White Noise Test: detecting autocorrelation and nonstationarities in long time series after ARIMA modeling

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

Abstract—Time series analysis has been a dominant technique for assessing relations within datasets collected over time 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 these relations 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, or to "prewhiten" the series). This ensures that the relations 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 and statistical tests (e.g., Ljung-Box test) 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 is not feasible to visually inspect each series independently. As a result, there is a need for robust alternatives to diagnostic evaluations of ARIMA modeling. Here, we demonstrate how to perform ARIMA modeling of long time series using *Statsmodels*, a library for statistical analysis in Python. Then, we present a comprehensive procedure (White Noise Test) to detect autocorrelation and nonstationarities in prewhitened time series, thereby establishing that the series does not differ significantly from white noise. This test was validated using time series collected from magnetoencephalography recordings. Overall, our White Noise Test provides a robust alternative to diagnostic checks of ARIMA modeling for long time series.

Index Terms—Time series, Statsmodels, ARIMA, statistics

Introduction

Time series are discrete, stochastic realizations of underlying data generating processes [Yaffee]. In other words, a time series is a set of consecutive samples collected over a time interval, such as temperature recordings at regular intervals. They are ubiquitous in any field where monitoring of data is involved. For example, time series can be environmental, economic, or medical. In addition, time series can provide information about trends (e.g., broad fluctuations in values) and cycles (e.g., systematic, periodic fluctuations in values). Time series analysis are also used to predict the next value in the series, given some model of its history. This is of special importance in environmental and econometric studies where forecasting the next set of values (e.g., the weather or a stock price) may have serious practical consequences. In other fields, time series provide crucial information about an evolving process (e.g., rate of spread of a disease or changing pollution levels) with implications about the effect of interventions. Finally, time series can provide fundamental information about the process that generates them, leading to a scientific understanding of that process (e.g., brain network analysis).

In time series analysis, there are two main investigative methods: frequency-domain and time-domain. In this paper, only analysis in the time-domain is considered. Within the time-domain, typically crosscorrelation analysis is utilized as a measure of the relation between two time series. Now, it is commonly the case that a time series contains some autocorrelation, meaning that values in the time series are influenced by previous values. It is also common for a time series to exhibit nonstationarities, such as drifts or trends over time. In either case, the crosscorrelation function calculated between two series containing either autocorrelation or nonstationarities will give misleading results, such as an inflated correlation between two series where there is none. To circumvent this, time series are modeled to remove such effects, as in the case of prewhitening.

Prewhitening

A white noise process is a continuous time series of random values, with a constant mean and variance, normally and independently distributed, and nonautocorrelated. If after modeling a time series the residuals are practically white noise, then we say the series has been prewhitened. 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]. The full specification of an ARIMA model comprises the orders of each component, (*p*, *d*, *q*), where *p* is the number of preceding values in the autoregressive component, *d* is the number of differencing, and *q* is the number of preceding values in the moving average component. An ARIMA model with orders *p*, *d*, and *q*, is a discrete time linear equations with noise of the form:

$$(1-\sum_{k=1}^{p}\phi_k L^k)(1-L)^d X_t = (1+\sum_{k=1}^{q}\theta_k L^k)\varepsilon_t$$

* *Corresponding author: mahan027@umn.edu*
‡ *Brain Sciences Center, Minneapolis VA Health Care System & University of Minnesota*

where L is the time lag operator, $Lx_t = x_{t-1}$.

In ARIMA modeling, the I component is addressed first, followed by jointly addressing the AR and MA components. Most importantly, the ARIMA method requires the input time series to be: (1) equally spaced over time, (2) of sufficient length, (3) continuous (i.e., no missing values), and, specifically for the ARMA portion, (4) stationary in the second or weak sense, meaning the mean and variance remain constant over time and the autocovariance is only lag-dependent.

Prewhitening using ARIMA modeling takes three main steps. First, identify and select the model, by detecting factors that influence the time series, such as nonstationarities or periodicities, and identifying the AR and MA components (i.e., model orders). Second, estimate parameter values, by using an estimation function to optimize the parameter values for the desired model. Third, evaluate the model, by checking the model's adequacy through establishing that the series has been rendered stationary and nonautocorrelated. This time series modeling is iterative, successively refining the model until stationary and nonautocorrelated residuals are obtained. Overall, a good model serves three purposes: providing the background information for further research on the process that generated the time series; enabling accurate forecasting of future values in the series; and yielding the stationary and nonautocorrelated residuals necessary to evaluate accurately associations between time series, since they are devoid of any dependencies stemming from within the series themselves.

Here, we implement two complementary tests to establish stationarity, which determines the value of the I(d) order. Using these stationary series, we use median correlation values at each lag of the autocorrelation (ACF) and partial autocorrelation (PACF) functions to identify a range of AR(p) and MA(q) orders to implement combinatorially. Then we utilize the *Statsmodels* package to find the method-solver combination that provides good metrics for long time series. Finally, we present a novel approach (White Noise Test) to diagnostic checking of ARIMA modeling for long time series, which evaluates residual series based on stationarity and nonautocorrelation. Using our approach, an investigator can perform ARIMA modeling and evaluate candidate models with ease for large datasets and datasets containing long time series.

Model Identification and Selection

There are several factors that can influence a value in a time series, which arise from previous values in the series, variability in these values, or nonstationarities (trend, drift, changing variance, or random walk). It is important to properly remove the effects of these factors by modeling the time series and taking the residuals. To identify the model orders for an ARIMA(p, d, q), the ACF and PACF are used.

First, nonstationarities need to be removed before ARMA modeling. A nonstationary process is identified by an ACF that does not tail away to zero quickly or cut-off after a finite number of steps. If the time series is nonstationary, then a first differencing of the series is computed. This process is repeated until the time series is stationary, which determines the value of d (i.e., the value of d is the number of times the derivative of the series is taken to achieve stationarity). Two of the most frequently used tests for detecting nonstationarities are the augmented Dickey-Fuller (ADF) test [Said] and the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test [Kwiatkowski]. The ADF is a unit root test for the null hypothesis that a time series is I(1) while the KPSS is a stationarity test for the null hypothesis that a time series is I(0). Since these tests are complementary, we use them together to determine whether a series is stationary. In our case, a series taken to be nonstationary, if the ADF null hypothesis is accepted and the KPSS null is rejected. We implement the ADF test using *Statsmodels* and the KPSS test using the *Arch* Python package.

Once nonstationarities have been removed, ARMA modeling can begin. To choose the p and q orders, the ACF and PACF of the stationary (differenced) series will show patterns based on which tentative ARMA model can be postulated. There are three main patterns. A pure MA(q) process will have an ACF that cuts off after q lags and a PACF that tails off with exponential or oscillating decay. A pure AR(p) process will have an ACF that tails off with exponential or oscillating decay and a PACF that cuts off after p lags. For a mixed-model ARMA(p, q) process, both the ACF and PACF will tail off with exponential or oscillating decay. Using these patterns, the model selection begins by using the minimum orders to achieve stationary and nonautocorrelated residuals.

Parameter Value Estimation

ARIMA modeling has been implemented in Python with the *Statsmodels* package [McKinney], [Seabold]. It includes parameter value estimation and model evaluation procedures. We import the *Statsmodels* and *Numpy* packages as:

```
import statsmodels.api as sm
import numpy as np
```

After the model orders have been selected, the model parameter values can be estimated with the `sm.tsa.arima_model.ARIMA.fit()` function to maximize the likelihood that these parameter values (i.e., coefficients) describe the data, as follows. First, initial estimates of the parameter values are used to get close to the desired parameter values. Second, optimization functions are applied to adjust the parameter values to maximize the likelihood by minimizing the negative log-likelihood function. If adequate initial parameter value estimates were selected, a local optimization algorithm will find the local log-likelihood minimum near the parameter value estimates, which will be the global minimum.

In *Statsmodels*, default starting parameter value estimations are calculated using the Hannan-Rissanen method [Hannan] and these parameter values are checked for stationarity and invertibility (these concepts are discussed in further detail in the next section). If `method` is set to `css-mle`, starting parameter values are estimated further with conditional sum of squares methods. However, parameter values estimated in this way are not guaranteed to be stationary; therefore, we advise specifying starting parameter values as an input variable (`start_params`) to `ARIMA.fit()`. A custom starting parameter value selection method may be built upon a copy of `sm.tsa.ARMA._fit_start_params_hr`, which forces stationarity and invertibility on the estimated `start_params` when necessary. For example,

```
if not np.all(np.abs(np.roots(np.r_
       [1, -start_params[k:k + p]])) < 1) or
   not np.all(np.abs(np.roots(np.r_
       [1, start_params[k + p:]])) < 1):
   start_params = np.array(start_params[0:k]
                  + [1./(p+1)] * p + [1./(q+1)] * q)
```

In addition, the Hannan-Rissanen method uses an initial AR model with an order selected by minimizing Bayesian Information Criterion (BIC); then it 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 value 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 (MLE), `css` is the conditional sum of squares (CSS) minimization, and `css-mle` involves first estimating the starting parameter values 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 solvers `nm` (Nelder-Mead) and `powell` are the most time efficient because they do not require a score, gradient, or Hessian. The next fastest solvers, `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. The `newton` (Newton-Raphson) solver requires a score, gradient, and Hessian. Lastly, a global solver `basinhopping`, displaces parameter values randomly before minimizing with another local optimizer. For more information about these solvers, see `sm.base.model.GenericLikelihoodModel`.

Model Evaluation

There are two components in evaluating an ARIMA model, namely, model stability and model adequacy. For the model to be stable, the roots of the characteristic equations

$$1 - \phi_1 L - \cdots - \phi_p L^p = 0$$

where ϕ_i are the estimated AR parameter values, L is the time lag operator, and

$$1 + \theta_1 L + \cdots + \theta_q L^q = 0$$

where θ_i are the estimated MA parameter values, should lie outside the unit circle, i.e., within bounds of stationarity (for the p parameter values) and invertibility (for the q parameter values) [Pankratz]. For the model to be adequate, the residual time series should not be significantly different from white noise; in other words, the series should have constant mean and variance, and each value in the series should be uncorrelated with other realizations up to k lags. If either model stability or adequacy have not been established, then model identification and selection should be revised, and the diagnostic cycle continued, iteratively, until established.

Inspecting the p and q parameter values for being within the bounds of stationarity and invertibility checks model stability. Typically, this will be accomplished during parameter value estimation. The model adequacy is checked by examining the time-varying mean of the residuals (should be close to zero), their variance (should not differ appreciably along time), and their autocorrelation (should not be different from chance). Finally, the ACF and PACF of the residuals should not contain statistically significant terms more than the number expected by chance. This number depends on the number of lags; for example, if k = 40 lags, one would expect 2 values (5% of 40) to exceed their standard error. Under the assumption that the process is white noise and when the length (N) of the series is long, the standard error of the sample autocorrelation (and partial autocorrelation) [Bartlett] approximates to:

$$\text{Standard Error} = 1/\sqrt{N}$$

Several statistical tests are available to detect autocorrelation. Most notable is the Ljung-Box test [Ljung], which is applied to residuals to detect whether they exhibit autocorrelation. The test statistic is calculated for each of h lags being tested. Another common test to detect autocorrelation is the Durbin-Watson test [Durbin]; however, unlike the Ljung-Box test which is calculated for h lags, the Durbin-Watson test is calculated only for lag 1. Therefore, any autocorrelation beyond lag 1 will not be detected by this test. Similar to the Ljung-Box test is the Breusch-Godfrey Lagrange multiplier test [Breusch], [Godfrey]. This test also aims to detect autocorrelation up to h lags tested. We compare our model evaluation, namely the White Noise Test, to both the Ljung-Box and Breusch-Godfrey tests.

White Noise Test

The White Noise Test (Figure 1) calculates multiple attributes on residuals. Inclusively, the attributes characterize an individual residual series by its "whiteness". To change the degree of "whiteness", the thresholds in the red boxes of Figure 1 may be made more or less conservative.

Excluded data: Channels that could not be modeled with the given model order were excluded from further analysis. Additionally, channels with extreme values beyond a threshold of 5 per channel, calculated on the residuals for each model order, were also excluded from further analysis (xVAL in Table 1 and 5). Extreme values are calculated as follows. For each raw series, the interquartile range (IQR) is calculated.

$$IQR = 75^{th}percentile - 25^{th}percentile$$

Using the IQR, Tukey's outer fences are calculated [Tukey].

$$Fence_{upper} = 75^{th}percentile + 3 \times IQR$$

$$Fence_{lower} = 25^{th}percentile - 3 \times IQR$$

Then, the values below the lower fence and above the upper fence are counted as extreme values. If this count is greater than 5, the series is removed from further consideration when selecting model orders.

Normality: Each residual series was tested for normality using the Kolmogorov–Smirnov test. Residual series not significantly different from normal ($\alpha = 0.01$) were retained.

Constant mean: Each residual series was split into 10% nonoverlapping windows (i.e., 10% of 50000 time points = 10 windows of 5000 time points). For each window, a one-sample t-test was calculated ($\alpha = 0.001$). A count of the number of windows with means significantly different from zero was retained for each residual series (maximum value = 10). Residual series with > 1 section containing means significantly different from zero were excluded (cMEAN in Table 1, 3 and 5).

Constant variance: For each residual series, the 10% nonoverlapping windows were also tested for equal variances using Bartlett's test ($\alpha = 0.001$). Each window was compared to the variance of the full residual series. A count of the number of windows with unequal variances was retained for each residual series (maximum value = 10). Residual series with > 1 section containing significantly different unequal variances were excluded (cVAR in Table 1, 3 and 5).

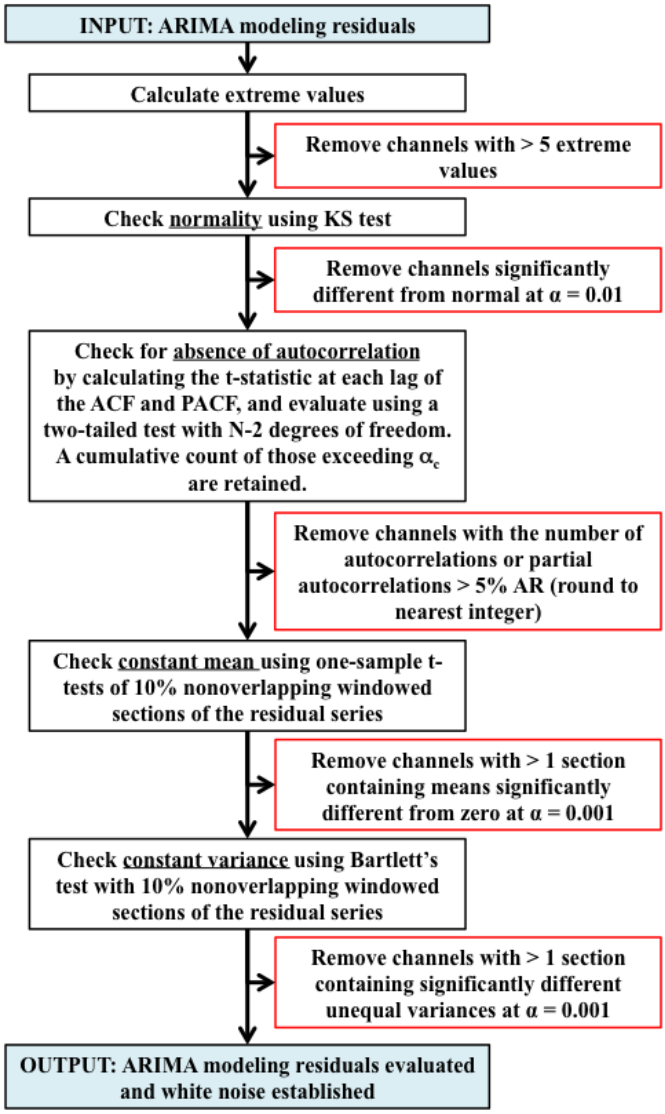

Fig. 1: *White Noise Test Procedure.*

Uncorrelated with other realizations: The ACF and PACF were calculated up to AR lags and the number of lags exceeding statistical significance were counted. To determine this, the

$$t_{statistic} = \frac{|ACF_k|}{StandardError}$$

is calculated at each lag, k, and evaluated against the null hypothesis that ACF_k using a two-tailed test with N-2 degrees of freedom. A cumulative count of those exceeding

$$\alpha_c = \frac{0.01}{AR} \tag{1}$$

are retained (note: α_c incorporates a Bonferroni correction, $\frac{1}{AR}$, and is rounded to the nearest integer). The result is a conservative threshold for detecting a significant autocorrelation or partial autocorrelation. We set a threshold for the cumulative count to be greater than 5% of the AR order (round to nearest integer) for either the ACF or PACF for each channel (tACF and tPACF in Table 1, 3 and 5).

To determine whether our thresholding levels are within what is expected by chance, we apply the White Noise Test procedure

Count	xVAL	tACF	tPACF	cMEAN	cVAR
0	531	593	594	597	596
1	67	0	0	3	4
≥ 2	2	7	6	0	0

TABLE 1: *White Noise Attributes, listed as extreme values (xVAL), thresholded ACF and PACF (tACF, tPACF), constant mean (cMEAN) and constant variance (cVAR), and the count column is the number of randomly generated series failing a given attribute.*

(Figure 1) to 600 randomly generated white noise series. Attributes calculated on these series are shown in Table 1.

Magnetoencephalography (MEG) Dataset

To evaluate the functional brain, MEG is a useful technique because it measures magnetic fluctuations generated by synchronized neural activity in the brain noninvasively and at high temporal resolution. 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 ~1 kHz 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 dataset contains 50 samples x 248 channels x 50,000 time points.

Performing ARIMA Modeling

Here, we first determine which method-solver combination from *Statsmodels* provides the most reliable and valid residuals, while also maintaining a respectable processing time for the MEG dataset. Then, using this method-solver, investigations into identifying and selecting model orders are performed, followed by parameter value estimations on a range of model orders. Residuals from these models are processed to detect autocorrelation and nonstationarities using our White Noise Test. Finally, these models are compared and evaluated.

Implementing Method-Solvers

The length and quantity of time series have a direct impact on the ease of modeling. Therefore, we aim to implement an iterative approach to ARIMA modeling while keeping focus on model reliability and validity of residuals, along with incorporating an efficiency cost (i.e., constraints on allowed processing time). The goal for this stage is to determine which method-solver in *Statsmodels* is most appropriate for the application dataset.

To accomplish this, we randomly select 5% (round to nearest integer) of the channels from each sample in the full MEG dataset (i.e., 5% of 248 channels with 50 samples gives N = 600) to construct the test dataset. Next, we select a range of model orders: AR = {10, 20, 30, 40, 50, 60}, I = {1}, MA = {1, 3, 5}. Using each method-solver group (N = 16) and model order combinations (N = 18), we now have 288 testing units. For each of the testing units, ARIMA modeling is performed on each channel in the test dataset.

If 2% of the test dataset channels have a processing time > 5 minutes per channel, the testing unit is withdrawn from further analysis and deemed inefficient. Otherwise, for each channel, four measures are retained. The first measure is the AIC_c (Akaike

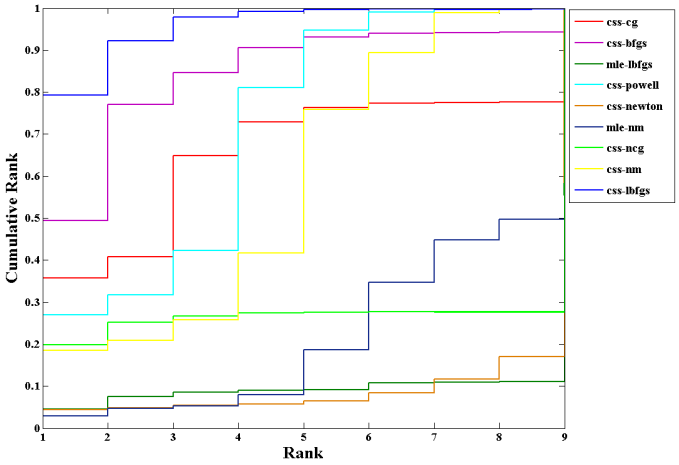

Fig. 2: *MEG CDF Ranks*

Method-Solver	Mean Time (s)	Highest Model	Mean Ranks	Final Rank
`css-lbfgs`	91.47	60-1-3	1.32	1
`css-bfgs`	115.22	60-1-3	2.23	2
`css-powell`	54.47	60-1-5	3.25	3
`css-cg`	132.78	50-1-1	3.77	4
`css-nm`	39.55	60-1-3	4.29	5
`css-ncg`	138.97	20-1-3	6.90	6
`mle-nm`	85.71	30-1-5	7.31	7
`mle-lbfgs`	57.7	10-1-5	8.29	8
`css-newton`	235.11	20-1-1	8.36	9

TABLE 2: *Ranking Method-Solvers for ARIMA modeling of MEG data.*

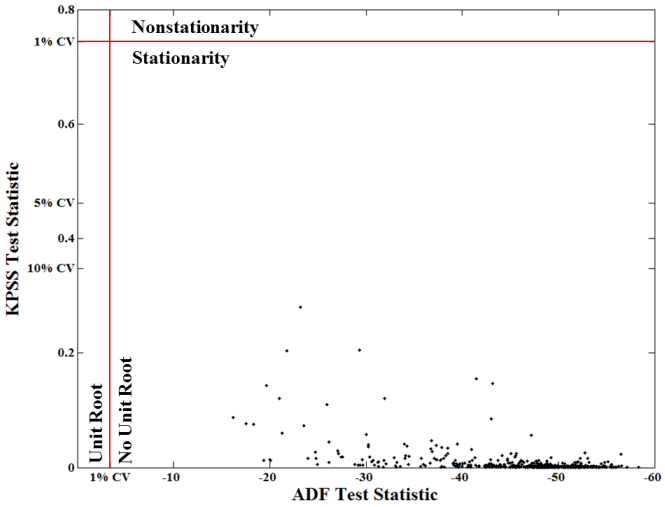

Fig. 3: *Stationarity (KPSS) and Unit Root (ADF) Tests*

Information Criterion with correction), which reflects the quality of the statistical model's performance. The second and third measures are the cumulative counts of tACF and tPACF. The final measure is the processing time, which is measured on each channel and is the time, in seconds, for the ARIMA modeling process to produce residuals. For all four measures, lower values indicate better performance. After calculating the measures, for each channel and model order, ranks for the first three measures are calculated across the method-solver groups, with tied ranks getting the same rank number.

For the 16 method-solver combinations tested, 7 were inefficient at all tested model orders (`css-basinhopping`, `mle-bfgs`, `mle-newton`, `mle-cg`, `mle-ncg`, `mle-powell`, `mle-basinhopping`). The cumulative distribution functions (CDFs) of each method-solver group ranks are calculated and plotted in Figure 2. In this plot, larger area under the curve indicates better performance. Thus, the `css-lbfgs` has the best performance.

In Table 2, the mean time per channel for each method, except withdrawn methods, is given, along with the highest order able to be modeled by the given method-solver group. Mean ranks were calculated for each method-solver, shown in Table 2, and used for the final rank calculation. In the test dataset, the `css-lbfgs` method-solver outperformed all others while maintaining a reasonable time per channel (91.47 seconds). The results also show that the CSS methods generally outperform the MLE methods, for long time series. The `css-lbfgs` method-solver was retained for all further analysis.

Identifying and Selecting Model Orders

Before selecting the differencing model order, d, each series is inspected for extreme values. To determine the model orders, channels with greater than five extreme values are excluded. As discussed previously, if a series is deemed nonstationary, then a first differencing of the series is computed. To determine nonstationarity, examine the ACF plot. A clear indication of nonstationarity will be if the ACF does not tail away to zero quickly or cut-off after a finite number of steps, which is the case with MEG raw time series. Therefore, the MEG time series are first differenced ($d = 1$).

Next we check the series for stationarity; recall, an appropriately differenced process should be stationary. Both the KPSS stationarity test and ADF unit root test are calculated for 60 lags. Their values plotted against each other are shown in Figure 3. The KPSS statistic ranges from 0 to 0.28; since all KPSS test statistics calculated are less than the critical value (CV) of 0.743 at $\alpha = 0.01$, the null hypothesis of stationarity cannot be rejected. The ADF statistic ranges from -16.19 to -58.32; since all ADF test statistics calculated are more negative than the CV of -3.43 at $\alpha = 0.01$, the null hypothesis of a unit root is rejected. Taken together, we have established lack of nonstationaritiy for our test dataset.

Taking the differenced series, the ACF and PACF are calculated for 60 lags. The median correlation value for each lag is plotted in Figure 4. From this figure, a mixed-model ARMA(p, q) process is seen since both the ACF and PACF tail off with oscillating decay. To decide on the p and q orders, we look at Figure 4 and see the highly AR nature of the PACF plot up to about 30 lags; we also see the MA component expressed in the ACF up to about 10 lags. Using this, we decide to implement a range of model orders. For the AR component, we choose to begin with AR = 20 and end with AR = 60 in increments of 5. For the MA component, we choose to begin with MA = 1 and end with MA = 9 in increments of 2. We implement all possible combinations of these ARMA orders (N = 45).

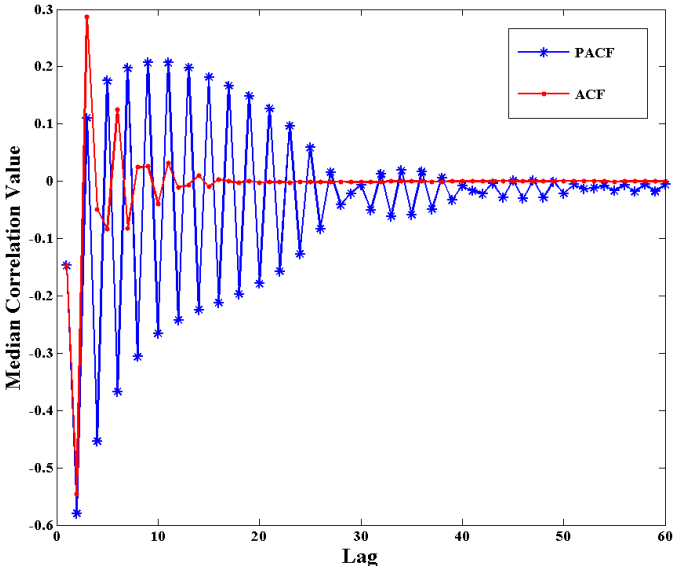

Fig. 4: *ACF and PACF of MEG data after first differencing*

Final Model Order Selection

For each of the 45 model order combinations, the White Noise Test was calculated on the residuals. In the case of the test dataset, there were 4 channels that could not be modeled in each of the model order combinations. Channels with greater than 5 extreme values, and thus excluded, were relatively consistent across model order combinations with a range of 26-29 channels (mean = 27.24, ~5% of the test dataset) per combination. Additionally, residual series were not significantly different from normal ($\alpha = 0.01$). The remaining attributes are shown in Table 3 for up to AR = 50 (AR = 55 and 65 showed similar patterns). In the table, unique channels is the count of unique channels across the tACF, tPACF, cMEAN, and cVAR attributes.

The results in Table 3, show multiple model order combinations provide low counts on several attributes, indicating more than one usable model order combination. However, there are two important patterns that emerge. First, as the AR increases (holding the MA constant), the ACF and PACF counts generally decrease. Second, as the MA increases (holding the AR constant), the ACF and PACF counts generally decrease. Taken together, there exists an ideal candidate model, namely ARIMA(30,1,3). This model order exhibits two qualities to use in evaluating model orders: it is within the lowest on all attribute counts as compared to other model orders, and among those with the lowest attribute values, it has the lowest model orders.

From an analyst perspective, an ideal candidate model is informed by the future analysis to be performed. Basically, when choosing the ideal candidate model, the next stage of analysis needs to be considered and used to identify the ideal candidate model. For instance, if the next stage of analysis is to calculate all possible pairwise partial correlation coefficients between each channel for ±50 lags, then the model order of choice should have an AR $\geq$ 50 or at a minimum, the tACF and tPACF attributes of the residuals need to be examined up to 50 lags. In general, choosing an ideal candidate model will be based on several factors including, but not limited to, the choice of method-solver, future analytic needs, and degree of "whiteness" desired.

We compare our ACF thresholding to two autocorrelation tests, the Ljung-Box and Breusch-Godfrey statistics for up to AR lags, tested at $\alpha = 0.001$, for each residual series. Figure 5 shows a bar graph of the Ljung-Box and ACF counts. The Ljung-Box statistic is calculated at three levels, with degrees of freedom (df) equalling AR, min(20, N-1) as suggested by [Box], and $\ln(N)$ as suggested by [Tsay]. Each bar is for one model order combination with the same labeling as in the first column of Table 3. The bar length is the sum of the elements in the model order combination for the given statistic. Each bar shows different colors for each statistic and the relative contribution each statistic makes to the total sum for that model order combination. The Breusch-Godfrey, in place of the Ljung-Box, showed similar results. It can be seen that the Ljung-Box corresponds well to our ACF thresholding when the df equal the AR order but fails to identify autocorrelation using either of the suggested df. Finally, the Breusch-Godfrey and Ljung-Box statistics are compared in terms of the percent of residual series failing each statistic (Table 4).

#	Model Orders	tACF	tPACF	cMEAN	cVAR	Unique Channels
1	20-1-1	570	570	0	12	570
2	20-1-3	54	54	7	12	70
3	20-1-5	31	31	6	12	49
4	20-1-7	27	27	7	12	46
5	20-1-9	15	15	8	12	34
6	25-1-1	569	569	0	12	569
7	25-1-3	16	16	7	10	33
8	25-1-5	31	31	6	12	49
9	25-1-7	10	10	9	12	31
10	25-1-9	3	3	10	12	24
11	30-1-1	569	569	6	11	569
12	30-1-3	5	5	8	13	26
13	30-1-5	7	7	8	11	26
14	30-1-7	3	3	10	12	25
15	30-1-9	3	3	10	12	23
16	35-1-1	563	563	2	11	563
17	35-1-3	8	8	9	12	28
18	35-1-5	3	3	8	12	23
19	35-1-7	6	6	8	12	26
20	35-1-9	0	0	7	12	19
21	40-1-1	529	529	8	11	530
22	40-1-3	30	30	7	12	47
23	40-1-5	1	1	7	12	20
24	40-1-7	8	8	8	12	27
25	40-1-9	1	1	7	11	19
26	45-1-1	222	222	7	10	234
27	45-1-3	6	6	9	11	26
28	45-1-5	0	0	8	12	20
29	45-1-7	3	3	7	12	22
30	45-1-9	2	2	7	12	21
31	50-1-1	15	15	7	11	33
32	50-1-3	0	0	7	11	18
33	50-1-5	0	0	7	12	19
34	50-1-7	0	0	7	12	19
35	50-1-9	0	0	9	12	21

TABLE 3: *Attributes for the White Noise Test shown for incrementing model order combinations, listed as thresholded ACF and PACF (tACF, tPACF), constant mean (cMEAN) and constant variance (cVAR), and the number of unique channels across the attributes.*

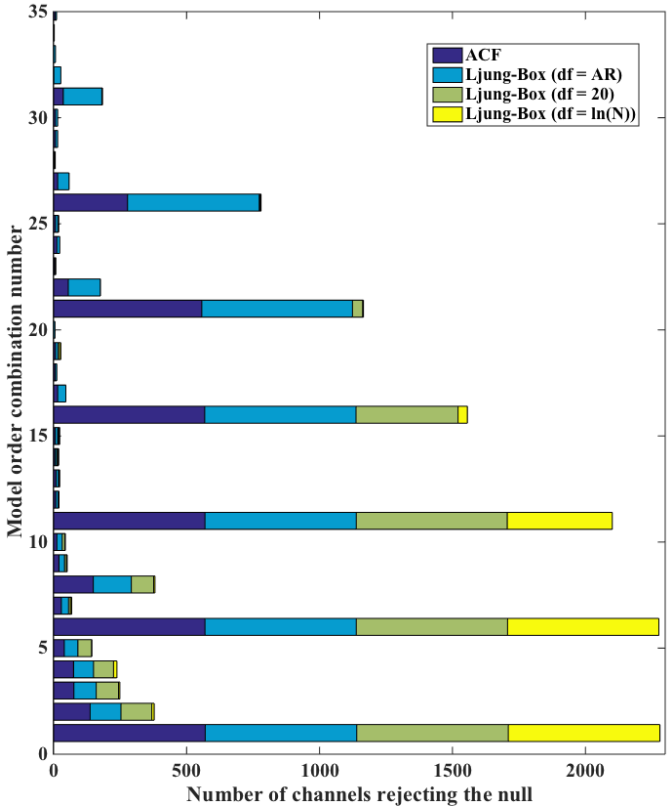

Fig. 5: *ACF and Ljung-Box Attributes Compared*

df	% =	% ≠ by 1	% ≠ by > 1
AR	55.6	20.0	24.4
20	77.8	6.7	15.6
ln(N)	84.4	11.1	4.4

TABLE 4: *Breusch-Godfrey test compared to Ljung-Box test*

MEG Dataset Evaluation

Finally, using ARIMA(30,1,3), we apply the White Noise Test procedure to the full MEG dataset. One channel at each stage of modeling is shown in Figure 6. Descriptive statistics on each of the attributes for the full MEG data are shown in Table 5 and the overall percent of channels removed per subject is shown in Figure 7. One subject had over 200 channels removed, likely due to errors within the recording, and was excluded from Table 5.

Conclusion

In this paper, we presented an expansion on the Box-Jenkins methodology to ARIMA modeling. First, during model identification and selection, we implement two complementary tests (KPSS and ADF) to establish stationarity. Using these stationary series, we use median correlation values at each lag of the ACF and PACF across 600 channels to identify a range of AR(p) and MA(q) order to implement combinatorially. This methodology allows for examining multiple time series simultaneously to determine a valid model order for the majority of time series in a dataset. Second, during parameter value estimation, we utilize the *Statsmodels* package to find the method-solver combination that provides good metrics (model reliability, validity of residuals, and time efficient) for long time series. We found the `css-lbfgs` to outperform all other method-solver combinations on these metrics. Third, during model evaluation, we present a novel approach (White Noise Test: Figure 1) to diagnostic checking of ARIMA modeling for long time series, which evaluates residual series based on stationarity and nonautocorrelation (i.e., "whiteness"). Using this approach, we identify the ideal candidate model for our dataset to be ARIMA(30,1,3). Applying this model to the full MEG dataset, we find an average of 20.55 channels removed from the White Noise Test (i.e., fail to establish "whiteness"), which is about 8.3% of the dataset. Overall, using our approach, an investigator can perform ARIMA modeling and evaluate candidate models with ease for large datasets and datasets containing long time series.

Step	Min	Max	Median	Mean	Std Dev
xVAL	0	60	1	9.67	16.63
Normal	0	0	0	0.00	0.00
tACF	0	51	0	2.53	8.12
tPACF	0	0	0	0.00	0.00
cMEAN	0	8	0	0.20	1.15
cVAR	0	40	4	7.24	9.05
Channels Removed	1	85	10	20.55	21.34

TABLE 5: *Results of White Noise Test on full dataset, with the steps listed as extreme values (xVAL), normality, thresholded ACF and PACF (tACF, tPACF), constant mean (cMEAN) and constant variance (cVAR), and the number of channels removed as a result.*

References

[Bartlett] Bartlett, M.S. 1946. "On the theoretical specification and sampling properties of autocorrelated time-series." *Journal of the Royal Statistical Society*, 8.1, 27-41.

[Box] Box, G. and Jenkins, G. 1976. "Time series analysis: forecasting and control." Holden Day, San Francisco, 2nd edition.

[Breusch] Breusch, T.S. 1978. "Testing for autocorrelation in dynamic linear models", *Australian Economic Papers*, 17, 334–355.

[Durbin] Durbin, J. and Watson, G.S. 1971. "Testing for serial correlation in least squares regression III", *Biometrika*, 58.1, 1–19.

[Godfrey] Godfrey, L.G. 1978. "Testing against general autoregressive and moving average error models when the regressors include lagged dependent variables", *Econometrica*, 49, 1293–1302.

[Hannan] Hannan, E.J. and Rissanen, J. 1985. "Recursive estimation of mixed autoregressive-moving average order". *Biometrika*, 69.1, 81-94.

[Kwiatkowski] Kwiatkowski, D., Phillips, P.C.B., Schmidt, P., Shin, Y. 1992. "Testing the null hypothesis of stationarity against the alternative of a unit root", *Journal of Econometrics*, 54, 159ñ178

[Ljung] Ljung, G.M. and Box, G.P. 1978. "On a Measure of a Lack of Fit in Time Series Models", *Biometrika*, 65.2, 297–303.

[McKinney] McKinney, W., Perktold, J., Seabold, S. 2011. "Time series analysis in python with statsmodels", *Proceedings of the 10th Python in Science Conference*, 96-102.

[Pankratz] Pankratz, A. 1991. "Forecasting with dynamic regression models", John Wiley and Sons, New York.

[Said] Said, S.E. and Dickey, D. 1984. "Testing for unit roots in autoregressive moving-average models with unknown order", *Biometrika*, 71, 599-607.

[Seabold] Seabold, S. and Perktold J. 2010. "Statsmodels: econometric and statistical modeling with python", *Proceedings of the 9th Python in Science Conference*, 57-61.

[Tsay] Tsay, R.S. 2005. "Analysis of Financial Time Series", John Wiley & Sons, Inc., Hoboken, NJ.

[Tukey] Tukey, J.W. 1977. "Exploratory data analysis", Addison-Wesley, Reading, MA.

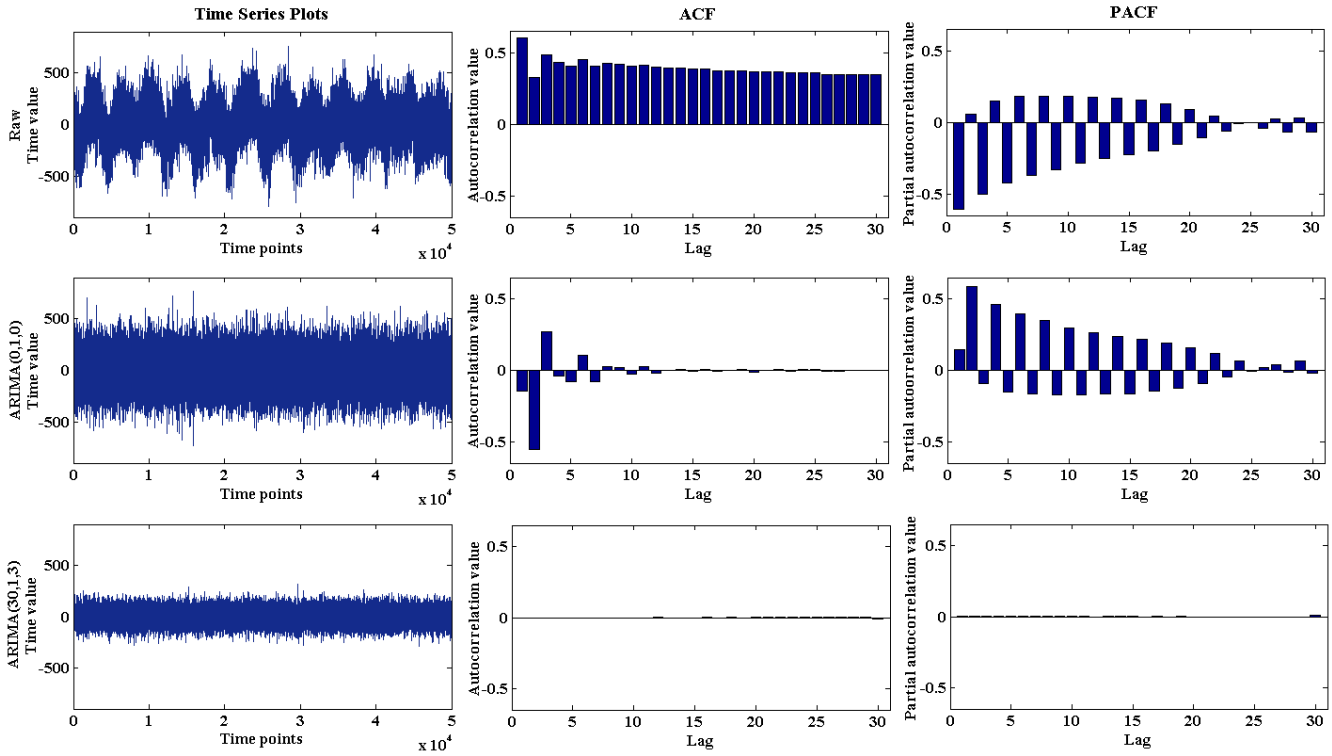

Fig. 6: *Raw, differenced, and ARIMA(30,1,3) series with corresponding ACF and PACF.*

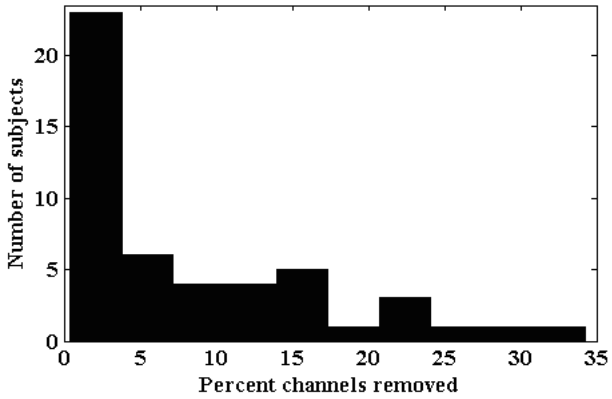

Fig. 7: *Percent of channels removed per subject.*

[Yaffee] Yaffee, R.A. and McGee, M. 2000. "Introduction to time series analysis and forecasting: with applications of SAS and SPSS", Academic Press.