.

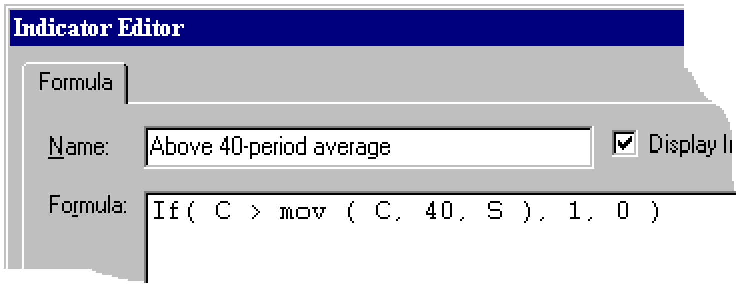
How do you compare two values? The function description shows a confusing jumble of arrows and equal signs. If this doesn't make sense, don't worry about it. The various meanings are listed below.

Symbols	Meaning	Example
>	greater than	2 > 1
>=	greater than or equal to	(2 + 2) >= 4
<	less than	3 < 4
<=	less than or equal to	(3 - 1) <= 2
<>	not equal to	(3 - 1) <> 3
=	equal to	(2 + 2) = 4

Exercise 8:

Suppose you wanted to know when a security was above its 40-period simple moving average. MetaStock can plot a 1 when this is true and a 0 when it's false. What formula would you write? Look at the If() function and the different logic comparisons. Write a formula using the If() function and name it Above 40-period average. Look at Formula 13 for the answer.

Formula 13:



Name: Above 40-period average

Formula:

```
If( C > mov ( C, 40, S ), 1, 0 )
```

Formula 13 in English

If the closing price is greater than a 40 period simple moving average of the closing price, plot a 1, otherwise plot a 0.

Above 40-period Average has the three parts required for an If() function. First it compares the close to the moving average. Then it lists the value to be plotted if the comparison is true. The last number is the value to be plotted if the comparison is false.

Surfing the Binary Wave

Plotting Above 40-period Average displays what some have called a *binary wave*. The term binary applies to a counting system that only uses 0's and 1's. A binary wave, therefore, is a line that moves back and forth between 0 and 1. As you get more proficient you may wish to elaborate on the basic binary wave and it may no longer stop at 1 or 0 but it will still have its roots in the If() function.

This binary wave is very important for other reasons. It is how system tests, explorations, and expert advisors are created. In these other formula-based tools, you give a condition under which an action is performed.

For example, the condition of the close being greater than a moving average could be used in a system test as the signal to enter a long position. It could also be used as the filter of an exploration to tell you all the securities you have that are currently in this condition. You could also use it in an expert advisor as the condition for displaying the bullish trend.

In all these cases, the same condition is used. The only difference is that in these other tools, the If() function is automatically assumed so you can abbreviate the formula to:

Formula 14:

```
C > mov ( C, 40, S )
```

This same abbreviated formula can be written and plotted with the indicator builder. Whenever a condition is written as the formula and no other information follows, MetaStock assumes the formula to be:

```
IF ( condition, 1, 0 )
```

This is because MetaStock, like most computer languages, equates 0 with a false condition. Logically, anything that is not false must be true. For simplicity sake, a 1 is used to plot true conditions. This creates a binary wave similar to the one from the Above 40-period Average formula.

Remember, when you are creating system tests, experts and explorations, the formulas are looking for true or false conditions. MetaStock interprets 0 as false and ANY non-zero number will be read as true.

AND Now for Something Completely Different

Now suppose you had several conditions that you needed to examine. Real life, after all, can get a whole lot more complicated that two conditions. Consider these situations:

You are at a theater's snack bar:

If you have money AND you are hungry then buy a snack, else leave

You want to turn right at an intersection:

If the light is red AND somebody is coming then don't go, else turn right.

The word AND is a keyword in MetaStock. It means both conditions on either side of it must be true or the entire statement is false. This diagrammed in table below

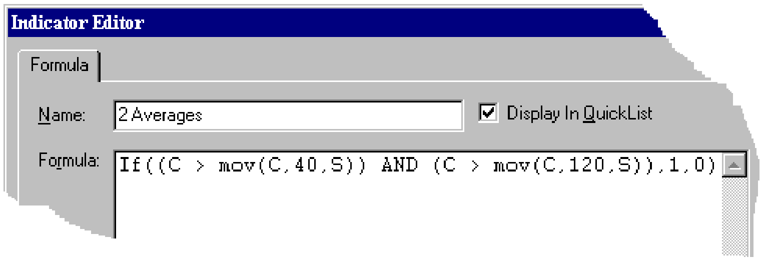
Condition 1		**Condition 2**		**Result**
True	AND	True	=	True
True	AND	False	=	False
False	AND	True	=	False
False	AND	False	=	False

Whenever you are using the AND function, it helps to put parentheses around the two conditions to make sure MetaStock understands what you want compared. These may not be required, but they are a good safety precaution.

Exercise 9:

For your first formula combining the If() and the AND functions, say you wanted to know when the close was above both its 40-period moving average and its 120-period moving average. Once again, have MetaStock plot a 1 for this condition being true and a 0 for false conditions. You can use the Above 40-period Average as a starting point. Write a formula to do this and name it 2 Averages. Remember to separate the two conditions with parentheses. Look at Formula 15 for the answer.

Formula 15:



Name: 2 Averages

☑ Display In QuickList

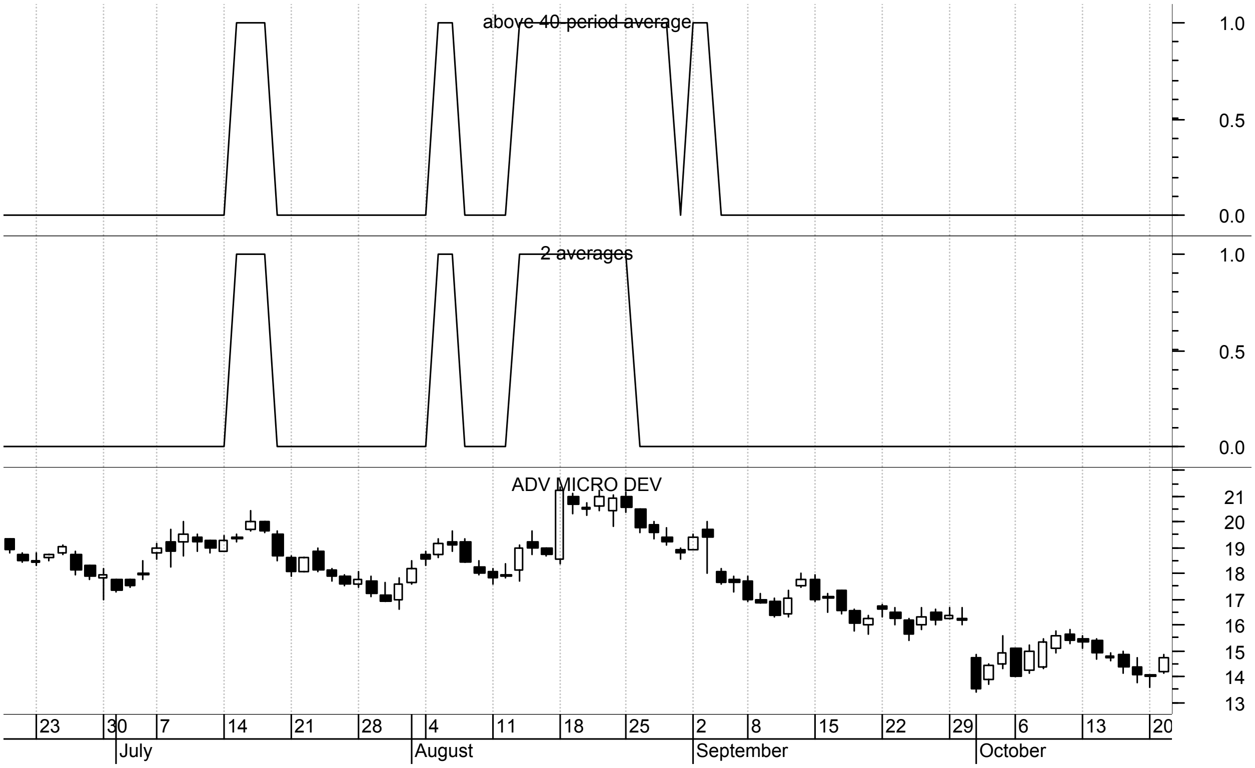
Formula:

```
If((C > mov(C,40,S)) AND (C > mov(C,120,S)),1,0)
```

Formula 15 in English

If the closing price is greater than a 40 period simple moving average of the close and the closing price is greater than a 120 period simple moving average of the close, plot a 1, otherwise plot a 0.

Only if the close is above both moving averages will the above formula plot a 1. At all other times, it will plot a 0. You should notice that the binary wave of Above 40-period Average has wider plateaus than 2 Averages. See the figure below.



This is because the more specific you make your conditions, the less often they occur. This may seem obvious but sometimes formulas have such detailed conditions that they never get a true result; what is being sought is theoretically possible, but practically never seen.

When making complex If() functions, try plotting each condition as a separate formula and then see where all the separate parts return true results. This way, you can make sure each part is accurate and make changes as needed.

OR You Can Have It All

Some times your decisions are based on one of several conditions. If any of the conditions are true, you will take a specific action. Consider the following real world examples.

<u>You look at the weather outside:</u>

If it is raining OR it is snowing then take an umbrella, else leave the umbrella.

<u>You are at home:</u>

If it is time for bed OR you are tired then go to bed, else stay up.

Where the AND function limited the condition, OR returns a true result if either of the conditions is true. The following table diagrams this for you

Condition 1		Condition 2		Result
True	OR	True	=	True
True	OR	False	=	True
False	OR	True	=	True
False	OR	False	=	False

Just like the AND function, I strongly recommend using parentheses to group the conditions of OR functions. They may not be needed, but they help you to be sure you are asking what you think you are asking.

Exercise 10:

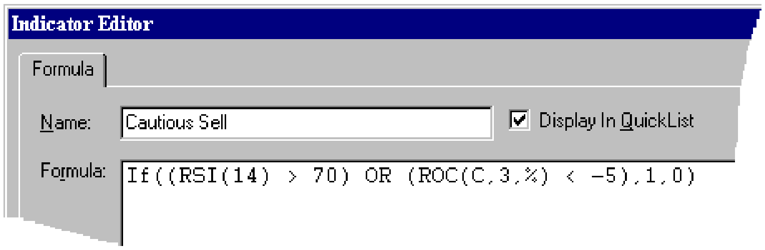
To see the OR function in action, consider the following situation. The Relative Strength Index (RSI) can be used in several ways, including displaying overbought and oversold conditions. An RSI value of over 70 is considered overbought and a sell signal. If you were a really cautious investor, you might want to know when either a 14-period RSI rises above 70 or the stock loses five percent, or more, of its value in three periods.

Below is the RSI() function. You can use either the Ref() or the RoC() functions for the second condition. Look these functions over and then write a formula that will help such cautious investing and name it Cautious Sell. Have the formula plot a 1 when either condition occurs and a 0 other wise. Look at formula 16 for the answer.

Relative Strength Index (RSI)

SYNTAX	rsi(PERIODS)
FUNCTION	Calculates the predefined RSI indicator.
EXAMPLE	rsi(14)

Formula 16:



Name: Cautious Sell

☑ Display In QuickList

Formula:

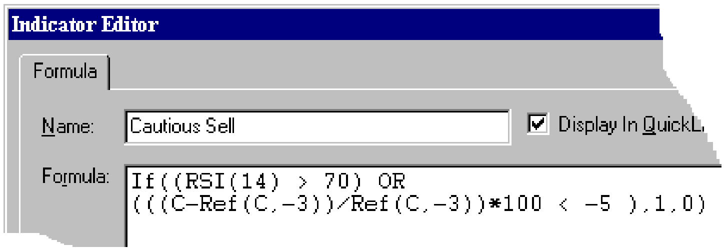
```
If((RSI(14) > 70) OR (ROC(C,3,%) < -5),1,0)
```

Formula 16 in English

If the 14-period Relative Strength Index is above 70 or the close has dropped by 5 percent or more over the last 3 periods, plot a 1, otherwise plot a 0.

This formula uses the RoC() function. It is the easier method. However, if you wanted to use Ref() instead, look at the next formula.

Formula 17:



Name: Cautious Sell

☑ Display In QuickL

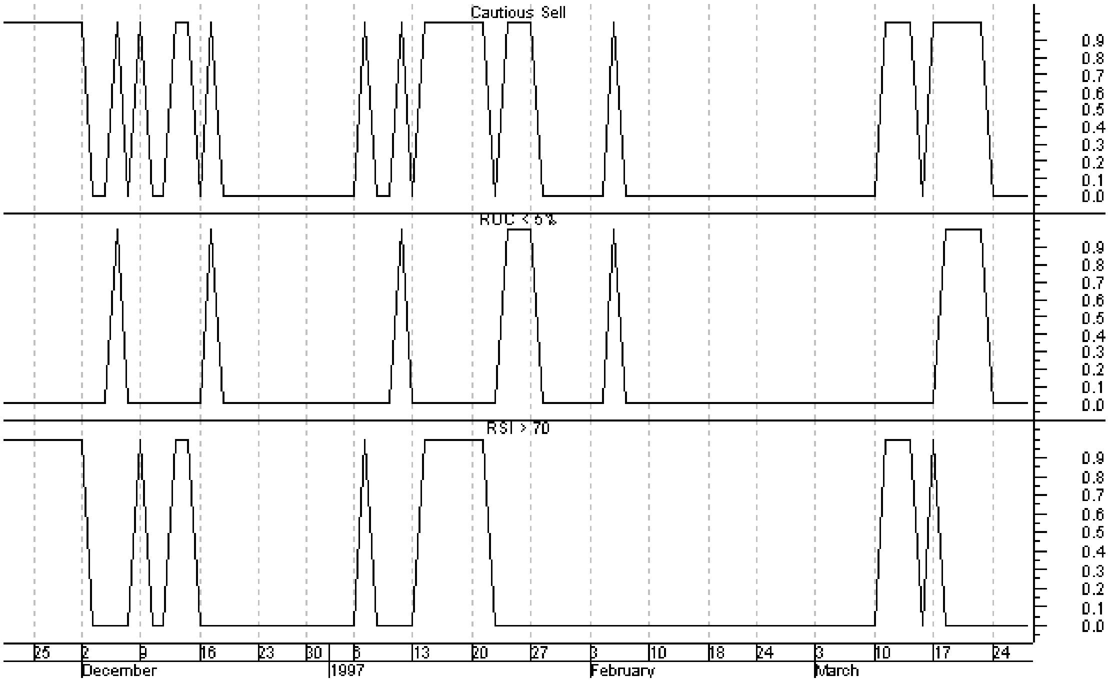
Formula:

```
If((RSI(14) > 70) OR
(((C-Ref(C,-3))/Ref(C,-3))*100 < -5 ),1,0)
```

Formula 17 in English

If the 14 period Relative Strength Index is above 70 or; the result of the close minus the close 3 periods ago divided by the close 3 periods ago, all multiplied by 100 is less than -5; then plot a 1, else plot a 0

Both these plot exactly the same line.



Formula 16 is a lot easier to read and write. This is why it helps to be familiar with the different functions. You can reproduce any built in function with a custom formula but why not use your time and effort for other tasks. After all, wouldn't you rather write

```
ROC( C, 3, % )
```

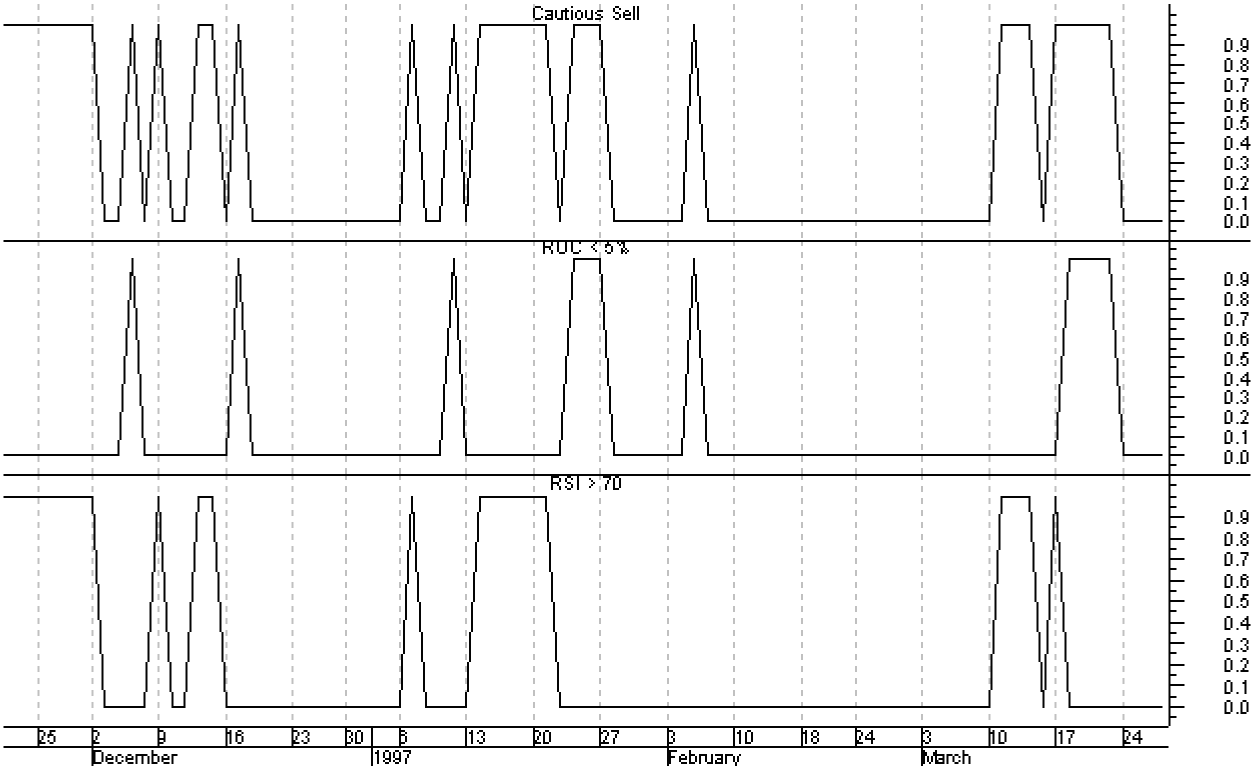
instead of

```
((C-Ref(C,-3))/Ref(C,-3))*100
```

These two formulas also prove that there is usually more than one way to write any formula. Some may be more elegant but if it does what you want, it's not wrong!

Back to the OR

While AND requires both conditions to be true and narrows the number of occurrences, OR requires only one true condition. This will, therefore, increase the frequency of true results. The following chart demonstrates this.



When you use OR, there is always the danger of a final result so general that it is always true. Again, try plotting all conditions separately before combining them. This will help you to visually see what the approximate results will be before the final formula is made.

This AND / OR That

The AND and OR functions can be combined in a single If() function. The resulting condition part of the IF may look something like this

```
((condition 1) OR (condition 2)) AND ((condition 3) OR
(condition 4))
```

You could replace any of the ANDs and ORs with the other function. You could also make the formula more complex by replacing any of the conditions with another set of conditions combined by an AND or an OR.

When doing something like this, testing each part separately can save you tremendous amounts of time correcting problems later. For example, if you made a formula similar to the four-condition beast shown above, you should first create the four formulas below:

```
Condition 1
```

```
Condition 2
```

```
Condition 3
```

```
Condition 4
```

After confirming that each of these formulas plot what you want them to, they can be combined to form the next layer of commands:

```
( Condition 1 ) OR ( Condition 2 )
```

```
( Condition 3 ) OR ( Condition 4 )
```

These two formulas will show the combined values of the paired conditions. After making sure they are the results you want the final version of the formula should be written.

```
((condition 1) OR (condition 2)) AND ((condition 3) OR
(condition 4 ))
```

Obviously, you should only do this when first creating the formula. Once the formula has been created and you know it is performing as expected, just plot the completed formula. As you gain proficiency in writing formulas, you will be able to skip writing the separate parts of the If() function and just write the final version. However, should you ever need to see exactly what is being done at some point in a formula, just break it down to its components and see how it looks.

An alternative to breaking the formula down into its separate parts, is using comments to mark parts of the formula to be ignored. For example:

```
{ ((condition 1) OR ( } condition 2 { )) AND ((condition 3)
OR (condition 4 )) }
```

is the same as:

```
condition 2
```

By putting the { } brackets around a section of text, you are telling MetaStock to ignore that part of the formula.

Try writing some If() functions with and without the AND and OR functions. The review questions will also help you gauge your understanding of this chapter. Please be sure you have a grasp on these concepts as they are used far more than you might initially think. When you feel comfortable with these commands, go on to the next chapter.

Review Questions

1. Today, stock XYZ has a closing price of 30, a high of 35, a low of 25, and an opening price of 28. Which of the following conditions will MetaStock evaluate to be true when used in an If() function and plotted on this security?

A	Close > Open
B	Close < 35
C	Open + Close

2. Using the same data from question one, is the following condition true or false?

```
( Close < 35 ) AND ( Open > 28 )
```

3. Are the following statements true or false:

1) (true OR false) AND true
2) (false AND true) OR false
3) (false AND false) OR (true OR false)
4) (true OR (false AND false)) AND (true OR true)

4. What are the three components of an If() function?

Chapter 3
Building a Nest

You have seen what to do when there are two possible outcomes, but what happens if you have three or more choices to decide between? The If() function can only examine a condition and return one of two results. However, the result returned is a data array. It does not have to be a single number. It could be a function, an equation, or another If(). Consider the following format:

```
IF ( condition 1 , Result A, IF ( condition 2, Result B,
Result C) )
```

If condition 1 is true, Result A is used. However, if condition 1 is false, condition 2 is then examined and a choice made between Result B and Result C.

Put in everyday terms, suppose you are at a snack bar in a movie theater. You would really like the large chocolate bar. Unfortunately, it costs five dollars and you are not sure you have that much. Before you look in your wallet, you also notice the theater sells jelly beans and roasted peanuts. Both of these are the same price and only cost three dollars. Your decision on what to buy can be written as:

```
IF ( I have five dollars or more, buy chocolate bar, IF ( I
want something crunchy, buy peanuts, buy jelly beans ) )
```

This format of putting one If() function inside another If() is called nesting. Any place a formula or function calls for a data array, you can place an If(). You can also put an If() function inside an If()that is already inside another If(). This can get kind of confusing. The most common result is to forget how many If()s have been used and not put enough closing parentheses. MetaStock will give you a warning that it is missing one, but will not know where it is suppose to go. Try using the following format:

```
If ( condition 1,
    result A,
    If ( condition 2,
        result B,
        result C
    ),
)
```

This is a method of organizing If()s that is borrowed from several computer programming languages. The If() and its condition are given on one line. The two possible results are indented by several spaces on the lines below. The closing parenthesis is put directly under the If() on a line below the last result. When a result is another If(), the entire function is written the same but indented several spaces further.

Obviously you do not need to do this with simple If() formulas, but the example above is given below without any of the extra spaces.

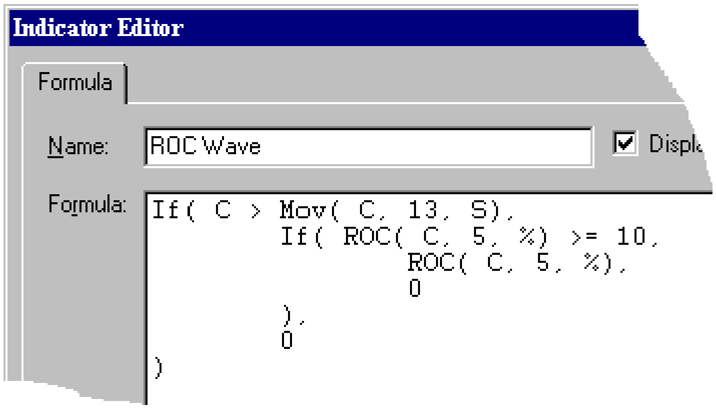
```
If ( condition 1, result A, If ( condition 2, result B,
result C ) )
```

Which of these would you rather work with? Before you answer, remember that an actual formula will have functions and mathematical comparisons instead of the words "condition" and "result." In a MetaStock formula; spaces and extra lines do not affect how a formula is calculated. They do, however, make your formula easier to read.

Exercise 11:

Before you start to think all of this discussion on nested If()s is confusing and of no practical use, suppose you wanted a formula to let you know when a security's price rises ten percent or more in 5 periods, but only if the security is above its 13-period simple moving average. To make things more interesting, instead of plotting a 1 and a 0, plot the actual rate of change when this condition is true and a 0 when it isn't. Write a formula to do this and name it ROC Wave. Look at formula 18 for the answer.

Formula 18:



ROC Wave

```
If( C > Mov( C, 13, S),
          If( ROC( C, 5, %) >= 10,
                    ROC( C, 5, %),
                    0
          ),
          0
)
```

Formula 18 in English

If the close is greater than a 13-period simple moving average, and if the 5-period rate-of-change is greater than or equal to 10, then plot the 5-period rate-of-change, otherwise plot a 0

Formula 18 is written in the expanded format referred to earlier. It could also be written as:

```
If( C > Mov( C, 13, S), If( ROC( C, 5, %) >= 10, ROC( C, 5, %), 0),0)
```

When displaying formulas with nested If()s, this document will use the expanded version for ease of reading. When writing your own formulas, use whichever version you prefer. If you have a hard time finding and correcting some errors in your formula, try expanding it. The only place MetaStock does not allow you to add a space or an extra line is between a function's name and its opening parenthesis. Everywhere else, add as much space as you need to make the formula easy for you to read.

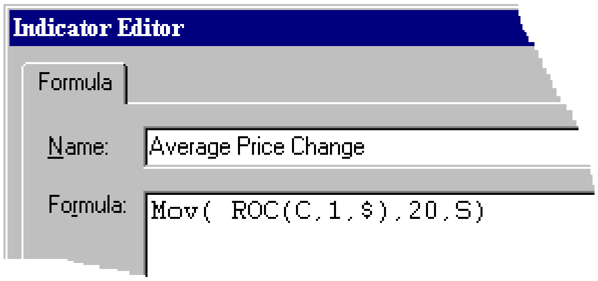
Functions get into the nest

Just like If()s can be nested, other functions can also be nested. Most functions require a data array as part of their parameters. Instead of putting one of the price fields or a number, you can insert another function. Consider the following.

Exercise 12:

A stock you are following is changing prices erratically. It will gain a few points one day and drop some the next. The exact amount of each day's move varies. Before investing your money, you would like to get a feel for how much it is likely to move in one day. This might be accomplished by a moving average of a 1-day price change. Write a formula to do this and name it Average Price Change. Use a 20-period simple moving average and points for the rate of change. Look at Formula 19 for the answer.

Formula 19:



Name: Average Price Change

Formula:
```
Mov( ROC(C,1,$),20,S)
```

Formula 19 in English

A 20-period simple moving average of a 1 day rate-of-change

The ROC() function is simply inserted in the Mov() function where a data array was asked for. This can be applied to any function that accepts a data array.

To understand how this is possible, go back to the definition of a data array. It was defined as a series of values. Examples were given of the different price fields. A data array can also be defined as a *constant*. When a data array location is given a fixed number, all calculations of the formula use that number. The number does not change and so it is called a constant. You have already used constants. While writing the If()s in chapter 2, the results were either 1 or 0. These numbers are constants.

A data array is also the result of any function's calculation. A moving average returns a series of numbers that is plotted by MetaStock. Those numbers are a data array. The same is true for the Rate of Change and every other function in MetaStock's formula language. Each one returns a numerical value for each period it is calculated. Those values make an array of data the can be plotted in a chart or used as part of a larger calculation. Here is another exercise that demonstrates this concept:

Exercise 13:

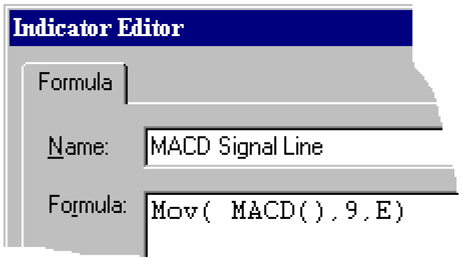
Back in chapter 1 when functions were first introduced, the MACD() function was mentioned. Here is the full command:

MACD

SYNTAX	macd()
FUNCTION	Calculates the predefined MACD indicator.
EXAMPLE	The formula "macd()" returns the value of the MACD indicator (i.e., the solid line).

As you can see, this is just the solid line part of the MACD indicator. The dotted signal line that appears when this indicator is plotted in MetaStock is not given by this function. That requires a separate formula. The dotted line is actually a 9-period exponential moving average of the MACD. How would you calculate this line in an indicator? Write a formula to do this and name it MACD Signal Line. Look at formula 20 for the answer.

Formula 20:



Name: MACD Signal Line

Formula: Mov(MACD(),9,E)

Formula 20 in English

A 9-period exponential moving average of the MACD

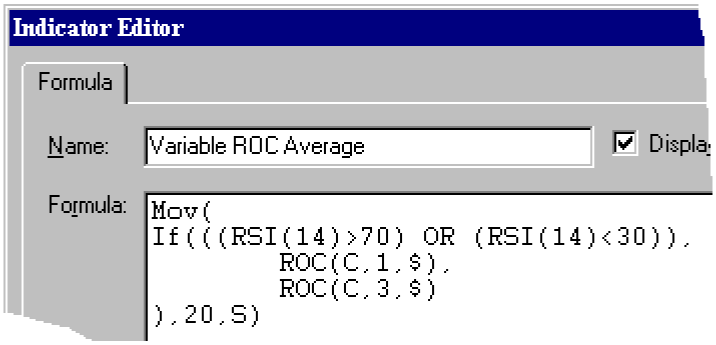
Functional IFs

Just like If()s can use other If()s and functions for their data arrays, functions can use other functions and If()s. An If() returns a data array of values. The only difference is that the If()'s array is dependent on a conditional clause. The resulting value can be used any place a function calls for a data array.

Exercise 14:

Consider formula 19, Average Price Change. Suppose you wanted to vary the ROC() function based on the RSI. When the 14-period RSI is getting near its overbought and oversold regions, the rate of price change might be far more important than when the RSI is in its mid-ranges. Taking such a premise as truth, you might wish to use a 1-period rate of change in the overbought and oversold ranges and a 3 period rate of change elsewhere. Write a formula to do this and name it Variable ROC Average. Use the same 20-period moving average and treat values above 70 and below 30 as the overbought and oversold ranges. Look at Formula 21 for the answer.

Formula 21:



Name: Variable ROC Average

Formula:

```
Mov(
If(((RSI(14)>70) OR (RSI(14)<30)),
        ROC(C,1,$),
        ROC(C,3,$)
),20,S)
```

Formula 21 in English

A 20-period simple moving average of the function: if the 14-period RSI is greater than 70 or the 14-period RSI is less than 30, plot a 1-period rate of change, otherwise plot a 3-period rate of change

The If() condition here looks for any RSI value in the overbought or oversold range. This formula could have been written differently. The If() could have been turned around to read:

```
If (((RSI(14)<=70) AND ( RSI(14)>=30)),
    ROC(C, 3, $),
    ROC(C, 1, $)
)
```

Now the If() looks for no overbought and oversold ranges. The result is the same even if the values have been switched. Notice however, that the equals sign was added to the second version of the If(). The conditions of the formula said that values above 70 and below 30 were overbought and oversold. This means that 70 and 30 are not themselves overbought or oversold. Thus the second version had to include the values of 70 and 30 as possible true conditions.

To put this in a real life condition, picture yourself walking down a street around Christmas time. You know you still need to get one more present and it will cost you exactly five dollars, including tax. On the corner ahead of you is a Santa ringing a bell for donations to the Salvation Army. Being a charitable soul you want to donate some money, but you must save the five dollars. Written as an If() statement, this situation could read:

```
IF ( money > $5, donate some, do not donate)
```

or it could be written:

```
IF ( money <= $5, do not donate, donate some)
```

Both are correct. They just look at the problem from different directions.

I can't believe it ate the whole thing

You have seen how functions and If()s can be nested but nesting can also include embedding a whole indicator. Back in chapter 1, you created the indicator Daily Range. This indicator can be referenced in other formulas through the Fml() function

Formula Call

SYNTAX fml("FORMULA_NAME")

FUNCTION Calculates the value of another formula. The formula can be referenced using the FORMULA_NAME in quotes.

When referencing a formula's name, the name must be contained in quotation marks (e.g., fml("Secret A")).

If you change a formula's name, you must also change any fml() calls that reference that formula.

EXAMPLE The formula "fml("Secret A") * fml("MyMACD")" calculates the value of the formula named "Secret A" multiplied by "MyMACD."

When using this function, it is absolutely necessary to spell the name of the indicator the same as it is spelled in the indicator builder. MetaStock is extremely picky about this and even an extra space can cause problems. As a result, it is highly recommended to use the Functions button in the indicator builder when inserting a Fml() function. If you have never used this feature before, please do the following steps to see how it is done.

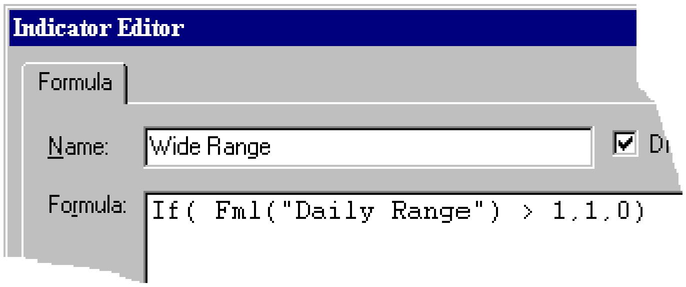
1) Go to the indicator builder and start a new indicator
2) After typing in the name, click in the large formula window
3) With the cursor blinking in the formula window, the Functions button in the bottom right corner is enabled
4) Click the Functions button
5) On the left hand side, click on Custom indicators
6) The right hand side now contains a list of all your custom indicators. Double-click on one to insert it into the new formula

The Paste Functions dialog should have closed and the selected function is now included in an Fml() function in the formula window.

Exercise 15:

Application of this is as easy as any other indicator, apart from all the trouble of entering the function. Assume you wanted an indicator to tell you when the daily range grew wider than 1. Write a formula to do this and name it Wide Range. Use an If() and the Daily Range indicator to write such a formula. Look in formula 22 for the answer.

<u>Formula 22:</u>



Name: Wide Range

Formula:

```
If( Fml("Daily Range") > 1,1,0)
```

<u>Formula 22 in English</u>

If the result of the formula “Daily Range” is greater than 1, plot a 1, otherwise plot a 0

Use Them Anywhere

The Fml() function is just like any other function and returns a data array. This value can be used in any other function that accepts a data array. The Fml() can be combined in If()s as shown above or referenced in other functions.

Using this function, you can create what programmers call subroutines. Subroutines are part of a larger formula that is separated from the rest. This might be done if the overall formula is extremely long or complex. A subroutine is also used if a section of a formula uses the same mathematical expressions several times.

In MetaStock, each subroutine, or sub-formula, is written as a separate indicator. These are then combined into a single indicator which functions as if the entire formula had been written in it instead of multiple sub formulas.

The Fml() function is also used in the other formula based tools of MetaStock. By calling a custom indicator, you do not have to retype the formula each time you wish to use it. This can save a lot of time and allows you to change the formula in one place and not have to chase down all the other places where you used it to keep things consistent.

Don't go in Circles

A Fml() function allows your formula to reference another formula. True to the nesting philosophy, the formula can reference another formula. This can continue building a deeper and deeper nest. However, any formula in this nest cannot reference an earlier formula. Thus if formula A referenced formula B; formula B cannot reference formula A.

The reason this cannot be done is dictated by how MetaStock calculates the formulas. Using the formulas A and B mentioned above, here is how MetaStock would attempt to evaluate them.

When you plot formula A in a chart, MetaStock looks at the formula and begins to run all he calculations. When it gets to the reference to formula B, it stops formula A's calculations and starts calculating formula B. If it then read a reference to formula A, it stops calculating formula B and starts calculation formula A from the beginning.

As you can see, MetaStock is now going in circles. In traditional programming languages, this would be called and endless loop. MetaStock, however, can detect when such a loop starts and will stop calculating the indicator. It will then display an error that it has detected a circular reference. The only way to resolve this is to write the formula to not reference itself.

Another drawback to the Fml() function is that sub-formulas take longer to calculate. Reading the additional formulas as they are called slows down MetaStock's calculations. The more sub-formulas referenced the greater the slow down. Please also be aware that faster computers with large amounts of memory may not be noticeably affected.

Try playing with different ways to nest functions. Go through the review questions and re-read any parts of this chapter you did not understand. Nesting functions and If()s is a major part of more advanced MetaStock formulas. From this point on, nesting functions and / or If()s will be required. If you are ever confused by any of this, please refer back to this chapter.

Review Questions

1. Which of the following can be put in a formula where a data array is required:
    a) MACD()
    b) C
    c) any If() function
    d) 10

2. What is wrong with the following custom indicators:

formula 1

```
Mov( C, 9, E) + Fml(“formula 2”)
```

formula 2

```
(H - L) * Fml(“formula 1”)
```

3. True or False, there is no reason not to nest formulas using the Fml() function.

4. What is the recommended way of entering formula names in the Fml() function?

5. True or False, any function can be nested in any other function.

Chapter 4
Trick or Treat

You've seen how the different functions are written and how to nest them to create a formula. This covers half the difficulty in writing formulas. The next fifty percent is deciding what you want the formula to tell you. In other words, you have to be able to express what you want the formula to do.

When you sit down and to write a formula, try to express it in the simplest terms possible. If you're not sure how to write it mathematically, write it in English. If you still do not know how to express it as a formula, try writing it as if you were explaining it to someone who just started looking at technical analysis today.

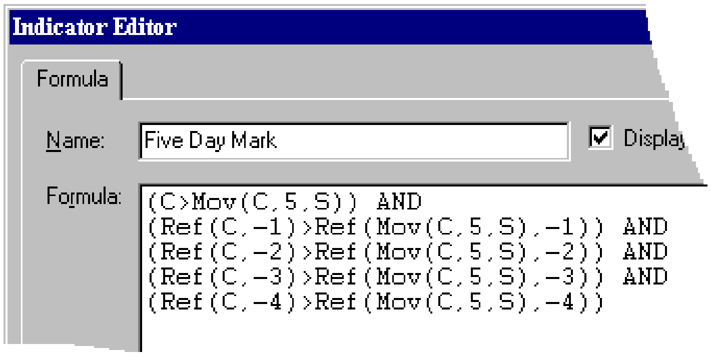
After an idea has been written out, the formula should be easier to create. It may, however, be very messy. Messy in this case refers to a formula being long, repetitive, and/or involved. This chapter is devoted to showing you some tricks and shortcuts that just might save you many hours of work.

Summing Things UP

Exercise 16:

Suppose testing of a security revealed a good buy condition every time its closing price is above its 5-period simple moving average for five consecutive periods. How would you mark these places on a chart with an indicator? Write a formula to do this and name it Five Day Mark. Hint: using just the functions discussed so far, you will need to nest functions with the Ref() function. Look at Formula 23 for the answer.

Formula 23:



Name: Five Day Mark ☑

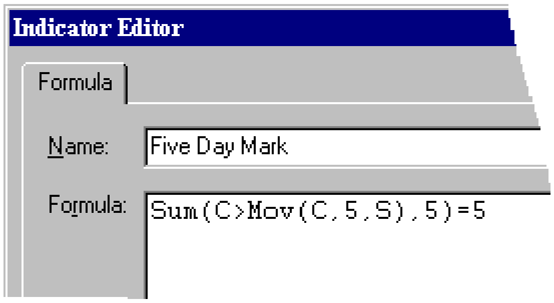
Formula:
```
(C>Mov(C,5,S)) AND
(Ref(C,-1)>Ref(Mov(C,5,S),-1)) AND
(Ref(C,-2)>Ref(Mov(C,5,S),-2)) AND
(Ref(C,-3)>Ref(Mov(C,5,S),-3)) AND
(Ref(C,-4)>Ref(Mov(C,5,S),-4))
```

Formula 23 in English

If the Close is above a 5-period simple moving average of the close; and the previous close was above the previous 5-period simple moving average of the close; and the close 2 periods ago was above the 5-perod simple moving average of the close 2 periods ago; and the close 3 periods ago was above the 5-period simple moving average of the close 3 periods ago; and the close 4 periods ago was above the 5-period simple moving average of the close 4 periods ago; then plot a 1, otherwise plot a 0

As was stated earlier, this is the long way to write this type of formula. The If() is implied in this formula and each period is examined individually to see if it is above it corresponding moving average. If you were to require a security to remain above its moving average for a longer duration, this formula could quickly get out of control. Formula 24 is much shorter.

Formula 24:



Name: Five Day Mark

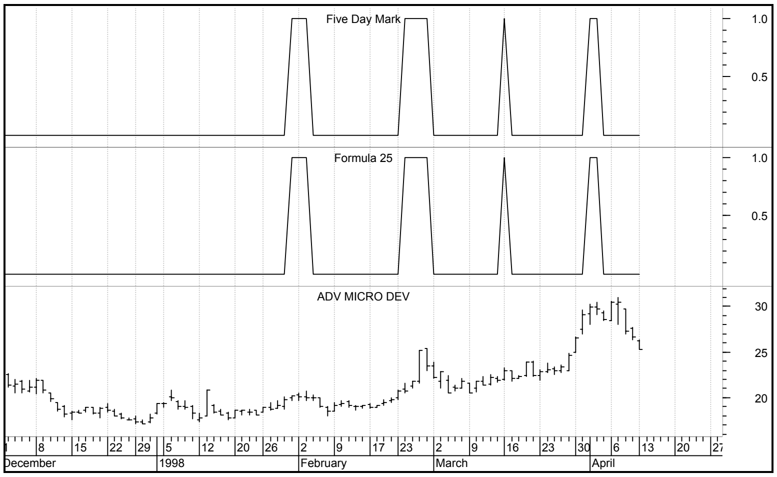
Formula:
```
Sum(C>Mov(C,5,S),5)=5
```

Formula 24 in English

The formula “when the close is greater than the 5-period simple moving average of the close, plot a 1 else plot a 0”; is calculated for the past five days and the results added together. If the result equals 5, plot a 1, else plot a 0.

The Sum() function adds values of a data array for a specified number of periods. In this case, it adds the last five results of "C>Mov(C,5,S).”. This condition returns either a 1 or a 0. So if it is true for five straight days, the result would be 1+1+1+1+1=5. Again, the condition contains an implied If() function. In the future, if a formula is simply plotting whether or not an event occurred, such as this one is, this document will omit the implied If(). Please refer to Chapter 2 if you have any difficulty understanding the If() function or how it is used.

As you can see from the following chart, both of these formulas plot exactly the same way.



If you are curious, here is the full text of the Sum() function.

Summation

SYNTAX sum(DATA ARRAY, PERIODS)

FUNCTION Calculates a cumulative sum of the DATA ARRAY for the specified number of lookback PERIODs (including today).

EXAMPLE The formula "sum(CLOSE, 12)" returns the sum of the preceding 12 closing prices. A 12-period simple moving average could be written "sum(C,12) / 12."

Summation is a very handy function when writing MetaStock formulas. Any time you need to add consecutive values in a data array, see if the Sum() function can help.

Exercise 17:

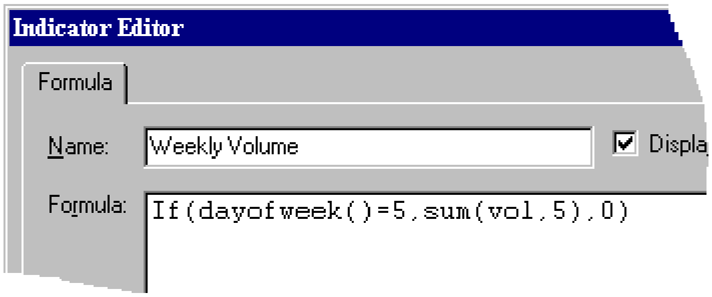
Suppose you wanted to know the total volume for a week while still looking at a daily chart. How would you create such an indicator using the Sum() function. To make it easier, on any day but Friday, only plot a 0. Write a formula to do this and name it Weekly Volume. Hint: you will need the DayofWeek() function shown below. Look at Formula 25 for the answer.

Day Of Week

SYNTAX dayofweek()

FUNCTION Plots the day of the week. 1=Monday, 2=Tuesday, 3=Wednesday, 4=Thursday, 5=Friday, 6=Saturday, 7=Sunday.

Formula 25:



Name: Weekly Volume

Formula:

```
If(dayofweek()=5,sum(vol,5),0)
```

Formula 25 in English

If the day of the week is Friday, plot the sum of the last five periods volume, else plot a 0

The Cumulative function is related to Summation but where Sum() allows you to limit the addition to a specific amount of data, Cum() automatically counts from the first period loaded. There is no way to make it use less data. Cum() has its uses, but do not confuse it with Sum() or your formula won't work.

Staying Alert

The Sum() function works well when you want to count how many times something occurs or total the recent values of a data array. But what do you do when you want to see if a specific event has occurred with in a specified time frame? You are not concerned with exactly when it occurred or even how many times. You just want to know if it occurred. True, you could use the Sum() function by writing something like this:

```
sum( event, number of periods) >0
```

There is a better way, however. Take a look at the alert function below.

Alert

SYNTAX alert(EXPRESSION, PERIODS)

FUNCTION Extends a "true" result of EXPRESSION for the specified number of periods. This true result is held true over the number of periods specified even if a "false" result is generated.

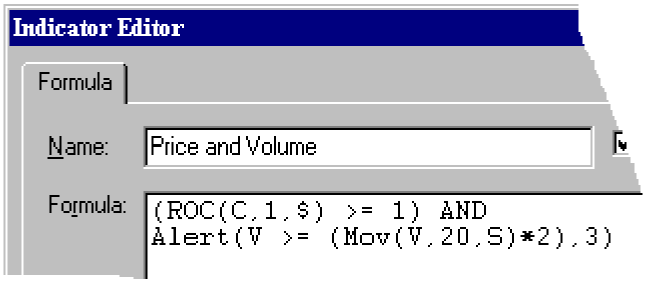
EXAMPLE alert(cross(rsi(14),70),5)

With this function, you save a little space, plus the formula becomes a little less cryptic to other people should you share your ideas with them.

Exercise 18:

Suppose you wanted to know when the price of a security dropped 1 or more whole points in one day. However, you also know that this security is prone to large moves and usually corrects itself. Therefore, you only want to know of the price drops if the volume traded is larger than normal. Write a formula to do this and name it Price and Volume. Use a 20-period simple moving average for the volume and look for trading twice this average within the last three periods of the price drop. Look at Formula 26 for the answer.

Formula 26:



Name: Price and Volume

Formula:

```
(ROC(C,1,$) >= 1) AND
Alert(V >= (Mov(V,20,S)*2),3)
```

Formula 26 in English

If the 1-period rate of change is greater than or equal to 1 and within the last 3 periods the volume has been greater than or equal to twice a 20-period simple moving average of the volume, plot a 1, else plot a 0.

The CROSSing Guard

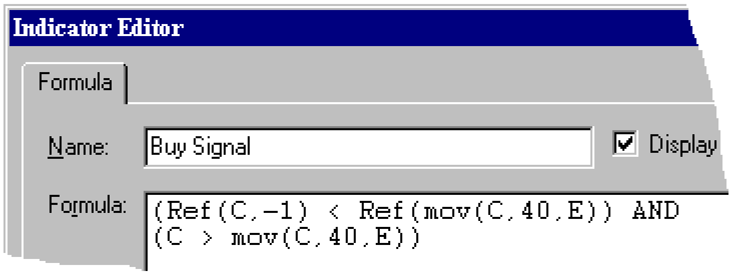
A common buy and sell signal is one line crossing another. The close crossing a moving average or the MACD crossing its signal line can be either a buy or a sell signal depending on which way it crosses. If the RSI passes 70, you might want to examine that security.

All of these market signals occur the day the lines cross. If you open a chart and see the security crossed above its moving average last week and is close to passing back below it now, you missed the buy in point. While that security might soon be a good short sell candidate, the time is not yet right. Timing is everything, correct?

Exercise 19:

You can use the If() function to see when a line crosses another. It would only require you to check that one line was below the other yesterday and above it today. How would you do this to see when a security's closing price rises above its 40-period exponential moving average. Write a formula to do this and name it Buy Signal. Look at Formula 27 for the answer.

Formula 27:



Name: Buy Signal ☑ Display

Formula:

```
(Ref(C,-1) < Ref(mov(C,40,E)) AND
(C > mov(C,40,E))
```

Formula 27 in English

If last period's close was less than last period's 40-period exponential moving average of the close and this period's close is greater than this period's 40-period exponential moving average of the close, plot a 1, else plot a 0.

That seems like a lot of trouble to go through for such a commonly watched event. Enter the cross function.

Cross

SYNTAX cross(DATA ARRAY 1, DATA ARRAY 2)

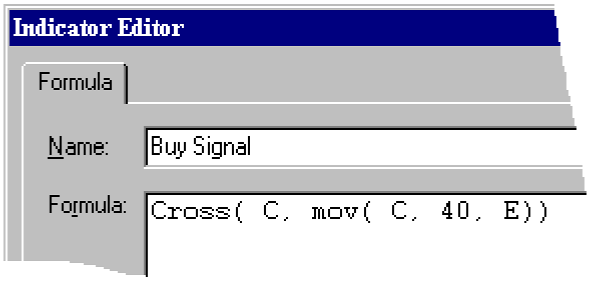
FUNCTION Plots a "+1" on the day that DATA ARRAY 1 crosses above DATA ARRAY 2. Otherwise, "0" is plotted.

If you want to know when DATA ARRAY 1 crosses below DATA ARRAY 2, use the formula "cross(DATA ARRAY 2, DATA ARRAY 1)"

EXAMPLE cross(close, mov(close,9,e))

Using Cross(), formula 27 can be rewritten as follows:

Formula 28:



Name: Buy Signal

Formula:
```
Cross( C, mov( C, 40, E))
```

Formula 28 in English

If the close crosses above the 40-period exponential moving average of the close, plot a 1, else plot a 0

This is a much simpler method. Cross() always lets you know when the value listed first increases past the value listed second. You can use any data arrays you want in the Cross() function but it only returns a true value when the first rises above the second.

A Quick Comparison

Here’s another quick shortcut for you. Suppose you wanted to make sure that some value was greater than two other values. An example would be the close being above its 9 period and 40 period moving averages. Normally, your would have to write this as:

```
( ( C > mov( C, 9, S ) ) AND ( C > mov( C, 40, S ) ) )
```

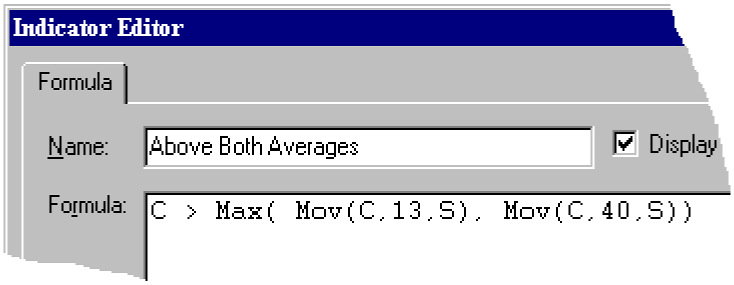
There is a way this can be condensed, however. The Max() function compares two values and returns the highest. The full text of the command follows:

Maximum	
SYNTAX	max(DATA ARRAY, DATA ARRAY)
FUNCTION	Returns the largest of the two parameters.
EXAMPLE	The formula "max(CLOSE, 10)" returns either the closing price or 10, whichever is greater. The formula "max(-14, 13)" always returns 13.

Exercise 20:

Some trading systems require the closing price to be above two different moving averages to return a valid buy signal.
Assume one such trading system uses a 13-period and a 40-period simple moving average. How would you use the Max() function here? Write a formula to do this and name it Above Both Averages. Look at Formula 29 for the answer:

Formula 29:



Name: Above Both Averages

Formula:
```
C > Max( Mov(C,13,S), Mov(C,40,S))
```

Formula 29 in English

If the close is greater than the larger value of the 13-period simple moving average of the close and the 40-period simple moving average of the close, plot a 1, else plot a 0.

This function may be a bit more cryptic, but it saves time when writing multiple comparisons. It also has a companion function, Minimum.

Minimum

SYNTAX min(DATA ARRAY, DATA ARRAY)

FUNCTION Returns the smallest of the two parameters.

EXAMPLE The formula "min(CLOSE, 10)" returns the closing price or 10, whichever is less. The formula "min(-14, 13)" always returns -14.

This function works the exact same way as Max(). In order to change the formula Above Both Averages to be true when the close was less than both moving averages, you only need to change the function name to Min and change the greater-than sign to a less-than sign. Formula 30 shows how the new formula would look.

Formula 30:

```
C < min( mov( C, 13, S ), mov( C, 40, S ) )
```

Formula 30 in English

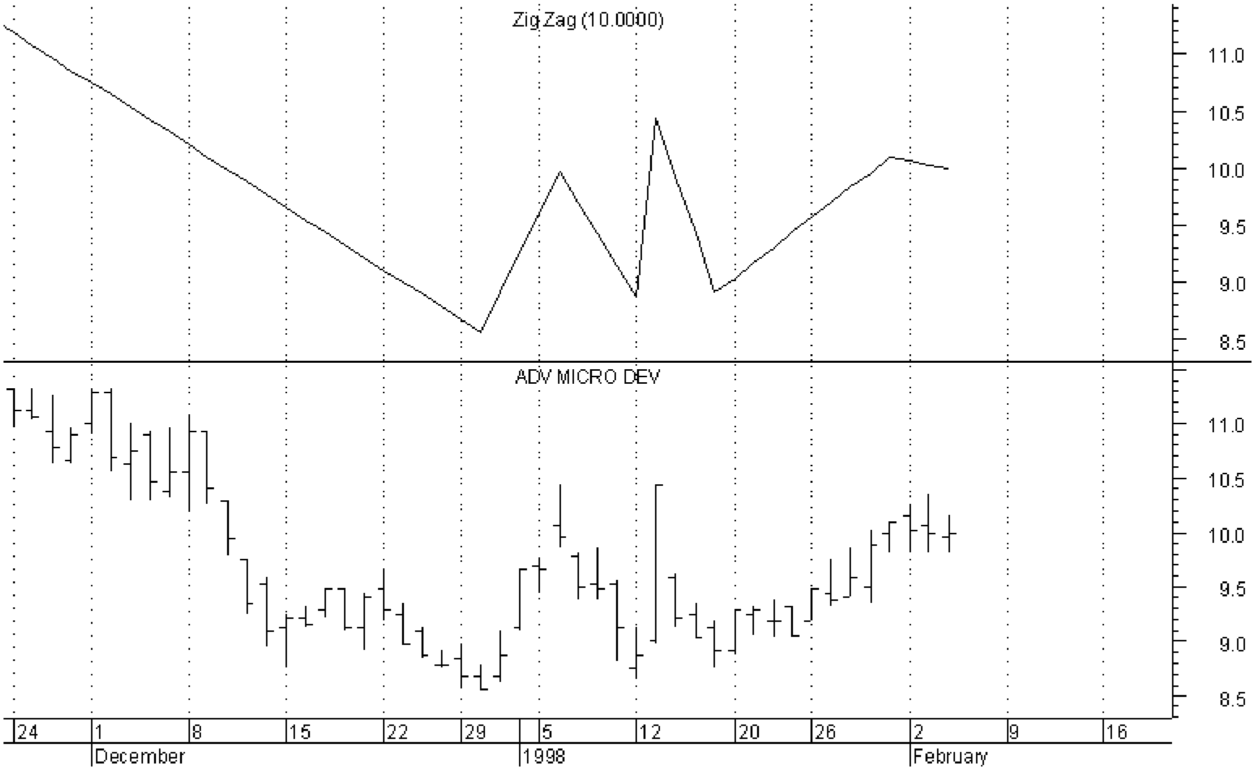
If the close is less than the least of the 13-period simple moving average of the close and the 40-period simple moving average of the close, plot a 1, else plot a 0.

Just like other functions, these two can be nested. Thus you could compare multiple values quickly without having to nest a lot of If()s.

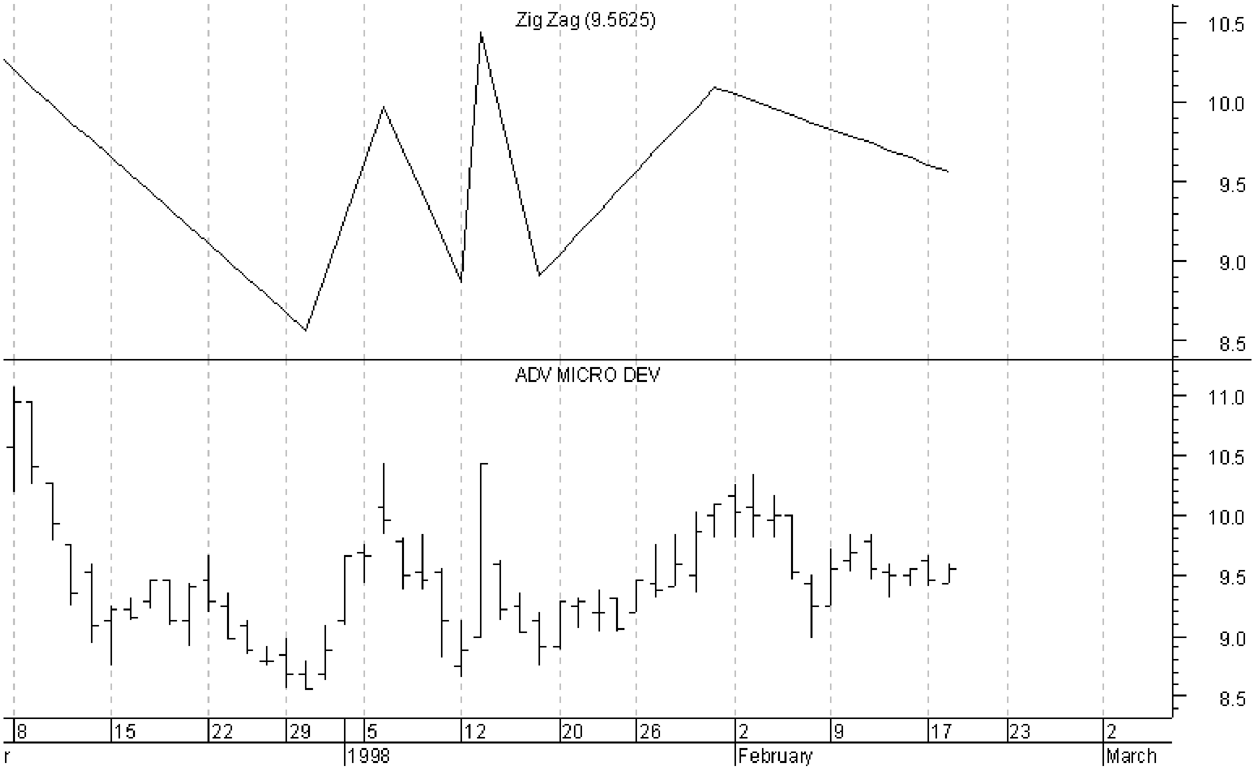
The Zig-Zag Ziggurat

The Zigzag indicator shows you the major moves of a plot by filtering out all moves of a certain percentage or less. This may seem like the holy grail of technical analysis, until you realize it does this after the fact.

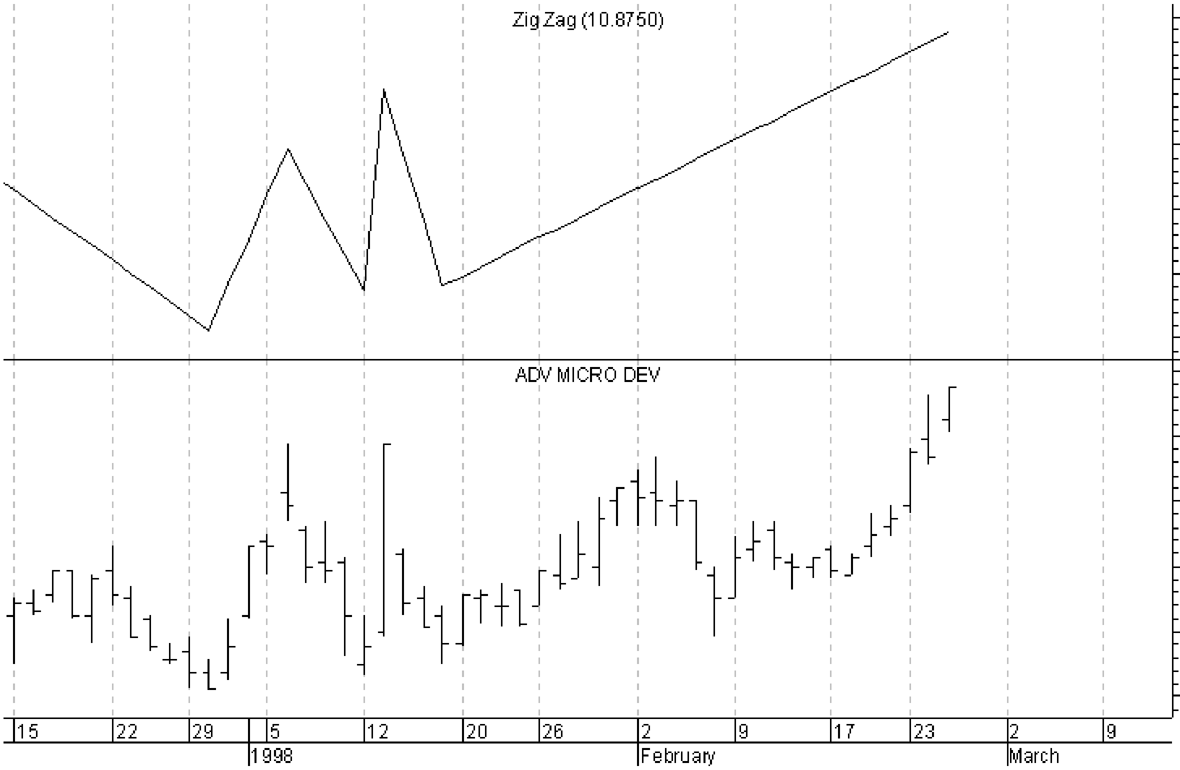
The Zigzag's plot is a line moving up and down in sync with the values it is plotted on, however, the last segment of the line is estimated. The direction of the last part of the line is determined by assuming that the plot will continue in the same direction. The following chart shows a stock and the Zigzag indicator with a ten percent filter.



This chart is accurate for the data that is shown. Notice the last part of the line is dipping downward even though the security has not lost ten percent of its value since the last peak. The Zigzag is anticipating the continued downward movement. Here is the same chart two weeks later.



Notice the line is still trending downward. The Zigzag is doing its job and has filtered out the minor drop and rise during the end of the first week of February. You might think by now that this last line is confirmed. Below is the same security one week later.



During the previous 3-week period, the prices never dropped the required ten percent. As a result, when the prices started rising again, the line corrected to show it as one continuous upward progression.

Understanding how and why the Zigzag does this is important because several functions in the MetaStock formula language are based on this indicator. These additional functions include Peak(), Trough(), and Divergence(), etc.

These functions monitor the general trends of the data array and report various values based on its movement.   However, because they are based on the Zigzag, the last value is always uncertain. These functions have many uses in examining historical data, but you should be careful with any buy and sell decisions based directly on them. Tomorrow, the signal might not be there.

Review Questions

1. What would be plotted by the formula: Cum(1)

2. Write the If() equivalent of the formula: Min(close, open)

3. True or False, The Cross() function returns a 1 when its data arrays cross each other.

4. True or False, The Alert() function allows you to specify conditions that do not have to happen simultaneously.

5. True or False, The Zigzag() function is a great tool for identifying trends and patterns.

Chapter 5
Getting Fancy

It's a Variable Circus

The MetaStock formula language draws many parallels from computer programming languages. It performs set calculations, can make decisions based on If() conditions, and has internal functions. It also has variables.

Anyone familiar with computer programming will find these are different from the variables they are used to. A *variable* in MetaStock is a name you can assign a static value to. *Static* means that once assigned, the value of the variable may not be changed.

If you have never dealt with programming before, you may still be confused about what exactly a variable is. Think of them this way: a variable is a symbol that stands for something else. For example, when you see this sign "=", you automatically think something like "equals" or "is equal to." That's because our culture has assigned that symbol the value of "equals."

In the same way, MetaStock allows you to create an object to represent something else. For instance, suppose you wanted to plot the difference between the stochastic oscillator and its signal line. Up until now, if you wanted to change the time periods for the stochastic, you would have to find each place it was used and manually change the value. Now, you can assign the time periods to the variable at the beginning of the formula and then use the variable where the Stoch() functions ask for the number of periods. Then, changing the time periods is easily accomplished by changing the value assigned to the variable. The Stoch() function is listed below for you convenience.

Stochastic Oscillator

SYNTAX	Stoch(%K PERIODS, %K SLOWING)
FUNCTION	Calculates the predefined Stochastic Oscillator.
EXAMPLE	The formula "Stoch(5, 3)" returns the value of a 5-period %K slowed 3-periods.

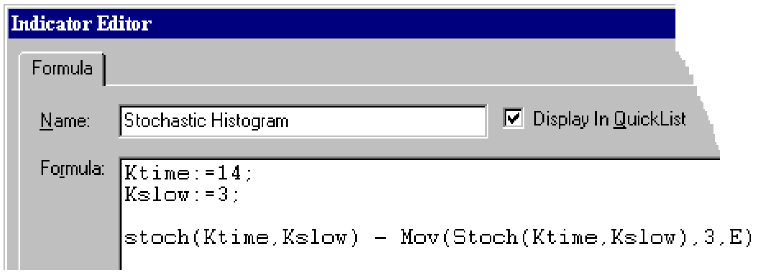
To create a variable, simply type the name of the variable followed by a colon and the equals sign. Then put whatever the variable is suppose to stand for. This entire line is ended with a semicolon. Below is an example of how this might look. Here the variable time is assigned the value of fourteen.

```
time:=14;
```

Exercise 21:

The MACD Histogram is the difference between the MACD and its signal line. For your first formula using variables, create a Stochastic Histogram. This indicator should subtract a 3-period exponential moving average of the stochastic from the stochastic. The values of %K and %K slowing should be assigned to variables named Ktime and Kslow, respectively. Set Ktime to equal 14 and Kslow to 3. Write a formula to do this and name it Stochastic Histogram. Look at formula 31 for the answer.

Formula 31:



Name: Stochastic Histogram

☑ Display In QuickList

Formula:

```
Ktime:=14;
Kslow:=3;

stoch(Ktime,Kslow) - Mov(Stoch(Ktime,Kslow),3,E)
```

Formula 31 in English

Assign the number 14 to the value Ktime.

Assign the number 3 to the value Kslow.

Subtract a Ktime periods stochastic using Kslow period slowing from a 3-period exponential moving average of the same stochastic.

More Input Please

So far, the variable may not seem that useful. However, consider the Input() function listed below.

Input

SYNTAX input("PROMPT TEXT", MINIMUM VALUE, MAXIMUM VALUE, DEFAULT VALUE)

FUNCTION This function instructs MetaStock to prompt for input when a custom indicator is plotted. This function is only supported by the Custom Indicator Builder.

prompt text. This defines the text displayed next to the input box. This is used to describe what should be entered.

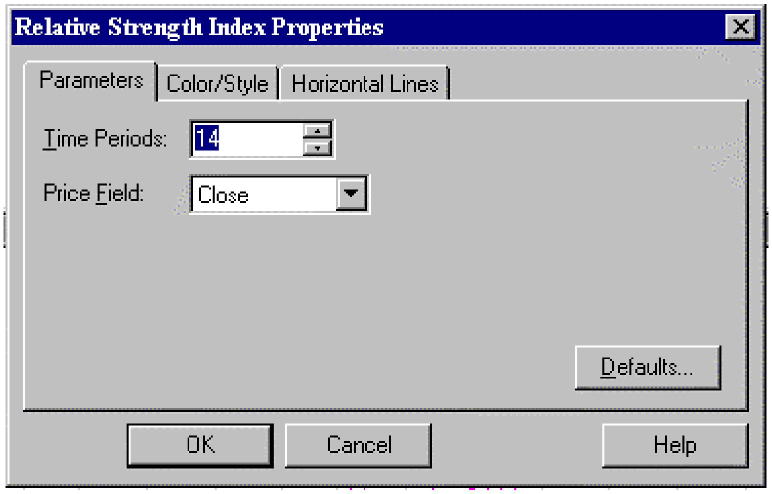
minimum value. This argument is the smallest value that can be entered. If you attempt to enter a value smaller than this value, MetaStock displays a message.

maximum value. This argument is the largest value that can be entered. If you attempt to enter a value larger than this value, MetaStock displays a message.

default value. This argument defines the default value (i.e., the value that will appear in the box when the dialog is initially displayed.) . Note that the default value is used if another formula using the fml() function calls the custom indicator.

EXAMPLE input("Enter the number of periods",1,50,9)

This rather verbose function allows the creation of custom indicators that request information from you when they are plotted.



Time Periods: 14

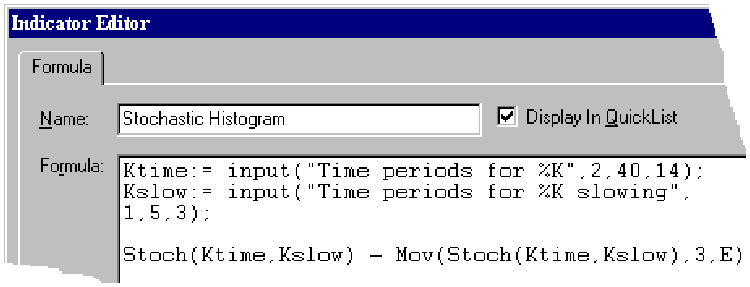
Price Field: Close

Just like how the RSI indicator of MetaStock prompts for the time periods, you can add the Input() prompt to the Stochastic Histogram formula and have it do the same.

Exercise 22:

Edit the formula Stochastic Histogram and change the values of Ktime and Kslow to Input() functions. The prompt text should state that you are asking for the time periods to be used. For Ktime, set the minimum value to 2 and the maximum to 40. For Kslow, use a minimum of 1 and a maximum of 5. The default can be set to any value between the minimum and maximum, use 14 and 3 if no other numbers present themselves. Look at Formula 32 for the answer.

Formula 32:



Name: Stochastic Histogram

☑ Display In QuickList

Formula:

```
Ktime:= input("Time periods for %K",2,40,14);
Kslow:= input("Time periods for %K slowing",
1,5,3);

Stoch(Ktime,Kslow) - Mov(Stoch(Ktime,Kslow),3,E)
```

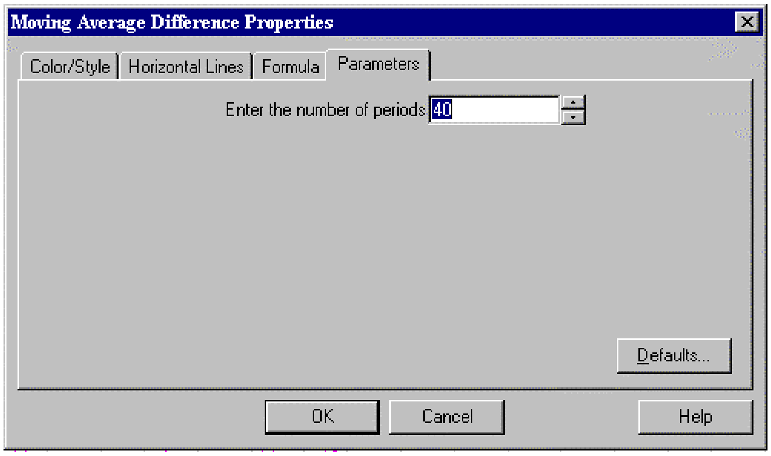
Formula 32 in English

Use the text “Time periods for %K” to prompt for a number between 2 and 40 with a default value of 14. Assign this number to the value Ktime.

Use the text “Time periods for %K slowing” to prompt for a number between 1 and 5 with a default value of 3. Assign this number to the value Kslow.

Subtract a Ktime periods stochastic using Kslow period slowing from a 3-period exponential moving average of the same stochastic.

You are allowed up to six Input() functions per formula. When the indicator is plotted, a window comes up with each input on a separate line followed by the text you entered for a prompt.



Enter the number of periods 40

A word of caution; if you reference a formula containing an Input() in another formula, the default value will be used. The Input() function's prompt is only displayed if the indicator containing the Input() function is plotted.

More on Subs

The variable can also be used as a type of sub-routine. Any expression that is used several times in the formula may work better if replaced by a variable. This will shorten the formula overall and make sure the expression is used the same way throughout the formula. A good example of this can be found by looking at formula 33. This is the formula for the Chande Momentum Oscillator. Not exactly something you want to type in just for fun.

Formula 33

```
periods:= Input("periods",1,100,14);

((( Sum(If(C>Ref(C,-1),C-Ref(C,-1),0),periods) )-(
Sum(If(C<Ref(C,-1),Ref(C,-1)-C,0),periods) ))/((
Sum(If(C>Ref(C,-1),C-Ref(C,-1),0),periods) )+(
Sum(If(C<Ref(C,-1),Ref(C,-1)-C,0),periods) )))*100
```

Now, look at formula 34. This is the exact same formula except two variables have been defined.

Formula 34

```
periods:= Input("periods",1,100,14);

up:= Sum( If( C>Ref(C,-1), C-Ref(C,-1),0), periods);

down:= Sum( If( C<Ref(C,-1), Ref(C,-1)-C,0), periods);

((up-down)/(up+down))*100
```

This should be a lot easier to read. This formula should also be understandable by you. It uses only commands that have been covered in this text.

Exercise 23:

This exercise is a little different than the previous ones. Look at formula 34 and see what it is doing.   Write the English version of the formula. When finished, it should be similar to the text below.

Formula 34 in English

Prompt for a number between 1 and 100. Use 14 for the default and store the value Periods.

Calculate the difference between this period’s close and last period’s close if this periods close was higher. Total this for the last Periods amount of time and store the result in Up.

Calculate the difference between last period’s close and this period’s close if last period’s close was higher. Total this for the last Periods amount of time and store the result in Down.

Divide the value of Up minus Down by the total of Up plus Down and multiply the result by 100

The variable’s ability to break out a part of the formula and make it into a sub-formula allows you to both simplify the formula and make it easier to read.   This is especially helpful if you need to examine a formula for errors or wish to modify an existing formula.

Getting Ahead of the Game

There's one other thing you can do with variables. Several can be created and stored in a formula. These can be referenced later in other formulas through a function very similar to Fml(). Those familiar with other programming languages will recognize this as similar to creating a header file.

The principle here is to avoid having to retype the same expression in every indicator and/or formula you write. In the formula Weekly Volume, the DayofWeek() function was used to see if the period being checked was a Friday. If you find yourself writing formulas that check this frequently, it might be easier to create variable name "Friday." Store this variable in an indicator, perhaps named "header." Then whenever you needed to check if it was a Friday, you could reference the Friday variable in the header indicator.

This example is admittedly a little weak. You would probably type more text referencing the variable that you would entering the DayofWeek() function. However, if you had a longer mathematical expression, this could definitely save you time retyping it for every formula. You also don't have to worry about making any typos and if you decide to change it, you only have to edit one indicator and change one expression.

The full text of the new function is listed below.

Formula Variable Call

SYNTAX fmlvar("FORMULA_NAME", "VARIABLE_NAME")

FUNCTION Calls the custom indicator named FORMULA_NAME and returns the value contained in the custom indicator's variable named VARIABLE_NAME.

Both the formula's name and the variable's name must be contained in quotation marks (e.g., fmlvar("Secret A", "MyVar")).

If you change a formula or variable name, you must also change any fmvarl() calls that reference that formula and variable.

EXAMPLE fmlvar("MyIndicator", "MyVariableA")

When inserting this function in a formula, it is strongly urged to use the functions button, just like you would in adding a Fml() function. When you look at the list of custom indicators, any indicator with a variable in it will have a listing for the indicator itself and another listing for each variable in the indicator. The variables are listed after the indicator's name and are enclosed in parentheses.

It Takes Two to Tango

Ever wonder why the variables require that little semicolon after each of their definitions. It is just a symbol to MetaStock that another formula is going to be added to this indicator. Yes, another formula, as in more than one. The Indicator Builder allows multiple formulas to be typed in for each indicator. This results is multiple plots on your chart when the indicator is used.

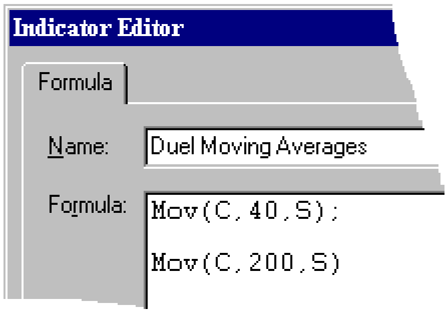
The only requirement is that each expression must be a complete formula or variable assignment. This little known feature is another one of those subtle things that may not immediately seem useful. After all, you're not likely to combine a modified MACD formula in the same indicator as another formula to spot unusually high and low volume. Well, maybe you might, but probably not as two separate plots.

The semicolon allows MetaStock to plot more than one formula with only one indicator. This is exactly like the bollinger bands or envelope indicators that come with MetaStock. That said, some useful purposes of multi-plot indicators should be apparent.

Exercise 24:

For this exercise, assume you always like to see both the 40-period and 200-period simple moving average when you look at a chart. You can easily plot both these custom indicators one at a time, but you would like to do it a lot faster. Write a formula to do this and name it Duel Moving Averages. Look at Formula 35 for the answer.

Formula 35



Name: Duel Moving Averages

Formula:

```
Mov(C,40,S);
Mov(C,200,S)
```

Notice that the last expression does not need the semicolon. If you put a semicolon at the end of an indicator, MetaStock expects to see something else. MetaStock will not give you an error if you have the semicolon there, but it really is not needed.

All things have their limits

You probably noticed that both the lines in the formula Duel Moving Averages plotted the same color. You may even have tried changing the color of one or the other. Unfortunately, this is not possible.

All lines plotted by one custom indicator have the same color and line style. After the lines are plotted, if you are using MetaStock version 7 or later, you can change the colors and styles individually. There is no way to set these values ahead of time.

You are limited in other ways, too. The Indicator Builder only allows 2500 characters in each indicator. A character for this purpose is any single letter, symbol or space.

There is also a limit on the number of expressions. You will most likely never have a problem with it, but you cannot have more than 20 different plots in one indicator. You also cannot have more than 20 different variables defined in a single indicator. Any formula with 40 expressions will truly be a sight to behold.

P Soup Anyone

There is one more variable that needs to be discussed. This last variable is a hybrid between the price field identifiers discussed in chapter 1 and the variables just mentioned. It is referred to only by the letter p.

The p variable takes the value of whatever plot is selected when it formula is calculated. For example, when you are plotting a moving average in a chart, some existing plot turns a purplish / magenta color. As soon as you let go of the mouse, the color returns to normal and the moving average is calculated on whatever was the magenta color. You could write a moving average formula to do the exact same thing. Consider formula 36 below.

Formula 36

```
Mov( p, 14, S)
```

The p variable is used where the data array would be specified. This tells MetaStock to calculate the moving average using whatever plot is highlighted / selected when the formula is used.

What's the point you may ask? Any plot on your chart can be duplicated in the MetaStock formula language, so why use p instead of the formula for the plot?

Versatility is the major reason. If you tend to use the same formula a lot and only vary the data array, it might be easier to write the formula once using p. You could then plot the base calculation and apply the p variable's indicator for the final plot.

For example; suppose you wanted to plot the MACD() or the RSI() of an indicator. Both of these formulas automatically assume you want to use the closing price. To use an indicator, you would have to create a custom formula and include the formula of the indicator where the close is normally used. If you had several custom indicators you wanted to do this to, you would have to duplicate the formula once for each indicator. Or, you could create the formula once and use the p variable where the closing prices are used.

Roll Your own Indexes

A second use of the p variable is to create a custom index. The MetaStock software, by itself, can add only two securities together. This is done with a composite in the DownLoader. It can only combine two regular security data files and nothing you do in the MetaStock or the DownLoader can increase that. Except a custom formula using the p variable.

Normally, an index has more than two securities in it. Most have eight or more. To make a custom index of any number of securities, use the following steps. The example will use only eight securities named A, B, C, D, E, F, G and H.

1. In the DownLoader create four composites. Choose Add as the operation for each composite. For this example the resulting composites are called AB, CD, EF, and GH. **Note: both components of a given composite must be in the same directory.**
2. In MetaStock, open up all four composites. Choose Window | Stack to show them all on the screen at one time. **Note: make sure that 'Remove Existing Charts' is not selected under File | Open when opening the charts.**
3. Create a custom indicator containing the formula C+P
4. Click on the CD plot and drag it onto the same chart as the AB composite. Close the CD chart.

5. Drag the C+P indicator and drop it onto the CD plot. Note: before dropping the indicator, make sure the CD plot turns magenta. When you see the scaling menu, choose 'Overlay without scale'.
6. Drag the resulting plot and drop in into the EF chart. Close the AB chart.
7. Drag the C+P indicator and drop it onto the new indicator plot in the EF chart. Note: before dropping the indicator, make sure the other indicator plot turns magenta. When you see the scaling menu, choose 'Overlay without scale'.
8. Drag the resulting plot onto the GH chart. Close the EF chart.
9. Drag the C+P indicator and drop it onto the new indicator plot in the GH chart. Note: before dropping the indicator, make sure the other indicator plot turns magenta. When you see the scaling menu, choose 'Overlay without scale'.
10. The resulting plot is the end result. You can take this plot and drop it into any chart that you would like. MetaStock automatically remembers what components were used to calculate it. It will automatically be updated from the information in all eight securities

Warning: P at work

The p variable has its limits. When it is plotted on a security's prices, it assumes you want the closing price. There is no way to make it use a different price field unless you create a custom indicator using just that price field.

Once you plot an indicator using p, you cannot change what p refers to unless you plot the indicator in the chart again, making sure the desired plot is colored magenta.

You also cannot refer to two different plots in the same formula. P always has the same value through the formula, no matter how many times or places you use it.

Acknowledging these restrictions, the p variable can still be very helpful. The custom index alone makes p extremely valuable. As you work with the MetaStock formula language, you are likely to find other uses. Don't be afraid to experiment, but remember the variable's limitations.

Professional Security

If you have MetaStock Professional version 7.0, a new function all but removes the need for the p variable. This function is Security().

Security Data

SYNTAX security("SYMBOL",DATA ARRAY)

FUNCTION Returns the value of DATA ARRAY for the specified security. If the security is in the same folder as the base security, a path is not required. You may also specify online data by using ONLINE: as the path.

EXAMPLES security("C:\Data\Sample\IBM",C)

security("ONLINE:IBM",C)

security("IBM",C)

With this function, you can quickly get the value of any data array from any security to use in any formula. The method of writing a customized index described with the p variable is now reduced
to an eight line formula.

Please note that the examples show three different ways of listing the ticker symbol. If the symbol is listed by itself, MetaStock looks for the symbol in the same folder as the base security of the chart you are currently looking at. If you insert the text ONLINE:, MetaStock goes to your Internet data vendor and collects the data as it is required. The final option of using the complete path specifies the exact location on your local drive where the data file can be found. If this location changes, or is incorrect, the formula will not work.

Exercise 25

Assume you want to make a custom index with the Security() function. The ticker symbols for each security you want to use is a single letter and the letters are A through H, inclusive. Since all your data is in the same folder, do not specify a path in the Security() function parameters. Write a formula to do this and name it My Index. Look at Formula 37 for the answer.

Formula 37

```
Security("A", C ) + Security("B", C ) + Security("C", C ) +
Security("D", C ) + Security("E", C ) + Security("F", C ) +
Security("G", C ) + Security("H", C )
```

Review Questions

1. What does the semicolon (;) at the end of a line mean.

2. Below is the indicator My Formula. If this is referenced with the function Fml("My Formula"), what value will be returned?

My Formula

```
apples := High;
oranges := Low;
(apples + oranges)/2;
H - L;
Close
```

3. True or False, you can exactly duplicate the built in MACD indicator, including the signal line, with a single custom indicator.

4. Are there any disadvantages to using the Fmlvar() function?

5. The p variable has what value in a function?

Chapter 6
Putting It All Together

By now you have seen about everything the MetaStock formula language can do. You've seen how to use math with the different data arrays and functions. If()s can be proposed and decisions made based on the outcome. You can build nests as well as any bird and you know how to gather input for your formulas. These are the building blocks for everything else.

When the cathedrals of Europe were built during the middle ages, their architects only needed three tools. They used a straight edge, a compass, and a square. From these very basic tools of geometry, the majestic buildings were carved from stone and raised to inspire their beholders.

You have been given the basic tools of the MetaStock formula language. Your first formulas will probably not equate to the Sistine Chapel, but through practice, you will find few limits beyond what you can imagine.

Where to go from Here

This text has been devoted to showing you how the MetaStock formula language works. It has only used the Indicator Builder. MetaStock, however, has three other power tools. Each of these uses the exact same formula language, but applies it slightly differently.

To fully detail all that can be done with these other tools is beyond the scope of this text. However, the following is a brief overview of how you can apply the concepts from here to these other tools.

Starting with the Indicator Builder, a sample trading system will be built and tested. An exploration will be created to search your securities for the appropriate buy and sell conditions, and an expert will be made to show these values on your chart.

In each case, the most common use of the tool will be described. The type of formula expected will then be detailed as well as any special requirements for formulas. If you have any further questions about these advanced features, please refer to the MetaStock manual or contact Equis technical support.

Before beginning, please note that this is just a sample system for the purpose of explaining how to use the formula language. Equis does not warrant or make any promises about the effectiveness of this system if used in the marketplace.

The Indicator Builder

Assume that you have examined your charts with various indicators. Through perseverance you have discovered that a moving average crossover gives pretty good signals, if it is in the direction of the current trend. You used simple moving averages of 40 and 21 periods for these observations. To determine the trend, you decide to compare the closing price to a 200-period simple moving average. Positions will be exited on the 21 period moving average crossing in the opposite direction.

This gives a buy signal of:
1) The close is greater than its 200-period simple moving average.
2) The 21-period simple moving average of the close has just risen above the 40-period simple moving average of the close.

The sell short signal would be:
1) The close is less than its 200-period simple moving average
2) The 21-period simple moving average of the close has just fallen below the 40-period simple moving average of the close.

Write the formulas for the Long and Short entry signals. Look at Formulas 38 and 39 for the answer

<u>Formula 38</u>

```
{ buy signal }

C > Mov( C, 200, S ) AND
Cross( Mov( C, 21, S ), Mov( C, 40, S ) )
```

<u>Formula 39</u>

```
{ sell signal }

C > Mov( C, 200, S ) AND
Cross( Mov( C, 40, S ), Mov( C, 21, S ) )
```

These formulas should have just been a simple review. If you had any difficulty, please reread the earlier chapters before continuing.

The System Tester

The system tester is designed to gauge how profitable a trading system is by applying its buy and sell conditions to historical data. Each system test has one to four formulas. Each formula specifies the condition for either entering or exiting a long or short position.

When writing the formulas, you only need the condition part of a basic binary wave. MetaStock will scan the historical data for a point that matches an entry condition. After finding one, it will proceed forward through the data looking for the conditions to trigger an entry for the opposite condition, the current position's exit condition, or any one of five possible stops.

From the formula writing perspective, just remember you only need the condition part of an If() statement. MetaStock will take care of everything else.

Therefore, the two formulas written above can be repeated for the Enter Long and Enter Short Conditions of a system test. The Exit conditions are just the reverse Cross() functions without the close comparisons. The full system test would then be written as:

```
Enter Long:
        C > Mov( C, 200, S ) AND
        Cross( Mov( C, 21, S ), Mov( C, 40, S ) )

Exit Long:
        Cross( Mov( C, 40, S ), Mov( C, 21, S ) )

Enter Short:
        C < Mov( C, 200, S ) AND
        Cross( Mov( C, 40, S ), Mov( C, 21, S ) )

Exit Short:
        Cross( Mov( C, 21, S ), Mov( C, 40, S ) )
```

After running the test on several securities, it is highly likely that it will not be as profitable as you desire. There is a feature in the System Tester that allows you to optimize your tests. It allows you to have MetaStock try different values for the numerical parameters of your formulas and will report which ones generated the most profit. The manual explains how the optimization variables are set, but the formulas using these variables for the number of periods in the three different moving averages would look like this:

```
Enter Long:
        C > Mov( C, OPT1, S ) AND
        Cross( Mov( C, OPT3, S ), Mov( C, OPT2, S ) )

Exit Long:
        Cross( Mov( C, OPT2, S ), Mov( C, OPT3, S ) )

Enter Short:
        C < Mov( C, OPT1, S ) AND
        Cross( Mov( C, OPT2, S ), Mov( C, OPT3, S ) )

Exit Short:
        Cross( Mov( C, OPT3, S ), Mov( C, OPT2, S ) )
```

In the above example, OPT1 is set to the length of the longest moving average, OPT2 is the medium length and OPT3 is the shortest. After many tests on different securities, assume you find the optimum lengths for this system are 150, 26 and 9. These values will be used from now on in the rest of this example

The Explorer

MetaStock's explorer allows you to scan through large numbers of securities and get a report on certain technical custom indicators. It will also tell which securities meet a specified condition. Each exploration can use up to seven formulas. Each formula is put in a column labeled A through F and one extra section marked Filter.

The Filter column will allow you to specify the type of securities you want to see. This is a condition just like those used in If() functions and the system tester. If the security's data meets the condition, it is included in the report. If not, the security is rejected. That is all the filter does.

The columns A through F will accept any formula you put in them. When the Explorer displays its report, it will include the value of the formulas in these columns. For example, if you were to put the formula for a 14-period simple moving average in column A, the report would give you the numeric value of the moving average for each security in the report. The formulas in the columns are not used to filter the securities unless they are specifically referred to in the filter.

For this example, you will want to make 4 explorations. They will all be the same except for the filter. Columns A through D will contain the Closing price, the 150 period moving average, the 26 period moving average, and the 9 period moving average respectively. These values are listed here merely for informational purposes. The filter will contain the entry and exit signals relative to the exploration being run. In this manner, the exploration for the long position buy signal would have the following formulas:

```
Column A:
        Close

Column B:
        Mov( C, 150, S )

Column C:
        Mov( C, 26, S )

Column D:
        Mov( C, 9, S )

filter:
        C > Mov( C, 150, S ) AND
        Cross( Mov( C, 9, S ), Mov( C, 26, S ) )
```

Since the filter can reference the values in the columns, it could have been written as:

```
filter:
        ColA > ColB AND
        Cross( Mov( C, 9, S ), Mov( C, 26, S ) )
```

Please note that the columns only report the last value of the formula listed for them. Therefore the columns are just a single value. They cannot be used as data arrays for other functions. Thus the filter will NOT calculate correctly if it is written as:

```
filter:
        ColA > ColB AND
        Cross( ColD, ColC )
```

The Expert Advisor

Many different capabilities are lumped into the expert advisor. The main theme behind them all is to give on-screen analysis of whatever security you are looking at. This is done through a ribbon bar, highlights, symbols, alerts, and a commentary. All but the commentary use the condition part of an If() function to determine when a specified signal is displayed.

For this example, a ribbon will be created to show the current trend and symbols assigned to show entry and exit conditions on the chart.

The formulas for the symbols are the same as the ones used in the explorations. Thus the formula for a symbol to show where you should have entered a long position would be:

```
filter:
        C > Mov( C, 150, S ) AND
        Cross( Mov( C, 9, S ), Mov( C, 26, S ) )
```

For the ribbon, all that is needed is the comparison of the close to the long term moving average. The ribbon asks for both a bullish and a bearish formula so you would use these:

```
bullish formula:
        C > Mov( C, 150, S )

bearish formula:
        C < Mov( C, 150, S )
```

The commentary is a bit more complex. It functions as a combination of features. Any value can be displayed, just like with the columns in the explorer. However, the commentary is not limited to six. Put in as many as you want.

The commentary also can display text messages about the security. These messages can even be altered based on conditions you specify. These are the same type of conditions you use in the rest of the expert, but you must use the special WriteIf() function. For more information on this function and the expert advisor, please consult your manual or examine some of the sample commentaries sent with MetaStock.

Any Last Words

This concludes your introduction to the MetaStock formula language and the power tools. You should have a grasp of all the basics and be able to apply that to any of the power tools. Additional information is contained in the MetaStock manual and segregated by topic. Together, these advanced features take up most of the last half of the manual.

Specific questions on what a function does, how to use it, or why something may not be working as expected, should be addressed to Equis customer support. They are very willing to help others learn how to use the MetaStock formula language.

Unfortunately, the support technicians cannot write formulas over the phone for you. If you do not know how to write a desired formula, try the following steps. After each step, try to write the formula. If you are still unable to, proceed to the next step.

1. Write in English what you want the formula to do.
2. Try writing out the logic the formula should follow. Use pseudo-formula commands if necessary (similar to what this text used in discussing the If() function's conditions).
3. Examine the text you wrote for the first two steps and make sure you did not gloss over any details. If you can add detail to any part, re-write it with the extra details.
4. Call Equis technical support. They will not be able to write it for you, but may suggest something you have not thought of.
5. Send in a custom formula request to Equis technical support. Be sure to include your work so far. This last option does have fee involved based on the time required to write the formula. The more detail provided, the less time may be required.

Use your imagination and good luck with your trading.

Formula Reference

Absolute Value

SYNTAX abs(DATA ARRAY)

This function returns the absolute value of the specified data array. The absolute value is equal to the value of the number without any positive or negative sign. Therefore the absolute value of 10 is equal to the absolute value of -10.

This function is most useful if you are looking for a value that does not exceed a certain range, either up or down. For example, to look for securities whose price has changed by more than 15%, you could type in the following function.

```
Abs( Roc( C, 1, %) ) > 15
```

Accumulation/Distribution

SYNTAX ad()

This calculates the value of the Accumulation/Distribution indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Accumulation Swing Index

SYNTAX aswing(LIMIT MOVE)

This calculates the value of the Accumulation Swing Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

This indicator requires the base security to have data on the opening prices or it cannot be calculated.

Addition

SYNTAX add(DATA ARRAY, DATA ARRAY)

This function is nothing more than a long way to add two values together. Writing:

```
add( 10, 20 )
```

is exactly the same as writing

```
10 + 20
```

The MetaStock formula language will treat them exactly the same. Feel free to use whichever you are most comfortable with.

Alert

SYNTAX alert(EXPRESSION, PERIODS)

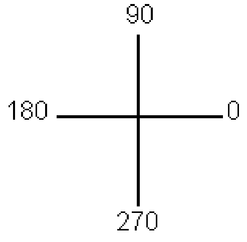
The Alert() function checks to see if a given condition has occurred at any time within a specified length of time. The condition is specified in the expression and the time frame is denoted in the PERIODS part of the function. If the condition occurs, the Alert() returns the value of 1. Otherwise, it returns a 0.

For more information on this, look in chapter 4.

Arc Tangent

SYNTAX atan(Y DATA ARRAY, X DATA ARRAY)

This is a trigonometric function dealing with the ratio between angles and the legs of a right triangle. The value is returned in degrees from 0 to 359.9. The degrees are returned as shown below:



EXAMPLE The formula "atan(10, 0)" returns 90.

This will probably never be used but is included should a formula require it.

Aroon Down

SYNTAX aroondown(PERIODS)

This calculates the value of the Aroon Down component of the predefined Aroon indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Aroon Up

SYNTAX aroonup(PERIODS)

This calculates the value of the Aroon Up component of the predefined Aroon indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Average Directional Movement

SYNTAX adx(PERIODS)

This calculates the value of the Average Directional Movement indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Average True Range

SYNTAX atr(PERIODS)

This calculates the value of the Average True Range indicator. This is an average over a specified number of periods of the True Range as calculated by Wilder.

This is normally not used in formulas except to compare the value of this indicator to some other value. However, the True Range has been used as a volatility gauge. As such, other uses are possible. This would most likely involve using he volatility as a smoothing value for some other calculation, the theories of which are not for this document to discuss.

Bars Since

SYNTAX barssince(DATA ARRAY)

This function counts how much time has passed since an event occurred. For example:

```
BarsSince( C > 40 )
```

This returns the number of periods that have passed since the closing price was greater than 40. If this is currently true, it will return a 0.

Warning. Until this event is true, any formula using it will not be calculable. This will cause custom indicators to not plot and other formula based tools to do nothing or reject the security.

Important: When using the barssince() function in an exploration, you must choose the "Load ___ Records" button in the Explorer Options dialog and specify a value equal to the number of bars loaded in your chart; otherwise, the exploration results may not be accurate.

Bollinger Band Bottom

SYNTAX bbandbot(DATA ARRAY, PERIODS, METHOD, DEVIATIONS)

This calculates the value of the bottom Bollinger Band of DATA ARRAY. METHOD refers to the type of moving average to use. Valid methods are SIMPLE, EXPONENTIAL, WEIGHTED, TIMESERIES, TRIANGULAR, and VARIABLE (these can be abbreviated as S, E, W, T, TRI, and VAR). DEVIATIONS refers to the number of standard deviations the band is shifted downward.

Bollinger Band Top

SYNTAX bbandtop(DATA ARRAY, PERIODS, METHOD, DEVIATIONS)

This calculates the value of the top Bollinger Band of DATA ARRAY. METHOD refers to the type of moving average to use. Valid methods are SIMPLE, EXPONENTIAL, WEIGHTED, TIMESERIES, TRIANGULAR, and VARIABLE (these can be abbreviated as S, E, W, T, TRI, and VAR). DEVIATIONS refers to the number of standard deviations the band is shifted upward.

Buying Pressure

SYNTAX buyp()

This calculates the value of the buying pressure component of the predefined Demand Index. Buying pressure is a measurement of the amount of volume related to buying.

This function is not normally used in formulas except to compare the value of this indicator with some other value.

Ceiling

SYNTAX ceiling(DATA ARRAY)

The Ceiling() function returns the lowest integer that is greater that the largest value in the data array. For example:

```
Ceiling ( High )
```

This would return the lowest integer that is larger than the highest high value loaded. For example if you had three years of data loaded and the highest value ever reached by the High during that time was 103 5/8ths, the formula will return 104.

This function is probably more useful in system tests and explorations, but can help you plot and calculate the new highs of securities.

Chaikin A/D Oscillator

SYNTAX co()

This calculates the value of the Chaikin A/D Oscillator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Chaikin's Money Flow

SYNTAX cmf(PERIODS)

This calculates the value of the Chaikin's Money Flow indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Chande Momentum Oscillator

SYNTAX cmo(DATA ARRAY, PERIODS)

This calculates the value of the Chande Momentum Oscillator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Commodity Channel Index (EQUIS)

SYNTAX ccie(PERIODS)

This calculates the value of the Commodity Channel Index. This is done using the Equis variant calculation. It is not normally used in formulas except to compare the value of this indicator with some other value.

Commodity Channel Index (Standard)

SYNTAX cci(PERIODS)

This calculates the value of the Commodity Channel Index. This is done using the standard method of calculation. It is not normally used in formulas except to compare the value of this indicator with some other value.

Commodity Selection Index

SYNTAX csi(PERIODS, VALUE, MARGIN, COMMISSION)

This calculates the value of the Commodity Selection Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

Correlation Analysis

SYNTAX correl(INDEPENDENT, DEPENDENT,PERIODS, SHIFT)

This calculates the value of the Correlation Analysis indicator. It compares the correlation of the DEPENDENT data array to the INDEPENDENT data array. This is statistically averaged over PERIOD time periods after shifting the DEPENDENT data array to the right by SHIFT periods. For example:

```
correl( macd(), CLOSE, 5, 10 )
```

This compares the MACD indicator to the closing price 10-periods in the future, after statistically averaging each data array over the preceding 5-periods.

Cosine

SYNTAX cos(DATA ARRAY)

Calculates the geometric function Cosine. This is a cyclic line alternating between 0 and 1 base on the angle value used for the DATA ARRAY. Any number larger that 360 has 360 subtracted from it until it is between 1 and 360 and then the function is calculated.

Normally this function is not used in formulas but it is useful to plot or watch a cycle on a chart. For example:

```
Cos( Cum( 1 ) )
```

With this function, 1 will be added constantly and each period the Sine() plot will continue in its cycle. After 360 periods, it will be back to it's original value, thus marking a 360 period cycle on your chart or in your formula.

To work with smaller time periods, increase the number in the Cum() function so it reaches 360 sooner. The exact number to use can be found by the equation:

360 / number of periods in cycle

You could even put it in your formula that way. For example, if the security had a 17 period cycle, you could write:

```
Cos( Cum( 360 / 17 ) )
```

Cross

SYNTAX cross(DATA ARRAY 1, DATA ARRAY 2)

This function returns a 1 on the period that DATA ARRAY 1 rises above DATA ARRAY 2. At all other times, a 0 is returned. For more information, refer to chapter 4.

Cumulate

SYNTAX cum(DATA ARRAY)

This function keeps a running total of the DATA ARRAY. Each period it is calculated, it adds the current value of DATA ARRAY to the previous total. For example:

```
cum( 1 )
```

This will keep adding 1 for each period of time loaded in the chart. In effect, it counts how many records are currently loaded.

Day Of Month

SYNTAX dayofmonth()

This returns the day of the month. If today was July 15, 1998, this function would return 15. Normally the date functions are used in Expert Advisors to display the actual date of the commentary. Below is an unusual use:

```
Posit:=Input("How many Positions",1,10000,100);
Days:=Input("What Day of the month",1,31,1);
Months:=Input("Which month",1,12,1);
years:=Input("Which year",1955,1999,1997);
commis:=Input("dollar amount of commission",0,1000,0)*-
1;

If(Year()>years,
   ((ROC(C,1,$)*posit)+PREV),
   If(Year()=years,
   If(Month()>months,
      ((ROC(C,1,$)*posit)+PREV),
      If(Month()=months,
         If(DayOfMonth()>days,
            ((ROC(C,1,$)*posit)+PREV),
            commis),
      commis)),
   commis))
```

This formula will plot the profits from a position taken in a given security. First, all the information required for the calculation are gathered through input() prompts. A series of If()'s then check to see if the specified date has occurred or not. If it has not, the commission's expense is displayed.

One the date has past, it multiples the change in price times the number of positions and adds the result to the value from the previous period. That is what the PREV stands for.

PREV is a special value the refers to the current formulas' value for the previous period. It makes a formula slower to calculate and increases the discrepancies if there is an error in the formula. However, PREV is the only way to create self-referencing formulas in MetaStock

Day Of Week

SYNTAX dayofweek()

This returns an number equal to the day of the week. Monday is assigned the value of 1 and the other days follow sequentially:

1=Monday
2=Tuesday
3=Wednesday
4=Thursday
5=Friday
6=Saturday
7=Sunday.

Delta

SYNTAX delta(TYPE, DATE, PRICE, INTEREST, DIVIDEND)

This calculates the value of the Delta indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Dema

SYNTAX dema(DATA ARRAY, PERIODS)

This calculates the value of the Dema indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Demand Index

SYNTAX di()

This calculates the value of the Demand Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

Detrended Price Oscillator

SYNTAX dpo(PERIODS)

This calculates the value of the Detrended Price Oscillator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Directional Movement Index

SYNTAX dx(PERIODS)

This calculates the value of the Directional Movement Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

Directional Movement Rating

SYNTAX adxr(PERIODS)

This calculates the value of the Directional Movement Rating. It is not normally used in formulas except to compare the value of this indicator with some other value.

Divergence

SYNTAX divergence(DATA ARRAY 1, DATA ARRAY 2, % MINIMUM CHANGE)

This returns either a +1, a -1, or a 0 depending on how closely the values of DATA ARRAY 1 follows the values of DATA ARRAY 2. If they diverge (i.e., the first array is increasing and the second is decreasing), it returns a +1. If the two arrays converge (i.e., the first is decreasing and the second increasing), it returns a -1. In all other cases, it returns a 0. Movements in DATA ARRAY 1 less than % MINIMUM CHANGE are ignored.

The Divergence function is based on the Zig Zag formula. Refer to Chapter 4 for additional information on using Zig Zag based function.

Division

SYNTAX div(DATA ARRAY, DATA ARRAY)

This function is nothing more than a long way to divide two values. Writing:

```
div( 10, 2 )
```

is exactly the same as writing

```
10 / 2
```

The MetaStock formula language will treat them exactly the same. Feel free to use whichever you are most comfortable with.

Dynamic Momentum Index

SYNTAX dmi(DATA ARRAY)

This calculates the value of the Dynamic Momentum Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

Ease of Movement

SYNTAX emv(PERIODS, METHOD)

Calculates a moving average the Ease of Movement value. The length of the moving average is specified in PERIODS and the type of average is defined by METHOD. Valid methods are SIMPLE, EXPONENTIAL, WEIGHTED, TIMESERIES, TRIANGULAR, and VARIABLE. (These can be abbreviated as S, E, W, T, TRI, and VAR.)

It function is not normally used in formulas except to compare the value of this indicator with some other value.

Exponent

SYNTAX exp(DATA ARRAY)

Calculates the mathematical constant e raised to the DATA ARRAY power. Normally this will only be used in Statistical calculations and curve fitting.

Fast Fourier Transform

SYNTAX fft(DATA ARRAY, PERIODS, LENGTH, DETREND or MEAN, AMPLITUDE or POWER)

This function calculates the Fast Fourier Transform indicator. The various parameters are all statistical information required for the calculation. The default fourier transform would be written as:

```
fft( CLOSE, 100, 1, DETREND, POWER )
```

Floor

SYNTAX floor(DATA ARRAY)

The Floor() function returns the highest integer that is lower that the smallest value in the data array. For example:

```
Floor ( Low )
```

This would return the highest integer that is smaller than the lowest low value loaded. I.e. if you had three years of data loaded and the lowest value ever reached by the low during that time was 83 1/4ths, the formula will return 83.

This function is probably more useful in system tests and explorations, but can help you plot and calculate the new lows of securities.

Forecast Oscillator

SYNTAX forecastosc(DATA ARRAY, PERIODS)

This calculates the value of the Forecast Oscillator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Formula Call

SYNTAX fml("FORMULA_NAME")

This is a request for MetaStock to include the calculated value of another formula in your current one. When referencing a formula's name, the name must be contained in quotation marks (e.g., fml("Secret A")).

If you change a formula's name, you must also change any fml() calls that reference that formula.

For more information on this command, please refer to chapter 3.

Formula Variable Call

SYNTAX fmlvar("FORMULA_NAME", "VARIABLE_NAME")

This command retrieves the requested variable form the specified formula and incorporates it into the current formula. Both the formula's name and the variable's name must be contained in quotation marks (e.g., fmlvar("Secret A", "MyVar")).

If you change a formula or variable name, you must also change any fmvarl() calls that reference that formula and variable.

For more information on this command, refer to chapter 5.

Fraction

SYNTAX frac(DATA ARRAY)

This is a number manipulation function. It takes any number and removes everything to the left of the decimal point. This then is the fractional portion of the number.

For example, if the close was 10.75, then:

```
frac( C )
```

would return .75.

Gamma

SYNTAX gamma(TYPE, DATE, PRICE, INTEREST, DIVIDEND)

This calculates the value of the Gamma Indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Gap Down

SYNTAX gapdown()

This function looks for a gap down. The gap down is defined as any period in which the previous period's low is greater than the current period's high. When this occurs, this function returns a 1. At all other times it returns a 0.

Gap Up

SYNTAX gapup()

This function looks for a gap up. The gap up is defined as any period in which the previous period's high is less than the current period's low. When this occurs, this function returns a 1. At all other times it returns a 0.

Herrick Payoff Index

SYNTAX hpi(CENTS, MULTIPLYING FACTOR)

This calculates the value of the Herrick Payoff Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

Highest

SYNTAX highest(DATA ARRAY)

This function examines the entire list of values in DATA ARRAY and returns the one with the greatest value. This will mainly be used to check for new highs though other uses may be possible. The key point to remember is that this function reads all data available for its calculation

Highest Bars Ago

SYNTAX   highestbars(DATA ARRAY)

This function reads the DATA ARRAY and returns the number of periods that have past since its highest value. The key point to remember is that this function reads all the data available and bases its result accordingly.

Highest High Value

SYNTAX   hhv(DATA ARRAY, PERIODS)

Calculates the highest value of the DATA ARRAY from the most recent PERIODS number of records. The calculation includes the current period. For Example:

```
HHV( C, 5 )
```

This returns the highest closing price between the current period and the preceding 4 periods. If you do not want to include the current period in this calculation, then use the Ref() function:

```
HHV( Ref( C, -1), 5 )
```

This returns the highest close from the previous five periods, not including the current one.

Highest High Value Bars Ago

SYNTAX   hhvbars(DATA ARRAY, PERIODS)

Looks at the last PERIODS number of records in DATA ARRAY. This function then returns how many periods have passed since the highest value of those records examined. Remember that this function always includes the current period in its calculations.

Highest Since

SYNTAX   highestsince (Nth, EXPRESSION, DATA ARRAY)

This function looks at all the data available. It then evaluates the condition detailed in EXPRESSION. Afterwards, it counts back from the most recent point of data till it finds the Nth number of times the EXPRESSION was true.   From that point, it looks forward at DATA ARRAY and returns the highest value found. For Example:

```
highestsince( 2, h > 50, close )
```

This returns highest value of the close since the second most recent occurrence of the high being above 50.

Highest Since Bars Ago

SYNTAX   highestsincebars(Nth, EXPRESSION, DATA ARRAY)

This function looks at all the data available. It then evaluates the condition detailed in EXPRESSION. Afterwards, it counts back from the most recent point of data till it finds the Nth number of times the EXPRESSION was true.   From that point, it looks forward at DATA ARRAY and finds the highest value and. The final result of these calculations it the number of periods that have passed since this highest value. For Example:

```
highestsincebars( 2, c > 50, close )
```

This returns the number of periods that have passed since the highest value of the close (after the second most recent occurrence of the high being greater than 50).

If

SYNTAX   if(EXPRESSION, TRUE DATA ARRAY, FALSE DATA ARRAY)

This is the basic decision making function. It evaluates an expression and then returns one of two values. If EXPRESSION is true, the result is whatever is in the TRUE DATA ARRAY position. Otherwise it returns what is in the FALSE DATA ARRAY position. This command is discussed extensively in chapter 2.

Inertia

SYNTAX   inertia(REGRESSION PERIODS, RVI PERIODS)

This calculates the value of the Inertia indicator. The RVI PERIODS is the number of periods used for the Relative Volatility Index component of the indicator. This function is not normally used in formulas except to compare the value of this indicator with some other value.

Input

SYNTAX input("PROMPT TEXT", MINIMUM VALUE, MAXIMUM VALUE, DEFAULT VALUE)

This function allows the building of custom indicators that prompt for information before they are plotted, like the built in MetaStock functions.

PROMPT TEXT is the actual text to be displayed when the prompt window opens. This should be short but descriptive of what is being requested.

MINIMUM VALUE is the smallest number the Input() function is to accept

MAXIMUM VALUE is the largest number the Input() function is to accept

DEFAULT VALUE is the number the Input() function will offer when displayed and the value used if no changes are made.

This function is discussed in more detail in chapter 5.

Inside

SYNTAX inside()

This will return a +1 whenever an inside day occurs. It will continue to return this same value until either a Rally, Reaction, or an Outside day occur. At any time where Inside() is not equal to +1, it will be equal to 0.

An inside day is defined as any period where the current high is less than the previous period's high and the current low is greater than the previous period's low.

Integer

SYNTAX int(DATA ARRAY)

This will return only the non-fractional portion of DATA ARRAY. In other words, anything to the right of the decimal point is dropped.

IntraDay Momentum Index

SYNTAX imi(PERIODS)

This calculates the value of the IntraDay Momentum Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

Klinger Volume Oscillator

SYNTAX kvo()

This calculates the value of the Klinger Volume Oscillator. It does not calculate the trigger line. For that, a moving average of this function is required. This function is not normally used in formulas except to compare its value with some other value.

Last Value in Data Array

SYNTAX lastvalue(DATA ARRAY)

This function will always return the last value of DATA ARRAY. Put another way, the function:

```
lastvalue( close )
```

returns the closing price for the last day of data available. By the same token, the function:

```
lastvalue( MACD() )
```

returns the value the MACD has on the last day of data available.

This same value is returned no matter when the lastvalue() function is calculated. If the DATA ARRAY is undefined for any reason (i.e., not enough data to calculate), lastvalue will return a 0.

Since this function, in effect looks into the future, it is not acceptable for most custom indicators. It is; however; very beneficial for things like pattern recognition.

Linear Regression Indicator

SYNTAX linearreg(DATA ARRAY, PERIODS)

This calculates the value of the Linear Regression Indicator. This function is a good way of determining the general direction of a series of data is going. Otherwise, it is rarely used except to compare its value with another number.

Linear Regression Slope

SYNTAX	linregslope(DATA ARRAY, PERIODS)

This function returns the value of the Linear Regression Slope indicator. This is normally only used to compare its value to some other number.

Logarithm (natural)

SYNTAX	log(DATA ARRAY)

This function calculated the natural logarithm of DATA ARRAY. It will normally only be of use in curve fitting and exponential calculations. Some other statistical uses are possible but beyond the scope of this text.

Lowest

SYNTAX	lowest(DATA ARRAY)

This function examines the entire list of values in DATA ARRAY and returns the one with the lowest value. This will mainly be used to check for new lows though other uses may be possible. The key point to remember is that this function reads all data available for its calculation

Lowest Bars Ago

SYNTAX	lowestbars(DATA ARRAY)

This function reads the DATA ARRAY and returns the number of periods that have past since its lowest value. The key point to remember is that this function reads all the data available and bases its result accordingly.

Lowest Low Value

SYNTAX	llv(DATA ARRAY, PERIODS)

Calculates the lowest value of the DATA ARRAY from the most recent PERIODS number of records. The calculation includes the current period. For Example:

```
LLV( C, 5 )
```

This returns the lowest closing price between the current period and the preceding 4 periods. If you do not want to include the current period in this calculation, then use the Ref() function:

```
LLV( Ref( C, -1), 5 )
```

This returns the lowest close from the previous five periods, not including the current one.

Lowest Low Value Bars Ago

SYNTAX llvbars(DATA ARRAY, PERIODS)

Looks at the last PERIODS number of records in DATA ARRAY. This function then returns how many periods have passed since the lowest value of those records examined. Remember that this function always includes the current period in its calculations.

Lowest Since

SYNTAX lowestsince (Nth, EXPRESSION, DATA ARRAY)

This function looks at all the data available. It then evaluates the condition detailed in EXPRESSION. Afterwards, it counts back from the most recent point of data till it finds the Nth number of times the EXPRESSION was true. From that point, it looks forward at DATA ARRAY and returns the lowest value found. For Example:

```
lowestsince( 2, h > 50, close )
```

This returns lowest value of the close since the second most recent occurrence of the high being above 50.

Lowest Since Bars Ago

SYNTAX lowestsincebars(Nth, EXPRESSION, DATA ARRAY)

This function looks at all the data available. It then evaluates the condition detailed in EXPRESSION. Afterwards, it counts back from the most recent point of data till it finds the Nth number of times the EXPRESSION was true. From that point, it looks forward at DATA ARRAY and finds the lowest value. The final result of these calculations it the number of periods that have passed since this lowest value. For Example:

```
lowestsincebars( 2, c > 50, close )
```

This returns the number of periods that have passed since the lowest value of the close (after the second most recent occurrence of the high being greater than 50).

MACD

SYNTAX macd()

This calculates the value of the MACD indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

This function only deals with the main (solid) line. To get the signal line, use the Moving Average function with this as the data array.

Market Facilitation Index

SYNTAX marketfacindex()

This calculates the value of the Market Facilitation Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

Mass Index

SYNTAX mass(PERIODS)

This calculates the value of the Mass Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

Maximum

SYNTAX max(DATA ARRAY, DATA ARRAY)

This examines both values inserted for the DATA ARRAYs and returns which ever of the two has the largest numerical value. For more information on this command, please refer to chapter 4.

Median Price

SYNTAX mp()

This calculates the value of the Median Price indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

MESA Lead Sine

SYNTAX mesaleadsine(CYCLE LENGTH)

This calculates the value of the MESA Lead Sine indicator that plot along with the MESA Sine Wave indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

MESA Sine Wave

SYNTAX mesasinewave(CYCLE LENGTH)

This calculates the value of the MESA Sine Wave indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Midpoint

SYNTAX mid(DATA ARRAY, PERIODS)

This function calculates the midpoint of the DATA ARRAY over a specified period of time. The midpoint is defined as the point halfway between the highest value and the lowest value.

Minimum

SYNTAX min(DATA ARRAY, DATA ARRAY)

This examines both values inserted for the DATA ARRAYs and returns which ever of the two has the smallest numerical value. For more information on this command, please refer to chapter 4.

Minus Directional Movement

SYNTAX mdi(PERIODS)

This calculates the value of the Minus Directional Movement indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Modulus

SYNTAX mod(DATA ARRAY, DATA ARRAY)

Calculates the remainder after the first DATA ARRAY is divided by the second DATA ARRAY. For example:

```
Mod( 16, 3 )
```

This will divide 16 by 3. After 3 goes into 16 a total of 5 times, it can not do so anymore and has a remainder of 1. The above formula will return the value of 1.

This not normally used in functions but is quite useful for timing an event. For example, suppose a formula was suppose to do something on every fourth period of data array. Consider the following condition for an If() statement:

```
If( Mod( cum( 1 ), 4 ) = 0,  TRUE,  FALSE )
```

The cum() function counts how many periods have occurred. After every fourth period, it triggers a true response and the TRUE condition is executed. On all other periods, the FALSE condition is used.

Momentum

SYNTAX mo(PERIODS)

This calculates the value of the Momentum indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Money Flow Index

SYNTAX mfi(PERIODS)

This calculates the value of the Money Flow Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

Month

SYNTAX   month()

This returns the month of the year. If today was July 15, 1998, this function would return 7. Normally the date functions are used in Expert Advisors to display the actual date of the commentary. See DayofMonth in this section for an unusual exception.

Moving Average

SYNTAX   mov(DATA ARRAY, PERIODS, METHOD)

This function calculates a moving average of the DATA ARRAY using PERIODS as the length of time. The moving average is calculated by the METHOD specified.

Valid methods are EXPONENTIAL, SIMPLE, TIMESERIES, TRIANGULAR, WEIGHTED, VARIABLE, AND VOLUMEADJUSTED (these can be abbreviated as E, S, T, TRI, W, VAR, and VOL).

This function is used for calculating trigger lines as well as the standard moving averages. It is also used to smooth data so trends are more easily discernible. For additional information on this function, please refer to chapter 1.

Multiplication

SYNTAX   mul(DATA ARRAY, DATA ARRAY)

This function is nothing more than a long way to multiple two values together. Writing:

```
mul( 10, 2 )
```

is exactly the same as writing

```
10 * 2
```

The MetaStock formula language will treat them exactly the same. Feel free to use whichever you are most comfortable with.

Negative

SYNTAX neg(DATA ARRAY)

This function multiplies the value of data array by negative one. Thus any positive number becomes a negative and a negative number is turned positive. For example:

```
Neg( 12 )
```

The above formula could also be written as:

```
12 * -1
```

Use which ever is easiest.

Negative Volume Index

SYNTAX nvi()

This calculates the value of the Negative Volume Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

On Balance Volume

SYNTAX obv()

This calculates the value of the On Balance Volume indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Option Expiration

SYNTAX optionexp()

This function calculates the number of days till the next option expiration date.

Option Life

SYNTAX life(EXPIRATION DATE)

This calculates the value of the Option Life indicator.

It will display the number of days until the date specified. Please note that the expiration data is written in the format year month day, using two digits for each value and no spaces. For example, 990121 refers to January 21, 1999.

Outside

SYNTAX outside()

This will return a +1 whenever an outside day occurs. It will continue to return this same value until either a Rally, Reaction, or an Inside day occur. At any time where Outside() is not equal to +1, it will be equal to 0.

An outside day is defined as any period where the current high is greater than the previous period's high and the current low is less than the previous period's low

Parabolic SAR

SYNTAX sar(STEP, MAXIMUM)

This calculates the value of the Parabolic SAR indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Peak Bars Ago

SYNTAX peakbars(Nth, DATA ARRAY, % MINIMUM CHANGE)

This function calculates how many periods have passed since a peak in the specified DATA ARRAY has occurred. The Nth value is the number of peaks back to count. The % MINIMUM CHANGE is use by the Zig Zag function to calculate when a peak has occurred. For more information on the Zig Zag, please refer to chapter 5.

Peak Value

SYNTAX peak(Nth, DATA ARRAY, % MINIMUM CHANGE)

Using the Zig Zag function, this calculates the value DATA ARRAY had at a peak. How many peaks back to reference is contained in the Nth parameter. %MINIMUM CHANGE is used by the Zig Zag in calculating the peaks and troughs. Form more information on the Zig Zag, please refer to Chapter 5.

Performance

SYNTAX per()

This calculates the value of the Performance indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Plus Directional Movement

SYNTAX pdi(PERIODS)

This calculates the value of the Plus Directional Movement indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Polarized Fractal Efficiency

SYNTAX pfe(DATA ARRAY, PERIODS , SMOOTHING PERIODS)

This calculates the value of the Polarized Fractal Efficiency indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Positive Volume Index

SYNTAX pvi()

This calculates the value of the Positive Volume Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

Power

SYNTAX power(DATA ARRAY, POWER)

This calculates the value of DATA ARRAY raise to a specified POWER. For example:

```
power( 2, 4 )
```

This actually means:

```
2 x 2 x 2 x 2
```

When calculated out, this comes to 16.

This function will normally only be used in complex math calculations like quadratic equations. Please be aware that a negative DATA ARRAY value raised to a non-integer POWER causes an error message to be displayed.

Precision

SYNTAX	prec(DATA ARRAY, PRECISION)

This will drop all the decimal places in DATA ARRAY after the number specified by PRECISION. Small rounding errors may cause some minor distortion in the decimal portion of any number stored in a computer. The effect does not normally show up until seven or more decimal places.

Price Channel High

SYNTAX	pricechannelhigh(PERIODS)

This calculates the value of the top line of the Price Channel indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Price Channel Low

SYNTAX	pricechannellow(PERIODS)

This calculates the value of the bottom line of the Price Channel indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Price Oscillator

SYNTAX	oscp(PERIODS, PERIODS, MA_METHOD, DIFF_METHOD)

This function calculates the predefined Price Oscillator indicator of using moving averages of the two PERIODS specified. The method for expressing differences between these two averages is set in the DIFF_METHOD parameter. Valid DIFF_METHODs are PERCENT and POINTS (these can be abbreviated as % and $).

The moving averages can be of several different types. The exact type desired is specified by MA_METHOD. Valid MA_METHODs are SIMPLE, EXPONENTIAL, WEIGHTED, TIMESERIES, TRIANGULAR, and VARIABLE (these can be abbreviated as S, E, W, T, TRI, VAR).

Price Volume Trend

SYNTAX	pvt()

This calculates the value of the Price Volume Trend indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Projection Band Bottom

SYNTAX Projbandbot(PERIODS)

This calculates the value of the bottom line of the Projection Band indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Projection Band Top

SYNTAX Projbandbot(PERIODS)

This calculates the value of the top line of the Projection Band indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Projection Oscillator

SYNTAX Projosc(REGRESSION PERIODS, SLOWING PERIODS)

This calculates the value of the Projection Oscillator indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Put/Call Price

SYNTAX option(TYPE, DATE, PRICE, INTEREST, DIVIDEND)

This calculates the Put/Call Price indicator. It is not normally used in formula except to compare the value of this indicator with some other value.
TYPE specifies whether the underlying security is an Equity or a Future (i.e., E or F). It also specifies whether the option is a Put or a Call (i.e., P or C). Valid types are EC, EP, FC, and FP. (These can be spelled out as CALL, PUT, FUTURECALL, and FUTUREPUT.)

DATE refers to the date that the option expires. The date must be formatted as YYMMDD. For example, December 31, 1996, should be entered as 961231.

The PRICE parameter specifies the option's strike price.

The INTEREST parameter specifies a "risk free" market interest rate (e.g., 8.75). The yield on a 3 or 6 month T-Bill is usually a good choice.

The DIVIDEND parameter specifies the total dividends received over the last 12 months.

An example of this function would be:

```
Option( EC, 961231, 125, 8.5, 6.31 )
```

This would calculate the fair market value of an equity call that matures on December 31, 1996, at a strike price of $125. The current market interest rates are 8.5% and the security paid an annual dividend of $6.31.

Qstick

SYNTAX   qstick(PERIODS)

This calculates the value of the Qstick indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

r-squared

SYNTAX   rsquared(DATA ARRAY, PERIODS)

This calculates the value of the r-squared indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Rally

SYNTAX   rally()

This will return a +1 whenever a rally occurs. It will continue to return this same value until either a Reaction, an Inside day, or an Outside day occur. At any time where Rally() is not equal to +1, it will be equal to 0.

A rally day occurs when today's high is greater than the previous day's high and today's low is greater than or equal to the previous day's low.

Rally With Volume

SYNTAX   rallywithvol()

This will return a +1 whenever a rally with volume occurs. It will continue to return this same value until either a Reaction, an Inside day, or an Outside day occur. At any time where Rally() is not equal to +1, it will be equal to 0.

A rally with volume occurs when today's high is greater than the previous day's high and today's low is greater than or equal to the previous day's low. Today's volume must also be greater than the previous day's volume.

Random Walk Index of Highs

SYNTAX rwih(PERIODS)

This calculates the value of the Random Walk Index of the Highs indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Random Walk Index of Lows

SYNTAX rwil(PERIODS)

This calculates the value of the Random Walk Index of the Lows indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Range Indicator

SYNTAX rangeindicator(PERIODS, SMOOTHING PERIODS)

This calculates the value of the Range Indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Rate of Change

SYNTAX roc(DATA ARRAY, PERIODS, DIFF_METHOD)

This function calculates the change in DATA ARRAY over PERIODS amount of time. The result can be expressed by either of two methods. Valid DIFF_METHODs are PERCENT and POINTS (these can be abbreviated as % and $). For more information on this funciton, refer to Chapter 1.

Reaction

SYNTAX reaction()

This will return a +1 whenever a reaction occurs. It will continue to return this same value until either a Rally, an Inside day, or an Outside day occur. At any time where Reaction() is not equal to +1, it will be equal to 0.

A reaction day occurs when today's high is less than or equal to the previous day's high and today's low is less than the previous day's low.

Reaction With Volume

SYNTAX reactionwithvol()

This will return a +1 whenever a reaction with volume occurs. It will continue to return this same value until either a Rally, an Inside day, or an Outside day occur. At any time where Reaction() is not equal to +1, it will be equal to 0.

A reaction with volume occurs when today's high is less than or equal to the previous day's high and today's low is less than the previous day's low. Today's volume must also be greater than the previous day's volume.

Reference

SYNTAX ref(DATA ARRAY, PERIODS)

This returns the value of DATA ARRAY on a previous or subsequent period. The time to be accessed is expressed by PERIODS. A negative number looks back in time while a positive number looks into the future. For example

```
Ref( C, -1 )
```

This function looks at the previous periods closing price.

Please be aware that using a positive value results in a formula that is not calculable on the final periods of a chart. If there is not sufficient data in the direction being referenced, the formula will not be calculated for that period of time. Therefore, explorations and system test that use reference with positive numbers will not work.

A positive number of PERIODS should normally only be used in pattern recognition formulas and even then are not suggested for explorations or system tests. For more information on Reference, please refer to chapter 1.

Relative Momentum Index

SYNTAX rmi(DATA ARRAY, PERIODS, MOMENTUM PARAMETER)

This calculates the value of the Relative Momentum Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

Relative Strength Index (RSI)

SYNTAX rsi(PERIODS)

This calculates the value of the Relative Strength Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

Relative Volatility Index

SYNTAX rvi(PERIODS)

This calculates the value of the Relative Volatility Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

Round

SYNTAX round(DATA ARRAY)

This rounds DATA ARRAY to the nearest integer (i.e., no decimal places). Any number with decimal value of .5 or higher is rounded up. All others are rounded down.

Security Data

SYNTAX security("SYMBOL",DATA ARRAY)

This extremely useful function is only available in MetaStock Professional, version 7.0 or later. It allows you to reference any value from another securities data. You can also nest other functions in this one, allowing you to get the values for indicators calculated on other securities. This function is more fully discussed in Chapter 5. Please look there for more details on its use.

Selling Pressure

SYNTAX sellp()

This calculates the value of the selling presure component of the Demand Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

Sine

SYNTAX sin(DATA ARRAY)

Calculates the geometric function Cosine. This is a cyclic line alternating between 0 and 1 base on the angle value used for the DATA ARRAY. Any number larger that 360, has 360 subtracted from it until it is between 1 and 360 and then the function is calculated.

Normally this function is not used in formulas but it is useful to plot or watch a cycle on a chart. For example:

```
Sin( Cum( 1 ) )
```

With this function, 1 will be added constantly and each period the Sine() plot will continue in its cycle. After 360 periods, it will be back to its original value, thus marking a 360 period cycle on your chart or in your formula.

To work with smaller time periods, increase the number in the Cum() function so it reaches 360 sooner. The exact number to use can be found by the equation:

360 / number of periods in cycle

You could even put it in your formula that way. For example, if the security had a 17 period cycle, you could write:

```
Sin( Cum( 360 / 17 ) )
```

Square Root

SYNTAX sqrt(DATA ARRAY)

This calculates the square root of DATA ARRAY. The square root of a negative number always returns a zero result.

Standard Deviation

SYNTAX stdev(DATA ARRAY, PERIODS)

This calculates the value of the Standard Deviation indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Standard Error

SYNTAX ste(DATA ARRAY, PERIODS)

This calculates the value of the Standard Error inidicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Standard Error Band Bottom

SYNTAX	stebandbot(DATA ARRAY, PERIODS, ERRORS)

This calculates the value of the bottom Standard Error Band of the specified DATA ARRAY. The result is shifted downward by the number of standard errors specified by ERRORS

Standard Error Band Top

SYNTAX	stebandbot(DATA ARRAY, PERIODS, ERRORS)

This calculates the value of the top Standard Error Band of the specified DATA ARRAY. The result is shifted upward by the number of standard errors specified by ERRORS.

Stochastic Momentum Index

SYNTAX	stochmomentum(PERIODS, SMOOTHING, DOUBLE SMOOTHING)

This calculates the value of the Stochastic Momentum Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

Stochastic Oscillator

SYNTAX	stoch(%K PERIODS, %K SLOWING)

This calculates the value of the Stochastic Oscillator indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Subtraction

SYNTAX	sub(DATA ARRAY, DATA ARRAY)

This function is the long way to subtract one value from another. Writing:

```
sub( 10, 2 )
```

is exactly the same as writing

```
10 - 2
```

The MetaStock formula language will treat them exactly the same. Feel free to use whichever you are most comfortable with.

Summation

SYNTAX sum(DATA ARRAY, PERIODS)

This function totals the values for DATA ARRAY over the last PERIODS number of records. It is important to note that this result includes the records for the date it is being calculated for. For more information on this function, please refer to Chapter 4.

Swing Index

SYNTAX swing(LIMIT MOVE)

This calculates the value of the Swing Index. It is not normally used in formulas except to compare the value of this indicator with some other value. Please note that the Swing Index requires opening prices

Tema

SYNTAX tema(DATA ARRAY, PERIODS)

This calculates the value of the Tema indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Theta

SYNTAX theta(TYPE, DATE, PRICE, INTEREST, DIVIDEND)

This calculates the value of the Theta indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

TYPE specifies whether the underlying security is an Equity or a Future (i.e., E or F). It also specifies whether the option is a Put or a Call (i.e., P or C). Valid types are EC, EP, FC, and FP. (These can be spelled out as CALL, PUT, FUTURECALL, and FUTUREPUT.)

DATE refers to the date that the option expires. The date must be formatted as YYMMDD. For example, December 31, 1996, should be entered as 961231.

The PRICE parameter specifies the option's strike price.

The INTEREST parameter specifies a "risk free" market interest rate (e.g., 8.75). The yeild on a 3 or 6 month T-Bill is usually a good choice.

The DIVIDEND parameter specifies the total dividends received over the last 12 months.

An example of this function would be:

```
theta( EC, 961220, 125, 7.50, 4.75 )
```

This would calculate the theta of a call option on an equity with an expiration date of December 20, 1996 and a strike price of 125. The interest rate is 7.50 percent and the equity has paid $4.75 in the last 12 months.

Time Series Forecast

SYNTAX tsf(DATA ARRAY, PERIODS)

This calculates the value of the Time Series Forecast indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Trade Volume Index

SYNTAX tvi(MINIMUM TICK)

This calculates the value of the Trade Volume Index. It is not normally used in formulas except to compare the value of this indicator with some other value.

TRIX

SYNTAX trix(PERIODS)

This calculates the value of the TRIX indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Trough Bars Ago

SYNTAX troughbars(Nth, DATA ARRAY, % MINIMUM CHANGE)

This function calculates how many periods have passed since a trough in the specified DATA ARRAY has occurred. The Nth value is the number of troughs back to count. The % MINIMUM CHANGE is use by the Zig Zag function to calculate when a trough has occurred. For more information on the Zig Zag, please refer to chapter 5.

Trough Value

SYNTAX trough(Nth, DATA ARRAY, % MINIMUM CHANGE)

Using the Zig Zag function, this calculates the value DATA ARRAY had at a trough. How many troughs back to reference is contained in the Nth parameter. %MINIMUM CHANGE is used by the Zig Zag in calculating the peaks and troughs. For more information on the Zig Zag, please refer to Chapter 5.

Typical Price

SYNTAX typical()

This calculates the value of the Typical Price indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Ultimate Oscillator

SYNTAX ult(CYCLE1, CYCLE2, CYCLE3)

This calculates the value of the Ultimate Oscillator indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

The parameters are three cycle lengths, each of which must be larger than the previous. In other words, the following example will not work:

```
Ult ( 10, 7, 5 )
```

However, if the parameters were rearranged as follows, the function would calculate correctly

```
Ult ( 5, 7, 10 )
```

Using cycle lengths of 7, 14, and 21 returns the default Ultimate Oscillator.

Value When

SYNTAX valuewhen (Nth, EXPRESSION, DATA ARRAY)

This looks at all the data in the chart and evaluates when the EXPRESSION was true. Then starting from the most recent, it counts back Nth times and reports the value of DATA ARRAY. For example:

```
ValueWhen( 1, Cross( -20, Willr(14), C )
```

This function returns the value of the close on the most recent occurrence of the Williams %R falling below -20

Variance

SYNTAX var(DATA ARRAY, PERIODS)

This function examines the DATA ARRAY and calculates the statistical variance over PERIODS number of records.

Vega

SYNTAX vega(TYPE, DATE, PRICE, INTEREST, DIVIDEND)

This calculates the value of the Vega indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

TYPE specifies whether the underlying security is an Equity or a Future (i.e., E or F). It also specifies whether the option is a Put or a Call (i.e., P or C). Valid types are EC, EP, FC, and FP. (These can be spelled out as CALL, PUT, FUTURECALL, and FUTUREPUT.)

DATE refers to the date that the option expires. The date must be formatted as YYMMDD. For example, December 31, 1996, should be entered as 961231.

The PRICE parameter specifies the option's strike price.

The INTEREST parameter specifies a "risk free" market interest rate (e.g., 8.75). The yeild on a 3 or 6 month T-Bill is usually a good choice.

The DIVIDEND parameter specifies the total dividends received over the last 12 months.

An example of this function would be:

```
.
Vega( EC, 961220, 125, 7.50, 4.75 )
```

This would calculate the vega of a call option on an equity with an expiration date of December 20, 1996 and a strike price of 125. The interest rate is 7.50 percent and the equity has paid $4.75 in the last 12 months.

Vertical Horizontal Filter

SYNTAX vhf(DATA ARRAY, PERIODS)

This calculates the value of the Vertical Horizonal Filter indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Volatility, Chaikin's

SYNTAX vol(MA PERIODS, ROC PERIODS)

This calculates the value of the Chaikin’s Volatility indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Volatility, Option

SYNTAX volo()

This calculates the value of the Option Volatility indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Volume Oscillator

SYNTAX oscv(PERIODS, PERIODS, MA_METHOD, DIFF_METHOD)

This calculates the value of the Volume Oscillator indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Two separate time periods are required for the moving averages used by this function. A moving average method is also required, as is the method of displaying the results.

Valid MA_METHODs are SIMPLE, EXPONENTIAL, WEIGHTED, TIMESERIES, TRIANGULAR, and VARIABLE (these can be abbreviated as S, E, W, T, TRI, and VAR).

Valid DIFF_METHODs are PERCENT and POINTS (these can be abbreviated as % and $).

Weighted Close

SYNTAX wc()

This calculates the value of the Weighted Close indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Wilder's Smoothing

SYNTAX wilders(DATA ARRAY, PERIODS)

This calculates the value of the Wilder’s Smoothing indicator. This is very similar to the standard exponential moving average and can be used anywhere that type f moving average is desired. This function also has to be used anytime one of Wilder’s custom formulas are being written.

Williams' %R

SYNTAX willr(%R PERIODS)

This calculates the value of the William’s %R indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Williams' A/D

SYNTAX willa()

This calculates the value of the Williams’ A/D indicator. It is not normally used in formulas except to compare the value of this indicator with some other value.

Writeif

SYNTAX WriteIf(EXPRESSION, "TRUE TEXT", "FALSE TEXT")

This function can only be used within the expert advisor as part of a commentary. It functions just like a regular If() function except it displays text instead of a value. For more information of the If() function, please refer to chapter 2.

Expert Advisors are not discussed in this text. For more information on writing commentaries, refer to you MetaStock manual.

Writeval

SYNTAX writeval(DATA ARRAY)

This function can only be used within the expert advisor as part of a commentary. It will display the contents of DATA ARRAY as text.

Expert Advisors are not discussed in this text. For more information on writing commentaries, refer to you MetaStock manual.

Year

SYNTAX year()

This function will return the value of the year in a four digit format. I.e., if the date this function was calculated on was October, 15 1996, the result would be 1996.

Zig Zag

SYNTAX zig(DATA ARRAY, MINIMUM CHANGE, DIFF_METHOD)

This calculates the value of the Zig Zag indicator on the DATA ARRAY specified. MINIMUM CHANGE is the amount required for a reversal and DIFF_METHOD is how the change is to be calculated.

Valid DIFF_METHODs are PERCENT and POINTS (these can be abbreviated as % and $).

Candlestick Functions

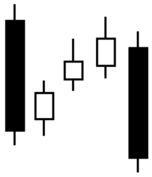
The Candlestick functions allow you to locate specific Japanese Candlestick patterns. Each function plots a "+1" when the pattern is found; otherwise a "0" is plotted.
The brief interpretations provided for the patterns shown on the following pages were taken from two documents written by Steve Nison. For additional information on Japanese Candlesticks, please refer to Mr. Nison's documents: Japanese Candlestick Charting Techniques and Beyond Candlesticks.

You should also be aware that candlestick pattern recognition is subjective. What one person calls a "Big Black Candle" may not qualify as such for someone else. To find these patterns on a chart, MetaStock must rely on predefined rules. These rules were defined based upon experience and with Steve Nison's assistance.

Important: When using the Candlestick functions in an exploration, you must choose the "Load __ Records" button in the Explorer Options dialog and specify at least "10"; otherwise the exploration results may be inaccurate.

Bearish 3 Method Formation

SYNTAX bear3formation()



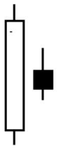
PATTERN

A long black body followed by three small, usually white, bodies and another long black body. The three white bodies are contained within the first black body's range.

INTERPRETATION A bearish continuation pattern.

Bearish Harami

SYNTAX bearharami()



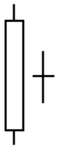
PATTERN

A small black body is contained within an unusually large white body.

INTERPRETATION A bearish pattern when preceded by an uptrend.

Bearish Harami Cross

SYNTAX bearharamicross()



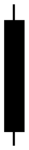
PATTERN

A Doji contained within a large white body.

INTERPRETATION A top reversal signal.

Big Black Candle

SYNTAX bigblack()



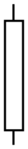
PATTERN

An unusually long black body with a wide range between high and low, and prices open near the high and close near the low.

INTERPRETATION A bearish pattern.

Big White Candle

SYNTAX bigwhite()



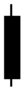
PATTERN

An unusually long white body with a wide range between high and low, and prices open near the low and close near the high.

INTERPRETATION A bullish pattern.

Black Body

SYNTAX black()



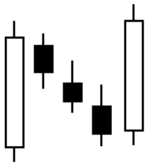
PATTERN

A candlestick formed when the closing price is lower than the opening price.

INTERPRETATION A bearish signal. More important when part of a pattern.

Bullish 3 Method Formation

SYNTAX bull3formation()



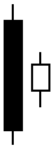
PATTERN

A long white body followed by three small, usually black, bodies and another long white body. The three black bodies are contained within the first white body's range.

INTERPRETATION A bullish continuation pattern.

Bullish Harami

SYNTAX bullharami()



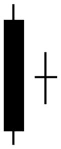
PATTERN

A small white body is contained within an unusually large black body.

INTERPRETATION A bullish pattern when preceded by a downtrend..

Bullish Harami Cross

SYNTAX bullharamicross()



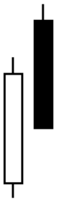
PATTERN

A Doji contained within a large black body.

INTERPRETATION A bottom reversal signal.

Dark Cloud Cover

SYNTAX darkcloud()



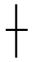
PATTERN

A long white candlestick is followed by a black candlestick. The black candlestick opens above the white candlestick's high and closes well into the white candlestick's body.

INTERPRETATION A bearish reversal signal during an uptrend.

Doji

SYNTAX doji()



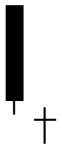
PATTERN

The open and close are the same.

INTERPRETATION Doji lines are usually components of many important candlestick patterns.

Doji Star

SYNTAX dojistar()



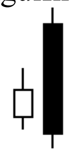
PATTERN

A Doji which gaps above or below a white or black candlestick.

INTERPRETATION A reversal signal with confirmation during the next trading day.

Engulfing Bearish Line

SYNTAX engulfingbear()



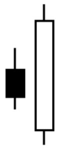
PATTERN

A small white body followed by and contained within a large black body.

INTERPRETATION A major top reversal signal.

Engulfing Bullish Line

SYNTAX engulfingbull()



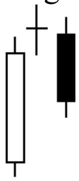
PATTERN

A small black body followed by and contained within a large white body.

INTERPRETATION A major bottom reversal signal.

Evening Doji Star

SYNTAX eveningdojistar()



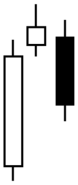
PATTERN

A large white body followed by a doji that gaps above the white body. The third candlestick is a black body that closes well into the white body.

INTERPRETATION A major top reversal signal, more bearish than the regular evening star pattern because of the Doji.

Evening Star

SYNTAX eveningstar()



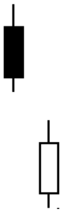
PATTERN

A large white body followed by a small body (white or black) that gaps above the white body. The third candlestick is a black body that closes well into the white body.

INTERPRETATION A major top reversal signal.

Falling Window

SYNTAX fallingwindow()



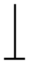
PATTERN

A window (i.e., gap) between the low of the first candlestick and the high of the second candlestick. This produces the same results as the Gap Down function (see page 91).

INTERPRETATION A rally to the window is highly probable. The window should provide resistance.

Gravestone Doji

SYNTAX gravestonedoji()



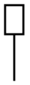
PATTERN

The open and close are at the low of the period.

INTERPRETATION A market top reversal signal. The longer the upper shadow the more bearish the signal.

Hammer

SYNTAX hammer()

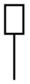
PATTERN 

A small body (white or black) near the high with a long lower shadow with little or no upper shadow.

INTERPRETATION A bullish pattern during a downtrend.

Hanging Man

SYNTAX hangingman()

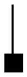
PATTERN 

A small body (white or black) near the high with a long lower shadow with little or no upper shadow. The lower shadow should be two or three times the height of the body.

INTERPRETATION A bearish pattern during an uptrend.

Inverted Black Hammer

SYNTAX invblackhammer()

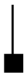
PATTERN 

An upside-down hammer with a black body.

INTERPRETATION A bottom reversal signal with confirmation the next trading day.

Inverted Hammer

SYNTAX invhammer()

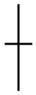
PATTERN 

An upside-down hammer (white or black).

INTERPRETATION A bottom reversal signal with confirmation the next trading day.

Long Legged Doji

SYNTAX longleggeddoji()

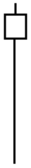
PATTERN 

A Doji pattern with very long upper and lower shadows.

INTERPRETATION A market top reversal signal.

Long Lower Shadow

SYNTAX longlowershadow()



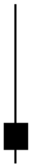
PATTERN

A candlestick (black or white) with a lower shadow that has a length 2/3 or more of the total range of the candlestick.

INTERPRETATION A bullish signal, particularly when around price support levels.

Long Upper Shadow

SYNTAX longuppershadow()



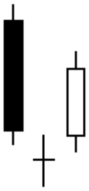
PATTERN

A candlestick (black or white) with an upper shadow that has a length 2/3 or more of the total range of the candlestick.

INTERPRETATION A bearish signal, particularly around price resistance levels.

Morning Doji Star

SYNTAX morningdojistar()



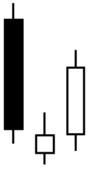
PATTERN

A large black body followed by a doji that gaps below the black body. The third candlestick is a white body that closes well into the black body.

INTERPRETATION A major bottom reversal signal, more bullish than the regular morning star pattern because of the Doji.

Morning Star

SYNTAX morningstar()



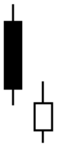
PATTERN

A large black body followed by small body (white or black) that gaps below the black body. The third candlestick is a white body that closes well into the black body.

INTERPRETATION A major bottom reversal signal.

On Neck-Line

SYNTAX onneckline()



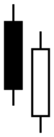
PATTERN

A black candlestick in a downtrend followed by a small white candlestick with its close near the low of the black candlestick.

INTERPRETATION A bearish pattern where the market should move lower when the white candlestick's low is penetrated.

Piercing Line

SYNTAX piercingline()



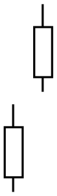
PATTERN

A black candlestick followed by a white candlestick that opens lower than the black candlestick's low, but closes more than halfway into the black body.

INTERPRETATION A bottom reversal signal.

Rising Window

SYNTAX risingwindow()



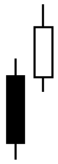
PATTERN

A window (i.e., gap) between the high of the first candlestick and the low of the second candlestick. This produces the same results as the Gap Up function (see page 91).

INTERPRETATION A selloff to the window is highly probable. The window should provide support.

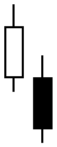
Separating Lines

SYNTAX separatinglines()



PATTERN

In an uptrend, a black candlestick is followed by a white candlestick with the same opening price.

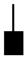


In a downtrend, a white candlestick is followed by a black candlestick with the same opening price.

INTERPRETATION A continuation pattern. The prior trend should resume.

Shaven Bottom

SYNTAX shavenbottom()



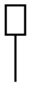
PATTERN

A candlestick (white or black) with no lower shadow.

INTERPRETATION See Inverted Hammer (see page 125) interpretation.

Shaven Head

SYNTAX shavenhead()



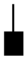
PATTERN

A candlestick (white or black) with no upper shadow.

INTERPRETATION See Hammer (see page 125) and Hanging Man (see page 125) interpretations.

Shooting Star

SYNTAX shootingstar()



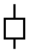
PATTERN

A candlestick (white or black) with a small body, long upper shadow, and little or no lower shadow.

INTERPRETATION A bearish pattern in an uptrend.

Spinning Top

SYNTAX spinningtop()

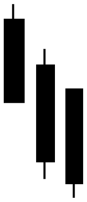
PATTERN 

A candlestick (white or black) with a small body. The size of the shadows is not important.

INTERPRETATION A neutral pattern. Spinning tops are more important when part of other formations.

Three Black Crows

SYNTAX 3blackcrows()

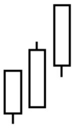
PATTERN 

Three long black candlesticks with consecutively lower closes that close near or at their low prices.

INTERPRETATION A top reversal signal.

Three White Soldiers

SYNTAX 3whitesoldiers()

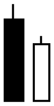
PATTERN 

Three white candlesticks with consecutively higher closes that close near or at their high prices.

INTERPRETATION A bottom reversal signal.

Tweezer Bottoms

SYNTAX tweezerbottoms()

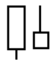
PATTERN 

Two or more candlesticks with matching bottoms. The size or color of the candlestick does not matter. The candlesticks do not have to be consecutive.

INTERPRETATION Minor reversal signal that is more important when the candlesticks form another pattern.

Tweezer Tops

SYNTAX tweezertops()

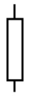
PATTERN 

Two or more candlesticks with matching bottoms.

INTERPRETATION Minor reversal signal that is more important when the candlesticks form another pattern. The candlesticks do not have to be consecutive.

White Body

SYNTAX white()

PATTERN 

A candlestick formed when the closing price is higher than the opening price.

INTERPRETATION A bullish signal. More important when part of a pattern.

Answers to Review Questions

Chapter 1:

1. If your indicator matched the plot of the indicator you were trying to duplicate, you got this one correct

2. This is quick way to remember the order of precedence: Multiplication, Division, Addition, Subtraction.

3. (5 - 4) * (3 / (2 - 1))

4. Mov(C, 21, S) * 1.2

5. They would all plot the exact same thing, The closing price of the base security of the chart

Chapter 2:

1. All three would be true. Remember, any number other than zero is considered to be true.

2. False. The opening price equals 28 but is not greater than 28.

3. 1) True
   2) False
   3) True
   4) True

4. All If()s have a condition, a data array to return for true results and a data array to return for false results

Chapter 3:

1. All of them can. The MACD() and the If() function both return data arrays. C is the closing prices which is already a data array. The number 10 is a data array of a single value i.e. a constant.

2. Formula 1 references formula 2 which references formula 1. This circular reference will not work and causes errors.

3. False, formulas that do not use the Fml() function run faster than those that do. However, sometimes the benefits of the Fml() function outweigh this drawback.

4. To ensure the name of the formula exactly matches what metastock sees, the Fml() function should be entered using the Paste Function window.

5. False, any function can be nested in any place where a data array is accepted, however, not all functions accept data arrays

Chapter 4:

1. A straight line that increases in value by one for each period of data in the chart.

2. If(Close < Open, Close, Open)

3. False, the Cross() function only returns a 1 when the first data array rises above the second. All other times, it returns a 0

4. True, with the Alert() function, you can create a condition that is true if something occurs today and something else has already occurred within the past few periods.

5. True, the Zigzag() function allows you to examine data for general trends and patterns. It is however strongly advised not to use it for buy and sell signals.

Chapter 5:

1. This symbol means that another formula is going to follow. The semicolon is most often used after declaring a variable but also when multiple formulas are put in one indicator. If used at the end of the last formula in an indicator, MetaStock will return an error.

2. Close

3. False. While you can plot both lines with one custom indicator, the lines will both be the same color, style and thickness. All formulas in a single custom indicator have the same appearance when plotted and can not be changed individually.

4. Yes, just like the Fml() function, the Fmlvar() is best entered through the Paste Function window. Referenced variables will also be accessed a little slower that if they were written in the formula in to begin with.

5. The p variable take the value of whatever plot was selected when its indicator is inserted into the chart.

Quick Reference

Function Name	Syntax
Absolute Value	abs(DATA ARRAY)
Accumulation/Distribution	ad()
Accumulation Swing Index	aswing(LIMIT MOVE)
Addition	add(DATA ARRAY, DATA ARRAY)
Alert	alert(EXPRESSION, PERIODS)
Arc Tangent	atan(Y DATA ARRAY, X DATA ARRAY)
Aroon Down	aroondown(PERIODS)
Aroon Up	aroonup(PERIODS)
Average Directional Movement	adx(PERIODS)
Average True Range	Atr(PERIODS)
Bars Since	barssince(DATA ARRAY)
Bollinger Band Bottom	bbandbot(DATA ARRAY, PERIODS, METHOD, DEVIATIONS)
Bollinger Band Top	bbandtop(DATA ARRAY, PERIODS, METHOD, DEVIATIONS)
Buying Pressure	buyp()
Ceiling	ceiling(DATA ARRAY)
Chaikin A/D Oscillator	co()
Chaikin's Money Flow	cmf(PERIODS)
Chande Momentum Oscillator	cmo(DATA ARRAY, PERIODS)
Commodity Channel Index (EQUIS)	ccie(PERIODS)
Commodity Channel Index (Standard)	cci(PERIODS)
Commodity Selection Index	csi(PERIODS, VALUE, MARGIN, COMMISSION)
Correlation Analysis	correl(INDEPENDENT, DEPENDENT, PERIODS, SHIFT)
Cosine	cos(DATA ARRAY)
Cross	cross(DATA ARRAY1, DATA ARRAY2)
Cumulate	cum(DATA ARRAY)
Day Of Month	dayofmonth()
Day Of Week	dayofweek()
Delta	delta(TYPE, DATA, PRICE, INTEREST, DIVIDEND)
Dema	dema(DATA ARRAY, PERIODS)
Detrended Price Oscillator	dpo(PERIODS)
Directional Movement Index	dx(PERIODS)
Divergence	divergence(DATA ARRAY1, DATA ARRAY2, % MINIMUM CHANGE)
Division	div(DATA ARRAY, DATA ARRAY)
Dynamic Momentum Index	dmi(DATA ARRAY)
Ease of Movement	emv(PERIODS, METHOD)

Exponent	exp(DATA ARRAY)
Fast Fourier Transform	fft(DATA ARRAY, PERIODS, LENGTH, DETREND or MEAN, AMPLITUDE or POWER)
Floor	floor(DATA ARRAY)
Forecast Oscillator	forecastosc(DATA ARRAY, PERIODS)
Formula Call	fml(“FORMULA_NAME”)
Formula Variable Call	fmlvar(“FORMULA_NAME”, “VARIABLE_NAME”)
Fraction	frac(DATA ARRAY)
Gamma	gamma(TYPE, DATE, PRICE, INTEREST, DIVIDEND)
Gap Down	gapdown()
Gap Up	gapup()
Herrick Payoff Index	hpi(CENTS, MULTIPLYING FACTOR)
Highest	highest(DATA ARRAY)
Highest Bars Ago	highestbars(DATA ARRAY)
Highest High Value	hhv(DATA ARRAY, PERIODS)
Highest High Value Bars Ago	hhvbars(DATA ARRAY, PERIODS)
Highest Since	highestsince (Nth, EXPRESSION, DATA ARRAY)
Highest Since Bars Ago	highestsincebars(Nth, EXPRESSION, DATA ARRAY)
If	if(EXPRESSION, TRUE DATA ARRAY, FALSE DATA ARRAY)
Inertia	inertia(REGRESSION PERIODS, RVI PERIODS)
Input	input(“PROMPT TEXT”, MINIMUM VALUE, MAXIMUM VALUE, DEFAULT VALUE)
Inside	inside()
Integer	int(DATA ARRAY)
IntraDay Momentum Index	imi(PERIODS)
Klinger Volume Oscillator	kvo ()
Last Value in Data Array	lastvalue(DATA ARRAY)
Linear Regression Indicator	linearreg(DATA ARRAY, PERIODS)
Linear Regression Slope	linregslope(DATA ARRAY, PERIODS)
Logarithm (natural)	log (DATA ARRAY)
Lowest	lowest (DATA ARRAY)
Lowest Bars Ago	lowestbars(DATA ARRAY)
Lowest Low Value	llv(DATA ARRAY, PERIODS)
Lowers Low Value Bars Ago	llvbars(DATA ARRAY, PERIODS)
Lowest Since	lowestsince(Nth, EXPRESSION, DATA ARRAY)
Lowest Since Bars Ago	lowestsincebars(Nth, EXPRESSION, DATA ARRAY)
MACD	macd()
Market Facilitation Index	marketfacindex()
Mass Index	mass (PERIODS)
Maximum	max(DATA ARRAY 1, DATA ARRAY 2)
Median Price	mp()
MESA Lead Sine	mesaleadsine(CYCLE LENGTH)
MESA Sine Wave	mesasinewave(CYCLE LENGTH)
Midpoint	mid(DAA ARRAY, PERIODS)

Minimum	min(DATA ARRAY, DATA ARRAY)
Minus Direction Movement	mdi(PERIODS)
Modulus	mod(DATA ARRAY, DATA ARRAY)
Momentum	mo(PERIODS)
Money Flow Index	mdi(PERIODS)
Month	month()
Moving Average	mov(DATA ARRAY, PERIODS, METHOD)
Multiplication	mul(DATA ARRAY, DATA ARRAY)
Negative	neg(DATA ARRAY)
Negative Volume Index	nvi()
On Balance Volume	obv()
Option Expiration	optionexp()
Option Life	life(EXPIRATION DATE)
Outside	outside()
Parabolic SAR	sar(STEP, MAXIMUM)
Peak Bars Ago	peakbars(Nth, DATA ARRAY, % MINIMUM CHANGE)
Peak Value	peak(Nth, DATA ARRAY, % MINIMUM CHANGE)
Performance	per()
Plus Directional Movement	pdi(PERIODS)
Polarized Fractal Efficiency	pfe(DATA ARRAY, PERIODS, SMOOTHING PERIODS)
Positive Volume Index	pvi()
Power	power(DATA ARRAY, POWER)
Precision	prec(DATA ARRAY, PRECISION)
Price Channel High	pricechannelhigh(PERIODS)
Price Channel Low	pricechannellow(PERIODS)
Price Oscillator	oscp(PERIODS, PERIODS, MA_METHOD, DIFF_METHOD)
Price Volume Trend	pvt()
Projection Band Bottom	projbandbot(PERIODS)
Projection Band Top	projbandtop(PERIODS)
Projection Oscillator	projosc(REGRESSION PERIODS, SLOWING PERIODS)
Put/Call Price	option(TYPE, PRICE, INTEREST, DIVIDEND)
Qstick	qstick(PERIODS)
r-squared	rsquared(DATA ARRAY, PERIODS)
Rally	rally()
Rally With Volume	rallywithvol()
Random Walk Index of Highs	rwih(PERIODS)
Random Walk Index of Lows	rwil(PERIODS)
Range Indicator	rangeindicator(PERIODS, SMOOTHING PERIODS)
Rate of Change	roc(DATA ARRAY, PERIODS, DIFF_METHOD)
Reaction	reaction()
Reaction With Volume	reactionwithvol()
Reference	ref(DATA ARRAY, PERIODS)
Relative Momentum Index	rmi(DATA ARRAY, PERIODS, MOMENTUM PARAMETER)

Relative Strength Index (RSI)	rsi(PERIODS)
Relative Volatility Index	rvi(PERIODS)
Round	round(DATA ARRAY)
Security Data	security("SYMBOL",DATA ARRAY)
Selling Pressure	sellp()
Sine	sine(DATA ARRAY)
Square Root	sqrt(DATA ARRAY)
Standard Deviation	stdev(DATA ARRAY, PERIODS)
Standard Error	ste(DATA ARRAY, PERIODS)
Standard Error Band Bottom	stebandbot(DATA ARRAY, PERIODS, ERRORS)
Standard Error Band Top	stebandtop(DATA ARRAY, PERIODS, ERRORS)
Stochastic Momentum Index	stochmomentum(PERIODS, SMOOTHING, DOUBLE SMOOTHING)
Stochastic Oscillator	stoch(%K PERIODS, %K SLOWING)
Subtraction	sub(DATA ARRAY, DATA ARRAY)
Summation	sum(DATA ARRAY, PERIODS)
Swing Index	swing(LIMIT MOVE)
Tema	tema(DATA ARRAY, PERIODS)
Theta	theta(TYPE, PRICE, INTEREST, DIVIDEND)
Time Series Forecast	tsf(DATA ARRAY, PERIODS)
Trade Volume Index	tvi(MINIMUM TICK)
TRIX	trix(PERIODS)
Trough bars ago	troughbars(Nth, DATA ARRAY, % MINIMUM CHANGE)
Trough Value	trough(Nth, DATA ARRAY, % MINIMUM CHANGE)
Typical Price	typical()
Ultimate Oscillator	ult(CYCLE1, CYCLE2, CYCLE3)
Value When	valuewhen(Nth, EXPRESSION, DATA ARRAY)
Variance	var(DATA ARRAY, PERIODS)
Vega	vega(TYPE, PRICE, INTEREST, DIVIDEND)
Vertical Horizontal Filter	vhf(DATA ARRAY, PERIODS)
Volatility, Chaikin’s	vol(MA PERIODS, ROC PERIODS)
Volatility, Option	volo()
Volume Oscillator	oscv(PERIODS, PERIODS, MA_METHOD, DIFF_METHOD)
Weighted Close	wc()
Wilder’s Smoothing	wilders(DATA ARRAY, PERIODS)
William’s %R	willr(%R PERIODS)
William’s A/D	willa()
Writeif	writeif(EXPRESSION, “TRUE TEXT”, “FALSE TEXT”)
Writeval	writeval(DATA ARRAY)
Year	year()
Zig Zag	zig(DATA ARRAY, MINIMUM CHANGE, DIFF_METHOD)

Glossary

Binary Wave: a line that cycles between two values, usually 0 and 1. This is a basic version of the more advance oscillators.

Constant: a fixed, numerical value.

Data Array: an ordered, sequential grouping of numbers

Formula: a logical arrangement of one or more functions and data arrays.

Function: a command to perform a specific mathematical calculation.

Keyword: a letter or word recognized by MetaStock to either a function command or a data array

Nesting: the practice of inserting one function or command inside another.

Operators: a subset of functions that can be expressed by a single symbol.

Parameter: a piece of information required by a function for it to be calculated. All parameters are listed in an order determined by the function and enclosed by the function's parentheses.

Ziggurat: a structure resembling the pyramids of Egypt, but normally associated with the ancient cultures of South America.

Index