



14,039,697 members

Sign in

[home](#) [articles](#) [quick answers](#) [discussions](#) [features](#) [community](#) [help](#)

Search for articles, questions, tips

Articles » General Programming » Algorithms & Recipes » General



Time Series Analysis in C#.NET Part I

Jeff B. Cromwell, 8 May 2012

★★★★★ 5.00 (5 votes) Rate this:

This article examines the use of the ABMath and MathNet .NET packages for time series analysis.

Introduction

These days my company at [TCW](#) is focused on developing linear and nonlinear time series applications in .NET languages for middleware applications for neuroscience research. Specifically, I am building hybrid applications in the R statistical and MATLAB languages for prototyping and then translating to C#.NET and/or C++.NET to analyze and predict fMRI/EEG data from BCI gaming applications.

This is the first of a five part series that examines the use of open source libraries to build .NET based time series applications that can be used for both neuroscientific research and commodity/stock price forecasting. The goal is to make available open source nonlinear time series algorithms as part of a statistical engine, i.e. TIME JAQUAR, for both the time and frequency domain that can run on multi-CPU/multi-GPU platforms for handling real-time modeling and prediction from BCI devices. All code and releases from this series of articles can be found on CodePlex at [modelmaker3.codeplex.com](#).

Because there are many good tutorials on the web for doing time series analysis along with what I have written, I refrain from doing that here.

Background

There are several good libraries to use as both namespaces and prototyping templates to help prototype linear and nonlinear time series applications in .NET. Here are a couple that are used in my applications:

Library 1: [Cronos](#)Library 2: [MathNet](#)

Please consult the documentation on these libraries for more information. Of course, the scalability of the linear algebra and optimization methods are the foundation stones and the subject of additional articles. I have provide the basics in Figure 1 to illustrate some of the features of both libraries and posting the code on CodePlex so that you can follow the project.

Currently, I am modifying these as well for GPU performance.

Using the code

The following steps are to be performed for doing the estimation:

Step 1: Specify the parameters of the ARMA Model**Step 2:** Generate the data for the simulation**Step 3:** Fit the model by using an estimation method like Maximum Likelihood Estimation (MLE)**Step 4:** Obtain the Model Data, Predictions, and Residual Output**Figure 1.** Code listing for steps 1-4

Hide Shrink ▲ Copy Code

```
using ABMath.ModelFramework.Models;  
using ABMath.ModelFramework.Data;
```

```

using MathNet.Numerics.Distributions;
using MathNet.Numerics.RandomSources;
using MathNet.Numerics;
namespace TCW_ModelMaker3
{
class Program
{
    static void Main(string[] args)
    {
        runARMATest();
    }
    public static void ConsoleWrite()
    {
        Console.WriteLine("Welcome to ModelMaker 3");
        Console.ReadKey();
    }
    public static void runARMATest()
    {
        //Specify the AR and MA parameters
        ARMAModel model1 = new ARMAModel(4, 3);
        ARMAModel model2 = new ARMAModel(0, 1);
        //Select an random distribution for the data
        var sd = new StandardDistribution();
        //Make new time series variable
        TimeSeries ts1 = new TimeSeries();
        var dt = new DateTime(2001, 1, 1);
        var normalSim = new StandardDistribution();
        var current = new DateTime(2001, 1, 1);
        //Create the data
        for (int t = 0; t < 100; ++t)
        {
            double s1 = normalSim.NextDouble();
            ts1.Add(current, s1, false);
            current = new DateTime(current.AddDays(1).Ticks);
        }

        model1.SetInput(0, ts1, null);
        //Maximum Likelihood Estimation
        model1.FitByMLE(10, 10, 10, null);
        //Compute the residuals
        model1.ComputeResidualsAndOutputs();
        //Get the predicted values
        var preds = model1.GetOutput(3) as TimeSeries;
        var predName = model1.GetOutputName(3);
        //Write the model description with parameter values
        Console.WriteLine(model1.Description);
        Console.ReadKey();
    }
}
}

```

The most interesting method above is the the *FitByMLE* method. This function samples from the parameter space with a Halton sequence and obtains the best model by optimizing the log-likelihood. Parameters values have three states:

ParameterState.Locked, **ParameterState.Free**, or **ParameterState.Consequential**. The locked parameters are held at their current values in optimization method, free parameters are optimized, and consequential parameters are computed as a function of other parameters and the data. See the library on Cronos for more information.

Figure 2. Code listing for the Maximum Likelihood Method

Hide Shrink ▲ Copy Code

```

public virtual void FitByMLE(int numIterationsLDS, int numIterationsOpt,
    double consistencyPenalty,
    Optimizer.OptimizationCallback optCallback)
{
    thisAsMLEEstimable = this as IMLEEstimable;
    if (thisAsMLEEstimable == null)
        throw new ApplicationException("MLE not supported for this model.");

    int optDimension = NumParametersOfType(ParameterState.Free);
    int numConsequential = NumParametersOfType(ParameterState.Consequential);
    int numIterations = numIterationsLDS + numIterationsOpt;

    var trialParameterList = new Vector[numIterationsLDS];
    var trialCubeList = new Vector[numIterationsLDS];

    var hsequence = new HaltonSequence(optDimension);

    for (int i = 0; i < numIterationsLDS; ++i)

```

```

{
    Vector smallCube = hsequence.GetNext();
    Vector cube = CubeInsert(smallCube);
    trialParameterList[i] = thisAsMLEEstimable.CubeToParameter(cube);
    trialCubeList[i] = cube;
}

var logLikes = new double[numIterationsLDS];
for (int i = 0; i < numIterationsLDS; ++i)
{
    Vector tparms = trialParameterList[i];
    if (numConsequential > 0)
    {
        tparms = ComputeConsequentialParameters(tparms);
        trialParameterList[i] = tparms;
    }

    double ll = LogLikelihood(tparms, consistencyPenalty);
    logLikes[i] = ll;

    if (optCallback != null)
        lock (logLikes)
            optCallback(tparms, ll, i*100/numIterations, false);
}

// Step 1: Just take the best value.
Array.Sort(logLikes, trialParameterList);
Parameters = trialParameterList[numIterationsLDS - 1];

// Step 2: Take some of the top values and use them to create a simplex, then optimize
// further in natural parameter space with the Nelder Mead algorithm.
// Here we optimize in cube space, reflecting the cube when necessary to make parameters valid.
var simplex = new List<Vector>();
for (int i = 0; i <= optDimension; ++i)
    simplex.Add(FreeParameters(thisAsMLEEstimable.ParameterToCube(
        trialParameterList[numIterationsLDS - 1 - i])));
var nmOptimizer = new NelderMead { Callback = optCallback, StartIteration = numIterationsLDS };
currentPenalty = consistencyPenalty;
nmOptimizer.Minimize(NegativeLogLikelihood, simplex, numIterationsOpt);
Parameters = ComputeConsequentialParameters(
    thisAsMLEEstimable.CubeToParameter(CubeFix(CubeInsert(nmOptimizer.ArgMin))));

ComputeResidualsAndOutputs();
}

```

Of course, this is the main algorithm of interest based on Nelder Mead and the subject of a future article. However, this should be enough to get you started.

Points of Interest

Fragments of the code in Figure 1. can be placed in the form of a snippet into Visual Studio 2010 using a Snippet Editor. Visit the Workshop to see the video for this.

History

- 5-8-2012: Code placed on CodePlex.

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)

Share

About the Author



Jeff B. Cromwell

CEO The Cromwell Workshop
United States 

Dr. Jeff B. Cromwell is the CEO/Neuroeconomist at The Cromwell Workshop.

Scholar Site: <http://independent.academia.edu/JeffCromwell>
Web Site: www.cromwellworkshop.com

You may also be interested in...



Time Series Analysis in C#.NET:
Part II



Chess Console Game in C++



More Texas Holdem Analysis in C#:
Part 2



EMGU Multiple Face Recognition
using PCA and Parallel
Optimisation



SQL DISTINCT Clause For SQL
Server



Diving in OOP (Day 3):
Polymorphism and Inheritance
(Dynamic Binding/Run Time
Polymorphism)

Comments and Discussions

You must [Sign In](#) to use this message board.

Search Comments 

Spacing

Relaxed

Layout

Open All


Per page

25

[Update](#)

[First](#) [Prev](#) [Next](#)

 **Code Not available.....Link is not working..**

 **AVI - The Gr8**

30-Nov-12 7:17

Hi,

I read your article..it seems promising esp. to visualize the large multivariate time series data..

but i couldn't download the code..
could you please provide me the download link..
Thanks in advance.

[Sign In](#) · [View Thread](#)

