White Noise Test: testing for serial correlation and nonstationarity of long time series after ARIMA modeling

Margaret Y Mahan*†, Chelley R Chorn‡, Apostolos P Georgopoulos†

Abstract—Functional connectivity analysis has been a dominant technique for assessing the brain as a dynamic network and is becoming increasingly prevalent in the neuroscientific community. The assessment of functional connectivity relies, in turn, on the proper calculation of interactions between neural time series, which is achieved by rendering each individual series stationary and nonautocorrelated (i.e. white noise -> "prewhiten" the series). This ensures that the functional connectivities computed subsequently are due to the interactions between the series and do not reflect internal dependencies of the series themselves. An established method for prewhitening time series is to apply an AutoRegressive (AR, p) Integrative (I, d) Moving Average (MA, q) model (ARIMA) and retain the residuals. To diagnostically check whether the model orders (p, d, q) are sufficient, both visualization (ACF & PACF plots) and tests (Ljung-Box, KPSS, etc.) of the residuals are performed. However, these tests are not robust for high-order models in long time series. Additionally, as dataset size increases (i.e. number of time series to model) it's not feasible to visually inspect each series independently. As a result, there is a need for robust alternatives to diagnostic evaluations of ARIMA modeling. In this presentation, we demonstrate how to perform ARIMA modeling of long time series using Statsmodels. For this process, we implemented all methods and solvers for parameter estimation and identified a set of fit parameters for which the long time series were modeled appropriately, as compared to known statistical packages. Then, we present a new algorithm to test the hypothesis that a prewhitened series does not differ significantly from white noise. This test was validated using three datasets: (1) magnetoencephalography recordings, (2) multielectrode arrays, and (3) simulations. Overall, the white noise test presented here provides a robust alternative to diagnostic checks of ARIMA modeling for long time series. (WARNING TO REVIEWERS: we had a mishap last week that put us behind on this paper; we will submit revisions throughout the week and will have a full paper by no later than Friday 6/19/15 at 2pm Central time - thanks for understanding!)

Index Terms—time series, ARIMA, statsmodels

Introduction

Correlation analysis has been a dominant technique for assessing relationships between time series and is becoming increasingly prevalent in the scientific community; for example, assessing brain networks by calculating pairwise correlations of time series generated from different areas of the brain. The assessment of correlated networks relies, in turn, on the proper calculation of interactions between time series, which is achieved by rendering each individual series stationary and nonautocorrelated (i.e. white noise = "prewhiten" the series). This ensures that the correlations computed subsequently are due to the interactions between the series and do not reflect internal dependencies of the series themselves. Therefore, it is essential to render each individual time series stationary and nonautocorrelated before any type of correlation analysis.

Prewhitening

A white noise process is a time series of random shocks, normally and independently distributed, with a zero mean and constant variance. After a time series is modeled, if the residuals are white noise, then we say the series has been prewitened. An established method for prewhitening time series is to apply an AutoRegressive (AR) Integrative (I) Moving Average (MA) model (ARIMA) and retain the residuals (Box 1976). This method requires the time series to be: (1) equally spaced over time, (2) contain no missing values, (3) of sufficient length, (4) stationary in the second or weak sense, and (5) are usually serially dependent. Three steps are followed for ARIMA modeling. First, factors that influence the time series are detected (model identification); second, the contribution of these factors are estimated (model estimation); and third, the adequacy ("goodness") of the model is evaluated (model evaluation).

Implementing ARIMA using Statsmodels

ARIMA modeling has been implemented in Python with the Statsmodels package. It includes parameter estimation and model evaluation procedures.

Once the number of ARIMA parameters has been selected by choosing a model order, these parameters can be estimated with the statsmodels.tsa.arima_model.ARIMA.fit() function to maximize the likelihood that these coefficients describe the data. First, initial estimates of the parameters are used to get close to the desired parameters. Second, optimization functions are applied to adjust the parameters to maximize the likelihood by minimizing the negative loglikelihood function. If adequate initial parameter estimates were selected, a local optimization algorithm will find

* *Corresponding author: mahan027@umn.edu*
† *Brain Sciences Center, Minneapolis VA Health Care System*
‡ *Department of Biomedical Informatics and Computational Biology, University of Minnesota*

the local loglikelihood minimum near the parameter estimates, which will be the global minimum.

In statsmodels, default starting parameter estimations are calculated using the Hannan-Rissanen method (Hannan and Rissanen 1982) and these parameters are checked for stationarity and invertibility. If "method" is set to "css-mle", starting parameters are estimated further with conditional sum of squares methods. Parameters estimated in this way are not guaranteed to be stationary, so starting parameters may be set as an input variable (start_params) to ARIMA.fit(). We have implemented a custom starting parameter selection method which forces stationarity and invertibility if necessary. In addition, the Hannan-Rissanen method uses an initial AR model with an order selected by minimizing BIC, then estimates ARMA using the residuals from that model. This initial AR model is required to be larger than max(p,q) of the desired ARIMA model, which is not guaranteed with an AR selected by BIC criterion. We have implemented a method similar to Hannan-Rissanen, the long AR method, which is equivalent to Hannan-Rissanen except the initial AR model is set to be large (AR=300). This results in an initial AR model order which is guaranteed to be larger than max(p, q), and starting parameter selection is more time efficient since fitting multiple AR model orders to optimize BIC is not required .

To fit ARIMA models, statsmodels has options for methods and solvers. The chosen method will determine the type of likelihood for estimation, where "mle" is the exact likelihood maximization, "css" is the conditional sum of squares minimization, and "css-mle" involves first estimating the starting parameters with css followed by an mle fit. The solver variable in ARIMA.fit() designates the optimizer from scipy.optimize for minimizing the negative loglikelihood function. Optimization methods "nm" (Nelder-Mead) and "powell" are the most time efficient because they do not require a score, gradient or hessian. The next fastest methods, "lbfgs" (limited memory Broyden-Fletcher-Goldfarb-Shanno), "bfgs" (Broyden-Fletcher-Goldfarb-Shanno), "cg" (conjugate gradient), and "ncg" (Newton conjugate-gradient), require a score or gradient, but no hessian. "newton" (Newton-Raphson) requires a score, gradient and hessian. A global optimizer "basinhopping" displaces parameters randomly before minimizing with another local optimizer. For more information about these solvers, see statsmodels.base.model.GenericLikelihoodModel.

Because mle and css-mle methods are not time efficient and their results approach those of css for long time series [Box Jenkins, section 12.3.2], the preferred method for long series is css. The solvers basinhopping and newton are methods which require noticeably longer calculation time which may be impractical for large datasets. In addition, for the methods that do not require hessian it is more time efficient to set skip_hessian to True when fitting the model, as it is otherwise only calculated after fitting is completed and adds significant computation time.

Application Datasets

Magnetoencepalography (MEG)

To evaluate the functional brain at high temporal resolution, MEG is the optimal technique among all other neuroimaging techniques. This technique noninvasively measures magnetic fluctuations generated by synchronized neural activity in the brain. For the applications below, MEG recordings were collected using a 248-channel axial gradiometer system (Magnes 3600WH, 4-D Neuroimaging, San Diego, CA) sampled at ~1kHz from 50 cognitively healthy women (40 - 93 years, 70.58 ± 14.77, mean ± std dev) in a task-free state (i.e. resting state). The data were time series consisting of 50,000 values per subject and channel. Overall, the full MEG data matrix contains 50 subjects x 248 channels x 50,000 time points.

Local Field Potentials (LFP)

Isolated cells from cortical tissue of four embryonic day 18 (BrainBits, Springfield, IL) rat brains were plated on multielectrode arrays (MEA). The MEA (Multi Channel Systems, model MEA 120-2-System, Reutlingen, Germany) consists of an array of 60 electrodes (59 channels, 1 reference) embedded on a flat surface surrounded by a circular wall that creates a well around the electrodes. Electrical activity was recorded simultaneously from all electrodes for 1 minute at a sampling frequency of 10 kHz from the brain cultures for 40 days in vitro. Local field potential (LFP) activity was derived from the data by applying a second-order band-pass filter to reject frequencies outside the LFP range. The filtered time series were then downsampled to 1 kHz for further analysis. Overall, the full LFP data matrix contains 4 cultures x 40 days in vitro x 59 channels x 50,000 time samples.

Simulated Time Series (STS)

Performing ARIMA modeling using Statsmodels on long time series

White Noise Test

Conclusions

REFERENCES

[ref] G. Box, G. Jenkins. *Time series analysis: forecasting and control*, (Holden Day, 1976).