iExpertAdvisor™

How to Test Any Forex Robot

3 Simple, Powerful Tests to Find the *Real* Performance of *Any* Expert Advisor

(Including MQL Code to Automate Your Testing)

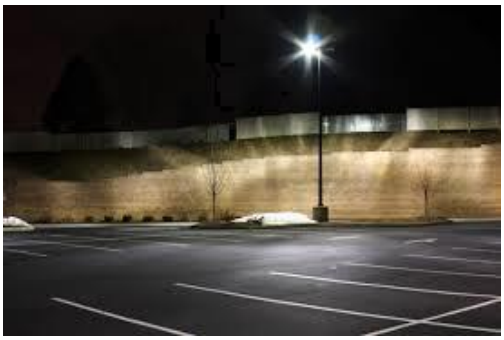
Contents

Parking Lot Science

A drunk man, staggering beneath a bright street light, is searching for his keys in an empty parking lot.

A passing friend stops to help and asks "Where did you last use your keys?" The man pointed to the other side of the lot, which was covered in darkness.

The friend inquires "If you lost them over there, why are looking here?" The drunk man stammers, "Because I can see better over here".



Not a true story (to my knowledge). It's an old parable about the human tendency to avoid the difficult paths in life. Why go into the darkness of the unknown when you can stay in your well-lit comfort zone?

If you're new to trading, you may not yet realize how emotional trading can be. It's very easy to deceive yourself using over-simplified rationalizations, like "I can see better over here ..."

If you have been trading a while, you probably already know that your emotions, your feelings, even your perceptions can't always be trusted. That's why we need tests. *Absolute, unbiased, repeatable tests.* Tests that help us debunk *Parking Lot Science*.

My Path

My name is David Williams. I've been developing **Forex Robots**, or ***Expert Advisors*** (EAs), for over 8 years.

I've built and tested hundreds of EAs. Some for my own personal use, but most for paying, ***and demanding***, clients. Each of these EAs was unique in its own way. But they all had one thing in common. They all required objective testing. Not just test results to measure their performance. I needed actionable test results.

I needed test results that could be used to improve the EA. Or, at the very least, show that EA could not be improved. That it was junk.

So, it was during this time I began creating my own personal methods for testing the quality of *any* Expert Advisor. It took me over 8 years to perfect these tests and I can't count how many times they've saved me from running truly disastrous EAs.

These are the same methods I'm going to share with you today.

How do I *Really* Feel About Back-Testing?

Back-testing is the process of running an EA against historical price data. We feed the price data to the EA. The EA processes each new price and opens, closes or modifies trades.

In the MetaTrader platform there is a built-in back tester named the "Strategy Tester". It's a pretty handy tool.

The only problem is that, well, for the most part, back-testing is nearly ***worthless***!

Now, this is a strange thing to hear from someone who is about to show you tools that rely entirely on back-testing. But, I think it's really important that you keep back-testing in the right perspective.

There are two major problems with back testing. The first is insurmountable. The second is technical in nature.

The first problem with back testing is that the future rarely behaves like the past. *Pretty much, it never does*. Back-testing was more reliable in the past, before computers, because the historical activity of a market was based on "real" market forces - and these forces were at times repeatable. Today, for the most part, that's all been ruined. If there is a significant advantage from a repeatable market force, you can believe it will be exploited by algorithmic traders in short order. (This is not to say there are no predictable opportunities; they just tend to be small and short-lived these days).

In any event, the insurmountable problem with back-testing is that the future is unpredictable. Even if your Expert Advisor performs great when back-testing, there is simply no guarantee that it will perform as well in the future.

The second problem with back-testing is a technical one. Most of us don't have access to quality tick data. "Tick data" is a record of the bid and ask prices for every change in a currency pair's price. As you can imagine, tick data is enormously large. Even with today's inexpensive memory and fast networks, tick data is very difficult to manage.

For most of us, the best we can do with respect to historical data is to use quality one-minute data. One minute data contains the so-called candle data (the open, close, high and low price of a currency pair).

The only drawback of using candle data is that we don't know *when* the high or low of the candle occurred. If the high price was reached *before* the low price, your takeprofit may have triggered. If the low occurred first, your stoploss may have triggered. Since the tester doesn't have this information, the test results are correspondingly inaccurate.

The bottom line*: In practice, I've found that it's generally not worth the effort to store, maintain and update tick data. One-minute data is usually sufficient and is much easier to manage. This is primarily due to the first, insurmountable problem with back-testing – that the future is always uncertain.*

Ok, so now that I've thoroughly bashed back-testing, let me tell you some good news. Back-testing is great for "relative" testing.

What I mean by "relative" testing is this: I want to test an Expert Advisor, make changes to the MQL - or even just change the EA's input parameters, test it again, and then know, ***beyond a shadow of a doubt***, if the changes improved the performance of the EA. These changes in performance are "relative" to the EA's baseline performance.

And more good news. Back-testing is actually great for finding out if an EA is curve-fit. Not sure what curve-fitting is? No problem, I'll explain below.

First, let me introduce you to my techniques and it will all start to make a lot of sense. Trust me.

Divide and Conquer

The best way to test, and improve, the performance of an Expert Advisor is to test the EA's entry and exit ***separately***.

Whenever I build, or modify an EA, my goal is to ***fully optimize*** the performance of the entry before **I even consider the exit**.

It can be difficult to test just the entry.

If you change how an EA opens trades, it has an effect on *when* trades are opened, which of course leads to different entry prices.

The problem here is that the exit is now operating on trades that were opened at different times and at different prices. This completely changes the dynamics of the EA.

You aren't really testing just the entry. You're testing the entry *and* the exit.

The solution to this problem is actually pretty simple. To test just the entry of an EA, we are going to move the stoploss and takeprofit values way out, and then exit every trade at the same exact, fixed time.

For example, consider a swing-type EA. A typical swing-type EA might have a stoploss and a takeprofit of 75 pips and usually closes a trade with 24-48 hours. For this system, we move the stoploss and takeprofit to 1000, and close every trade ***exactly one hour*** after it has opened.

With this test, we make sure the stoploss and takeprofit are not hit (at least not in a normal market) by setting them far outside the normal price range. Then we close the trade quickly after it has opened - before any potential exit-logic would normally close the trade. In this case, "quickly" equates to about an hour. (This is somewhat arbitrary – you can test different exit times, but the idea is that the exit-time is much less than a typical trade's duration.)

We are left with test results that show us how often the EA's entry gets a trade ***started in the right direction***. Note, this is the *only* responsibility of the entry: to get the trade started right. It's not the entry's responsibility to make a profit. (That's the job of the EA's exit.)

Now we need an objective way to measure the results. This is an important part of the test.

We don't look at total profit. All we measure is how many trades were profitable when they were closed. It doesn't matter *how* profitable. Anything above zero is considered profitable.

We divide the number of profitable trades by the total number of trades opened. For example, if 100 total trades were opened, and 65 were profitable, the percentage would be:

65/100 = 65%

(Incidental, 65% is not too bad. I usually shoot for 70% or more, but it's not always possible.)

You definitely want to be over 50%. If the entry's winning percentage is at, or near, 50%, the entry logic is ***no better than random***.

This test allows you to make massive or minor changes to your entry logic and find out if the changes improve the entry. It's objective and it is easy to repeat.

I've included the MQL code to run this test on any EA for which you have the MQL code. (The name of the MQL function is ***fnBarExitMql***).

If you have a black-box EA (an EA for which you don't have the MQL source code) you can still run this test - if the EA has any inputs.

If the EA has inputs, you vary the input values (one at a time), run the EA on a back-test and then manually record the profit of each trade at different set times. The manual test is not as powerful as the automatic test because the trades are not actually closed and this affects the dynamic of the EA. However, the results can still be used for a relative comparison with the EA's baseline results.

The Exit Test

Now that we have a method to objectively test our Entry logic we can focus on the EA's exit.

As I said above, the responsibility of the EA's entry is simply to get the trade started in the right direction. It's the job of the exit to make the money. ***Most of your time and focus should be on your EA's exit logic.***

This test, like the entry test, is pretty simple. My exit test measures the price action for the entire length of every trade, finds the highest possible profit and compares it to the actual closed profit.

So for each trade, we find the point in time when the trade was the most profitable, and then compare that to the actual exit price.

For example, if a trade's highest profit was $200.00, and the trade exited sometime later for a profit of $150, we measure the *exit-quality* of the EA as:

150/200 = 75%

If a trade exits at its highest possible profit, then the quality of the exit is 100%. Of course, this is the percentage we want for each trade. The exit test averages the *exit-quality* of every trade and produces the ***overall*** *exit-quality* of the EA. The ***overall*** *exit-quality is* the metric that you watch and compare as you make changes to the exit logic of the EA.

I've included the MQL code for my exit function. (The name of the function is ***fnExitTestMql***).

If you want to test an EA that you do not have the MQL code for, you manually record all of this information by viewing the results of a back-test. It's tedious, but it's possible.

Curve Fitting

The following is an excerpt from a blog post I wrote about curve-fitting.

Basically, to *curve-fit* means to create an EA that works for a specific time period. It's not that hard to do. As matter of fact you can do it to yourself without even realizing it!

Luckily, there's a way to find out.

Robustness is a term used when designing a *control-system*.

(An example of a simple *control system* is the thermostat in your house: when the temperature drops, the heat is turned on. When the temperature rises, the heat is turned off).

Anyway, *robustness* defines the *stability* of a control system. Think of it this way: A stable system acts like a marble inside a bowl. If you gently shake the bowl, the marble will move for a little while, but it will eventually stop and settle at the bottom of the bowl.

An unstable system acts like a marble on the top of an upside down bowl. If you gently move the bowl the marble will fall off and roll away!

Why am I talking about this? ***Because an EA that has been curve-fit behaves like an unstable system.***

Shaking the bowl is like changing the input. **How do you shake an EA?** You change the EA's input parameters.

So, to test your EA for robustness you vary its input parameters. Suppose you have an EA that uses a 12 period moving average to open trades. You change the period from 12 to 24 and record your EA's performance.

If your EA performs significantly different when you change the period of the moving average, then the EA is not robust. It's sensitive to changes.

Does this mean it is a bad EA? Not necessarily. It means that it's sensitive to changes to its input. So, you should not be terrible surprised if the EA's performance changes in the future, because, well, things in the future change.

The bottom line: if the EA is super-sensitive to changes it is probably curve-fit. ***It's likely that it will not perform in the future the way that it has performed in the past***.

Summary

To summarize, there are 3 main tests I use to evaluate the quality of an Expert Advisor. I run these tests repeatedly as I make changes to the EA, carefully comparing the results of each test-run.

- **The Entry Test**. Exit each trade after it's been open for a fixed amount of time. Measure the *winning percentage* as the number trades profitable when closed versus the total number of open trades.

- **The Exit Test**. Find the most profitable level of each open trade and compare it to the actual profit of the closed trade. Average all of these results for an overall *exit-quality* metric for the EA.

- **The Curve-Fit Test**. Vary the inputs of the EA and monitor how deeply the changes in input affect the output results. Significant *input-versus-output* changes indicate the EA is sensitive to new or varied input. If the EA is sensitive to new input it will most likely not perform in the future as it has in the past.

All of these tests can be run manually or programmatically. The MQL code is included with instructions on how to implement the entry and the exit test.

One final note: A valuable modification to the exit test would be to extend the "viewing" period of each open trade *beyond* the closed time of the trade. For example, if a typical trade is usually open for about 8 hours, look for the highest profit not only during the entire life of the trade, but extend the period by about 2 hours. This is the new interval to find the highest possible profit of the trade. I have not included this feature in the attached MQL code. I will likely modify this in the near future and will send out an email to everyone on my current mailing list. If you are reading this, you are most likely on my mailing list. If you are not sure, visit our website at **www.iExpertAdvisor.com** and sign up.

How to Use the MQL Code

The attached MQL includes 3 functions used to implement the tests.

fnBarsinTradeMql: This function records how many bars (or candles a trade has been opened) It's used by the ***fnBarExitMql*** function.

The ***fnBarExitMql*** forcibly closes a trade after it has been open for a specified time or number if bars.

The **fnExitTestMql** function records the highest profit of every trade.

The ***deinit*** function, which is a standard MQL function, includes code to compile the results of the exit test.

If you are able to edit the MQL of your EA:

1. For the entry test, substitute the ***fnBarExitMql*** function for the normal exit code. Don't forget to move the stoploss and takeprofit far out to prevent any trades from closing normally.

2. For the exit test, simply include the ***fnExitTestMql*** in the beginning of the start function, and add the ***deinit*** function to the EA.

The code is included below for your convenience. You can also download the MQL file here:
http://expert-advisor-builder.com/mql-code-for-testing-any-expert-advisor/

MQL Source Code

```
// system7 MT4 Expert Advisor
// Build number: 3 Timestamp: 2014/01/30 14:43:25
// Visual Trader Studio Connect, version: 4.0.0.59
// Registered to: david.williams22@iexpertadvisor.com

#property copyright "Copyright 2014, iExpert   (bld #:3   2014/01/30 14:43:25)"
#property link "http://www.iExpertAdvisor.com"

#include <stdlib.mqh>

#include <WinUser32.mqh>
#import "user32.dll"
int     MessageBoxA(int hWnd ,string szText,string szCaption,int nType);

extern int Slippage = 3;

string chartcomment;
string eaname = "TestMql";
double ExitTestResults[];
int ExitTestCount;
string ExitTestString;

#define ALL_TRADES 0
#define WINNING_TRADES 1
#define LOSING_TRADES 2


int fnBarsInTradeMql(int ordertype, int ticket, int magic, string symbol)
{
           int TradeBars=0;
           int total=OrdersTotal();
           for(int pos=0;pos<total;pos++)
           {
                     if( OrderSelect(pos,SELECT_BY_POS) == true )
                     {
                               if( (ticket > 0) && (OrderTicket() != ticket) )
                                  continue;
                               if( (StringLen(symbol) > 0) && (OrderSymbol() != symbol) )
                                  continue;
                               if( (magic > 0) && (OrderMagicNumber() != magic) )
                                  continue;
                               if( (ordertype > 0) && (ordertype != OrderType()) )
                                  continue;
                               TradeBars = MathRound((CurTime()-OrderOpenTime())/(Period()*60));
                     }
           }
           return(TradeBars);
}

void fnBarExitMql(int bars, int ordertype, int ticket, int magic, string symbol )
{
           int total=OrdersTotal();
           for(int pos=0;pos<total;pos++)
           {
                     if( OrderSelect(pos,SELECT_BY_POS) == true )
                     {
                               if( (ticket > 0) && (OrderTicket() != ticket) )
                                  continue;
                               if( (StringLen(symbol) > 0) && (OrderSymbol() != symbol) )
                                  continue;
                               if( (magic > 0) && (OrderMagicNumber() != magic) )
                                  continue;
                               if( (ordertype > 0) && (ordertype != OrderType()) )
                                  continue;

                               int barsintrade = fnBarsInTradeMql(ordertype, ticket, magic, symbol);

                               if( barsintrade >= bars )
                               {
                                         double price = MarketInfo(OrderSymbol(), MODE_ASK);
                                         if( (OrderType() == OP_BUY) || (OrderType() == OP_BUYLIMIT) || (OrderType() == OP_BUYSTOP) )
                                         price = MarketInfo(OrderSymbol(), MODE_BID);
                                         OrderClose(OrderTicket(), OrderLots(), price, Slippage, Red);
                               }
                     }
           } // end for
}

void fnExitTestMql(int type)
{
           if( IsTesting() == false )
             return;

           static double profit;
           static double hiprofit;
           static int ticket;
           double price, point;

           // only supports 1 trade
```

```
		for( int pos=0;pos<OrdersTotal();pos++)
		{
			if(OrderSelect(pos,SELECT_BY_POS)==false) continue;
			ticket = OrderTicket();
			// save hi profit for this trade
			if( OrderType() == OP_BUY )
			{
				price = MarketInfo(OrderSymbol(), MODE_BID);
				profit = price - OrderOpenPrice();
			}
			else
			{
				price = MarketInfo(OrderSymbol(), MODE_ASK);
				profit = OrderOpenPrice() - price;
			}
			if( profit > hiprofit )
			hiprofit = profit;
		}
		// a trade was just closed
		if( (ticket > 0) && (OrdersTotal() == 0 ) )
		{
			double result = 0;

			if( (profit > 0) && (hiprofit > 0) )
			 result = profit/hiprofit;

			// both negative, flip
			if( (profit < 0) && (hiprofit < 0) )
			 result = hiprofit/profit;

			// allow result to be a negative %
			if( (profit < 0) && (hiprofit > 0) )
			 result = profit/hiprofit;

			// last condition not possible : profit > 0 && hiprofit < 0

			bool use = false;
			if( type == 0 )
			{
				use = true;
			}
			else
			{
				if( (type == 1) && (profit > 0) )
				   use = true;
				if( (type == 2) && (profit < 0) )
				   use = true;
			}

			if( use == true )
			{
				double qavg = 0;
				if(hiprofit != 0)
				   qavg = profit/hiprofit;

				point = MarketInfo(OrderSymbol(), MODE_POINT);
				string nl = "\t\t";

				if( MathMod( ExitTestCount,2) != 0 )
				   nl = "\n";

				ExitTestString = StringConcatenate(ExitTestString, ticket,  ": Highest Profit: ", DoubleToStr(hiprofit/point,
2), " Actual Profit: ", DoubleToStr(profit/point, 2), " Quality: ", DoubleToStr(qavg*100,2) ,"%", nl);
				ArrayResize(ExitTestResults, ExitTestCount+1);
				ExitTestResults[ExitTestCount] = result;
				ExitTestCount++;
			}
			ticket = -1;
			profit = 0;
			hiprofit = 0;
		}
}


int start()
{

		static int tick;
		tick++;

		chartcomment="Build # 3, 2014/01/30 14:43:25  (VTS 4.0.0.59)\nTICK:"+tick+ "\n" ;

		fnBarExitMql(  8, 0, 0, 0, "" );
		fnExitTestMql(  ALL_TRADES );

		 return( 0 );
}
// end elements\user\systems\system7\system7.se
```

```
int deinit( )
{
   if( IsTesting() == true )
   {
        double avg=0, sum=0;

                        for( int i=0; i<ExitTestCount; i++)
                          sum += ExitTestResults[i];

                        if( ExitTestCount > 0 )
                          avg = sum/ExitTestCount;

                        ExitTestString = StringConcatenate(ExitTestString, "\n", eaname , " Exit Quality: ",  DoubleToStr(avg*100, 2), "%" );
        MessageBoxA(  0, ExitTestString,eaname + " ExitTest Results", 0 );
        Print(ExitTestString);
   }
}
```