

Using ARIMA modelling techniques to predict United States' military spending

Tanha Kate

FA51 Spring 2018

1. Introduction

This paper forecasts the United States' military spending in absolute terms and as a percentage of Gross Domestic Product (GDP) based on autoregressive integrated moving average (ARIMA) models. Military spending data for the period 1949-2016 collected by the Stockholm International Peace Research Institute (SIPRI) is used (SIPRI Military Expenditure Database, n.d.). For each forecast period, the ARIMA model outputs the forecast value, a standard error estimate for the forecast value, and the lower and upper limits for a 95% prediction interval. Results obtained reveal that the ARIMA model has a strong potential for short-term prediction.

2. Methodology

2.1. Data Collection

I collected military spending data in USD for the period 1949-2016 from the Stockholm International Peace Research Institute (SIPRI) Military Expenditure Database. 'All figures for the USA are for the financial year (1 Oct. of the previous year-30 Sep. of the stated year) rather than a calendar year' ('Military Budget in the United States,' n.d.)

Military spending is measured in both absolute terms and as a share of GDP. On the one hand, absolute levels of expenditure are important for military planning and war outcomes. In the example, the US spending 10% of its GDP fighting a war is likely to defeat a low or middle-income country (e.g., Bangladesh) spending 50% or more of its GDP. Conversely, percentages of GDP allow inferences about the priorities of the United States. The study predicts military expenditure from the previous 20 years extrapolates into the next five years from 2016.

2.2. Exploratory Analysis

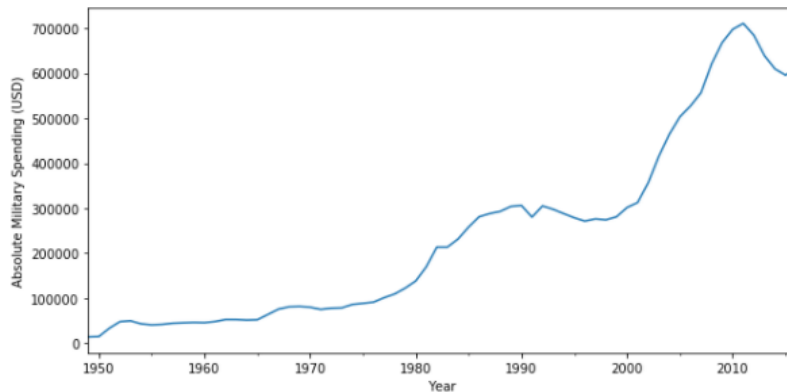


Figure 1. Line graph depicting US military expenditure in absolute terms for the 1949-2016 period.

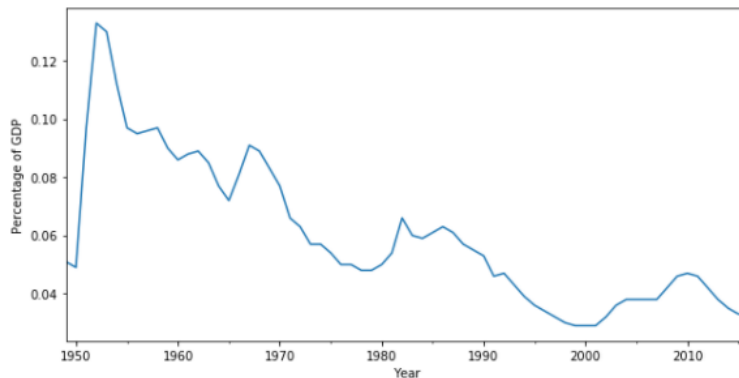


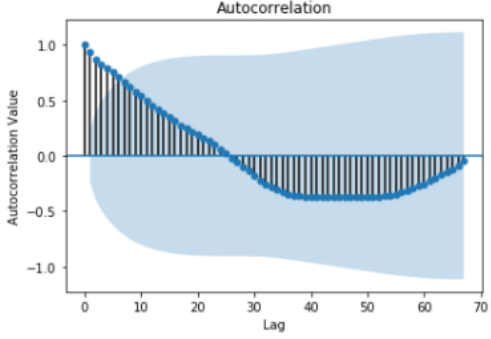
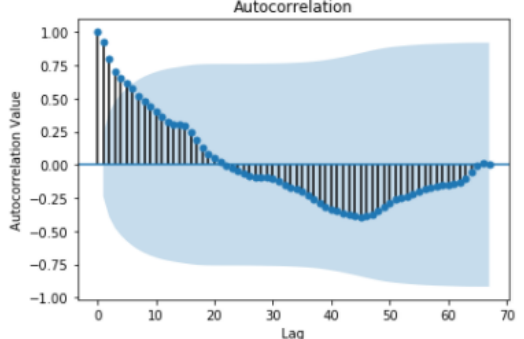
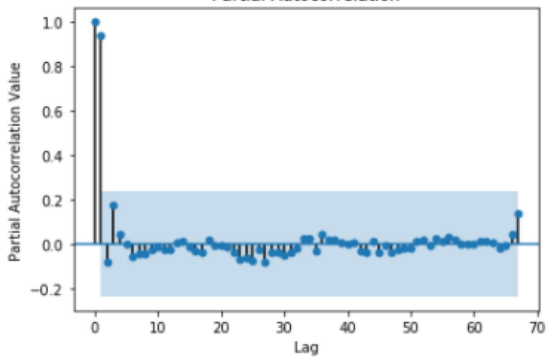
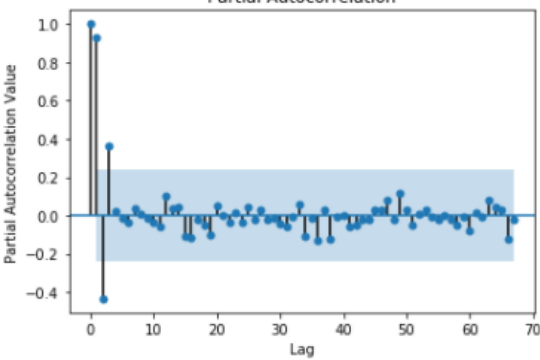
Figure 2. Line graph depicting US military expenditure in terms of percentage GDP for the 1949-2016 period.

The first step in applying ARIMA modelling is to check for stationarity, i.e. ‘the series remains fairly constant overtime and shows a constant variance in its fluctuations over time’ (ARIMA, n.d.). Almost by definition, the datasets aren’t heavily seasonal since military spending is annual. However, a continuous curved upward (Figure 1) and downward trend (Figure 2) violates the stationarity. We will perform various tests to determine if differencing¹ is necessary².

¹ The differenced series is the change between each observation in the original series and is used to stationarize time series data for model estimations.

² ARIMA procedures are applicable only for stationary data. A non-stationary process must be transformed and changed to a suitable stationary form. (Stage et al., n.d.)

2.3. Model Identification

Absolute Terms	Percentage of GDP Terms
	
<p>Both the autocorrelation³ distributions display a sinusoidal curve, and several autocorrelations fall outside the i.i.d. hypothesis 95% confidence interval (highlighted region), i.e., they seem significant (Boshnakov, n.d.), providing evidence towards non-stationarity. However, 5% of the results are likely to be statistically significant purely by chance because of Type I error.</p>	
	

³ The autocorrelation (ACF) is a summary of the relationship between an observation in a time series with observations at prior time steps (lags). The autocorrelation³ of stationary data drops to zero relatively quickly, while it decreases slowly for non-stationary data (“8.1 Stationarity and differencing,” n.d.).

In both PACF⁴ graphs, there are significant correlations at the first or second lag, providing further evidence towards non-stationarity.

An Augmented Dickey Fuller (ADF) test⁵ confirms that differencing is necessary to stationarize the data:

Null Hypothesis (H ₀):	$\gamma = 0$ <p>The time series has a unit root and is non-stationary.</p>
Alternate Hypothesis (H _A):	$\gamma \leq 0$ <p>The time series doesn't have a unit root and is stationary.</p>

*Results*⁶:

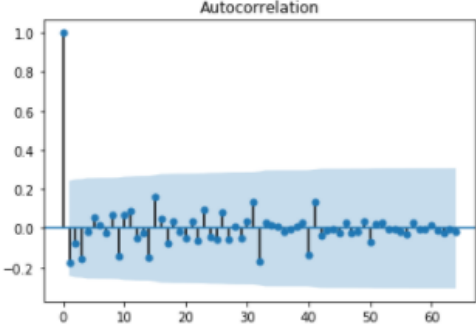
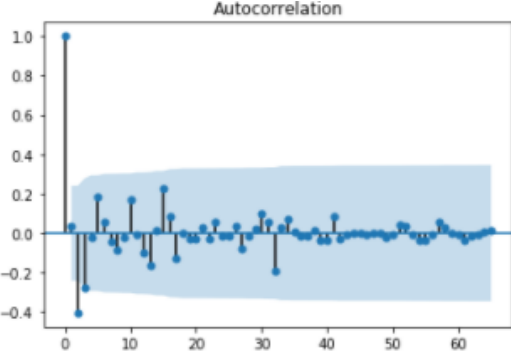
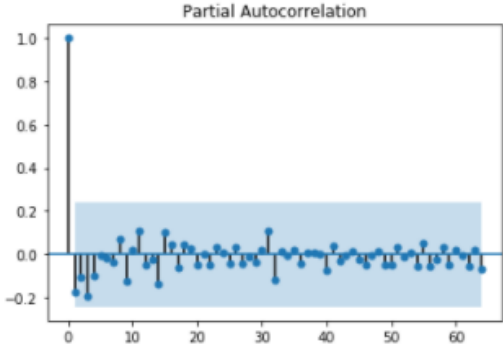
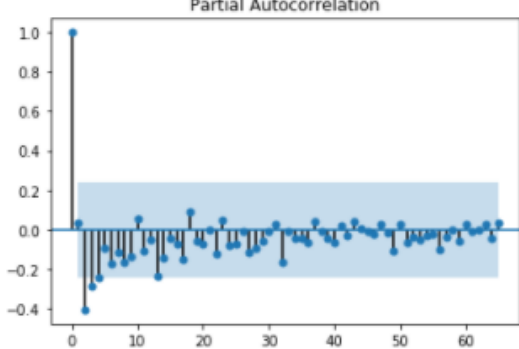
$$\text{Absolute (USD)} = 0.171317 > 0.05$$

$$\% \text{ GDP} = 0.882556 > 0.05$$

⁴ 'Partial autocorrelation (PACF) is a summary of 'the relationship between an observation in a time series with observations at prior time steps' (lag) after adjusting for the effects of intermediate observations. (Brownlee, 2017).

⁵ 'The ADF test tests whether a time series variable is non-stationary and possesses a unit root' (Unit root test, 2018). A unit root is a stochastic trend in a time series, which if significant indicates an unpredictable, random pattern and further modelling isn't possible (Stephanie, n.d.).

Since p-values are greater than 0.05, we fail to reject the null hypothesis that the data is non-stationary and has a unit root. Before forecasting models can be applied, both series are transformed into a second order difference to establish stationarity, and the ACF and PACF are displayed below:

Absolute Terms	Percentage of GDP Terms
 <p>The plot shows the autocorrelation function for the absolute terms. The y-axis ranges from -0.2 to 1.0, and the x-axis ranges from 0 to 60. A light blue shaded region represents the 95% confidence interval, which is approximately between -0.1 and 0.3. The autocorrelation starts at 1.0 at lag 0 and drops sharply to near zero by lag 10, remaining within the confidence interval for the rest of the lags.</p>	 <p>The plot shows the autocorrelation function for the percentage of GDP terms. The y-axis ranges from -0.4 to 1.0, and the x-axis ranges from 0 to 60. A light blue shaded region represents the 95% confidence interval, which is approximately between -0.2 and 0.3. The autocorrelation starts at 1.0 at lag 0 and drops sharply to near zero by lag 10, remaining within the confidence interval for the rest of the lags.</p>
<p>The ACF of stationary data drops to zero relatively quickly, and subsequent observations fall inside the 95% confidence interval (highlighted region), i.e., i.i.d. hypothesis holds, and the data is stationary (Boshnakov, n.d.). A similar trend is observed for the PACF below:</p>	
 <p>The plot shows the partial autocorrelation function for the absolute terms. The y-axis ranges from -0.2 to 1.0, and the x-axis ranges from 0 to 60. A light blue shaded region represents the 95% confidence interval, which is approximately between -0.1 and 0.2. The partial autocorrelation starts at 1.0 at lag 0 and drops sharply to near zero by lag 10, remaining within the confidence interval for the rest of the lags.</p>	 <p>The plot shows the partial autocorrelation function for the percentage of GDP terms. The y-axis ranges from -0.4 to 1.0, and the x-axis ranges from 0 to 60. A light blue shaded region represents the 95% confidence interval, which is approximately between -0.2 and 0.2. The partial autocorrelation starts at 1.0 at lag 0 and drops sharply to near zero by lag 10, remaining within the confidence interval for the rest of the lags.</p>

ADF p-values⁷ for second-order differenced data is less than 0.05. Therefore, the null hypothesis is rejected in favor of the stationarity of the dataset.

Datasets are split into training and test data for the Cohen's d test (below). Near-zero values indicate there is no practically significant⁸ difference between the means in the two partitions in each data set. Hence, the time series displays a predictable, stationary trend.

	Absolute Military Spending	% GDP
Cohen's d	0.018	0.032
Interpretation (Ellis, n.d.)	No effect	No effect

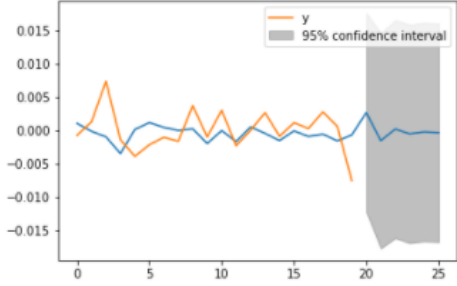
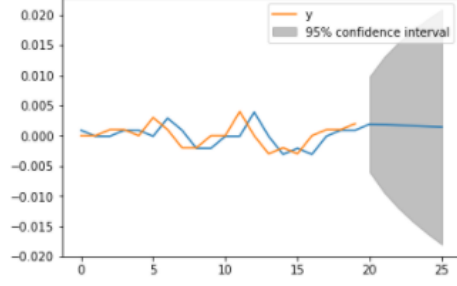
2.3.1. Model Estimation

An ARIMA model is fit on the training dataset by evaluating parameters which yield the minimum error score for predictions compared to expected values. ARIMA⁹ is appropriate for this study because it can non-stationary datasets after appropriate differencing (Ho et al., 1998). ARIMA models require three distinct integers (p, d, q) to account for the effect of past values, the amount of differencing and the linear combination of the error values observed at previous time points in the past respectively.

⁷ P-value (second differenced log-form military spending) = 0.00, P-value (second differenced % GDP spending) = 0.00

⁸ **#significance:** To evaluate the stationarity of the time series data, I used an appropriate test followed my appropriate differencing. The practical significance of the difference in means between two partitions of each dataset is further assessed to confirm stationarity after second order differencing.

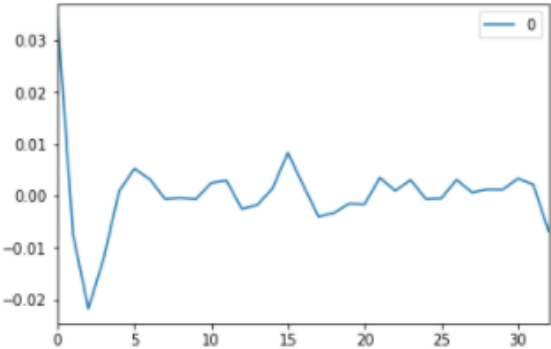
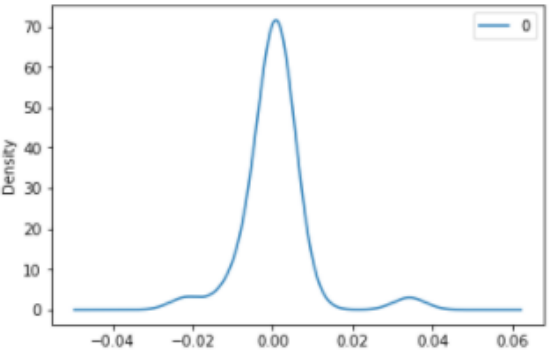
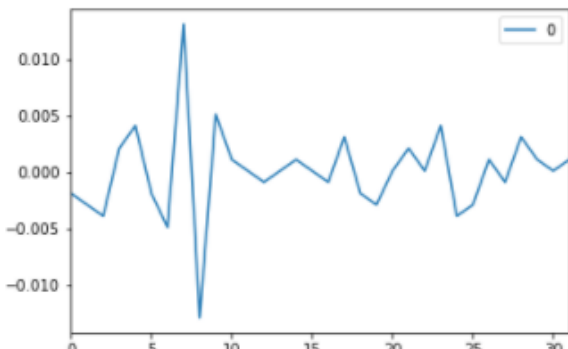
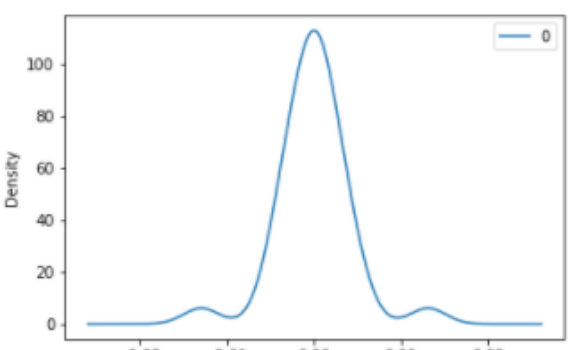
⁹ ARIMA, which stands for Autoregressive Integrated Moving Average models, incorporates both autoregressive and moving average features, along with detrending of the data. The AR part means that forecasted values depend on its past values and the MA part means that values depends on a linear combination of random error terms which follow a white noise process.

	Absolute Military Spending	% GDP
Model ¹⁰	ARIMA (1,0,0) 	ARIMA(0,1,0) 
Forecast	0.003	0.005
Expected	0.003	0.002
Standard Error	0.008	0.004
95% Prediction Interval	(-0.012, 0.018)	(-0.006, 0.010)

¹⁰ Orange: Actual observations, Blue: ARIMA forecasts

2.3.2. Model Diagnostics

Two important assumptions of the ARIMA model are that ‘1) the residuals (observed minus predicted values) are normally distributed, and 2) that they are independent of each other’ (Statistica Help | Example 2: Single Series Arima, n.d.). We check these assumptions by analyzing the residuals below:

Absolute Military Spending	% GDP
 	 
<p>The residual plot¹¹ for military spending appears stationary and does not display obvious seasonality or trend behavior. Conversely, the % GDP residuals display non-stationarity in the beginning, so prediction performance will depend on whether the timeframe considered</p>	

¹¹ **#distributions:** I diagnosed the normality and independence of forecasts by evaluating the residual distribution. Subsequently, I interpreted the shape and central tendency to assess the performance of the ARIMA models.

corresponds with residual stationarity. Also, the kernel density estimation of both residuals show near normal distribution, with a zero-mean suggesting there is no bias¹² with the model prediction.

3. Results

Using the ARIMA models, we forecasted the previous 20 years and the next five years. The 95% confidence intervals consist of an upper and a lower limit between which future military spending is expected to lie 95% percent of the time¹³. Since ARIMA observations depend on previous observations and errors, the level of certainty of the % GDP model decreases with each step and the size of prediction interval increases. Conversely, the absolute military spending model confers greater certainty because the prediction interval is constant. The higher stationarity of the residual plot for absolute military spending is consistent with this interpretation.

¹² **#induction:** I generalized the strength of the respective ARIMA models based on a residual plot and kernel density estimation and assessed to what extent they fulfill necessary assumptions. Furthermore, I specified the conditions for high-prediction performance and visually assessed the prevalence of bias according to the zero-mean.

¹³ **#confidenceinterval:** I specified upper and lower limits for future military spending values in the next five years, with an accurate interpretation of the level of uncertainty associated with each interval. In addition, the prediction interval is visualized around the forecasted values to confer a spatial feeling for parameter estimates.

Extra Section: Conditional Probability

In California, 44.8% of registered voters are Democrats, 25.9% are Republicans, and 24.5% are Independent ('California Voter and Party Profiles', n.d.).

45% percent of independents and 42% of Democrats are college graduates, compared to slightly fewer Republicans (35%) ('California Voter and Party Profiles', n.d.).

Assume the following percentage support for cuts in military spending:

	College Student	Not College Student
Democrat	71%	67%
Republican	34%	26%
Independent	42%	53%

If a voter from California is chosen at random, and he/she supports cuts in military spending, what are the probability that he/she is a college student and a democrat?

Notation	Probability
$P(D)$	Democrat
$P(R)$	Republican
$P(I)$	Independent
$P(C)$	College Student
$P(\neg C)$	Not a College Student
$P(S)$	Supporting cuts in military spending
$P(\neg S)$	Opposing cuts in military spending increase
$P(C \cap D)$	College Student and Democrat
$P(S (C \cap D))$	Supporting cuts in military spending given College Student and Democrat

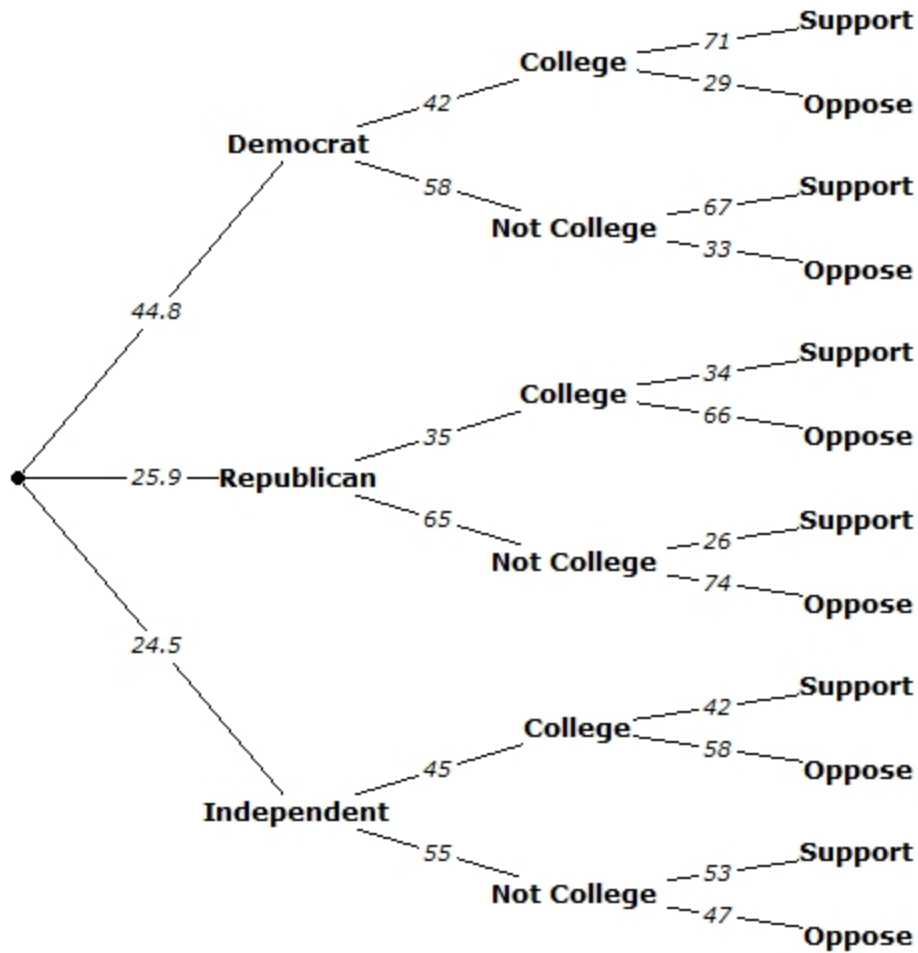


Figure 3. Probability tree diagram is depicting support for cuts in military expenditure. Numerical figures represent percentages of California's voting population.

Solution:

$$P(S) = ((P(D) * P(C|D) * P(S|(C|D))) + (P(D) * P(\neg C|D) * P(S|(\neg C|D)))) + ((P(R) * P(C|R) * P(S|(C|R))) + (P(R) * P(\neg C|R) * P(S|(\neg C|R)))) + ((P(I) * P(C|I) * P(S|(C|I))) + (P(I) * P(\neg C|I) * P(S|(\neg C|I))))$$

$$= ((0.448 * 0.42 * 0.71) + (0.448 * 0.58 * 0.67)) + ((0.259 * 0.35 * 0.34) + (0.259 * 0.65 * 0.26)) + ((0.245 * 0.45 * 0.42) + (0.245 * 0.55 * 0.53))$$

$$= 0.500$$

$$\begin{aligned}P(C \cap D) &= P(C|D) * P(D) \\&= 0.42 * 0.448 \\&= 0.188\end{aligned}$$

$$P(S|C \cap D) = 0.71$$

$$\begin{aligned}P((C \cap D)|S) &= \frac{P(S|(C \cap D)) * P(C \cap D)}{P(S)} \\&= \frac{0.71 * 0.188}{0.500} \\&= 0.267\end{aligned}$$

Answer¹⁴: There is a 26.7% chance that a randomly selected Californian voter is a college student and a democrat given they support cuts in military spending.

(759 words excluding
titles, in-text citations and footnotes)

¹⁴ **#probability**: I applied my understanding of conditional probabilities to construct a problem, broke down the steps to the solution and used a tree diagram appropriately to find the correct solution.

Bibliography:

8.1 Stationarity and differencing. (n.d.). Retrieved March 9, 2018, from [/fpp/8/1](#)

AUTOREGRESSIVE INTEGRATED MOVING AVERAGE MODELS (ARIMA). (n.d.).

Retrieved March 9, 2018, from <http://www.forecastingsolutions.com/arima.html>

Boshnakov, G. (n.d.). *Autocorrelations and white noise tests*. The University of Manchester.

Retrieved March 6, 2018, from [https://cran.r-](https://cran.r-project.org/web/packages/sarima/vignettes/white_noise_tests.pdf)

[project.org/web/packages/sarima/vignettes/white_noise_tests.pdf](https://cran.r-project.org/web/packages/sarima/vignettes/white_noise_tests.pdf)

Brownlee, J. (2017, February 6). A Gentle Introduction to Autocorrelation and Partial

Autocorrelation. Retrieved March 8, 2018, from

<https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/>

California Voter and Party Profiles. (n.d.). Retrieved March 9, 2018, from

<http://www.ppic.org/publication/california-voter-and-party-profiles/>

Ellis, P. (n.d.). Thresholds for interpreting effect sizes. Retrieved March 8, 2018, from

http://www.polyu.edu.hk/mm/effectsizefaqs/thresholds_for_interpreting_effect_sizes2.html

Ho, S. L., & Xie, M. (1998). The use of ARIMA models for reliability forecasting and analysis. *Computers & Industrial Engineering*, 35(1), 213–216.

[https://doi.org/10.1016/S0360-8352\(98\)00066-7](https://doi.org/10.1016/S0360-8352(98)00066-7)

Military Budget In United States – American Military Spending. (n.d.). Retrieved from
<http://militarybudget.org/united-states/>

SIPRI Military Expenditure Database | SIPRI. (n.d.). Retrieved March 10, 2018, from
<https://www.sipri.org/databases/milex>

Stage, F., & Statements, U. A. P. (n.d.). The ARIMA Procedure.

STATISTICA Help | Example 2: Single Series ARIMA. (n.d.). Retrieved March 10, 2018,
from
<http://documentation.statsoft.com/STATISTICAHelp.aspx?path=TimeSeries/TimeSeries/Examples/Example2SingleSeriesARIMA>

Stephanie. (n.d.). ADF — Augmented Dickey Fuller Test. Retrieved March 9, 2018, from
<http://www.statisticshowto.com/adf-augmented-dickey-fuller-test/>

Stephanie. (n.d.). Unit Root: Simple Definition, Unit Root Tests. Retrieved March 9, 2018,
from <http://www.statisticshowto.com/unit-root/>

Unit root test. (2018, January 26). In *Wikipedia*. Retrieved from
https://en.wikipedia.org/w/index.php?title=Unit_root_test&oldid=822483302

Wang, G. C. S., & Jain, C. L. (2003). *Regression Analysis: Modeling & Forecasting*. Institute
of Business Forec.

What is the Ljung-Box q (LBQ) statistic? (n.d.). [mtbconcept]. Retrieved March 9, 2018, from
<https://support.minitab.com/en-us/minitab/18/help-and-how-to/modeling-statistics/time-series/supporting-topics/diagnostic-checking/what-is-the-ljung-box-q-statistic/>

Technical Appendix

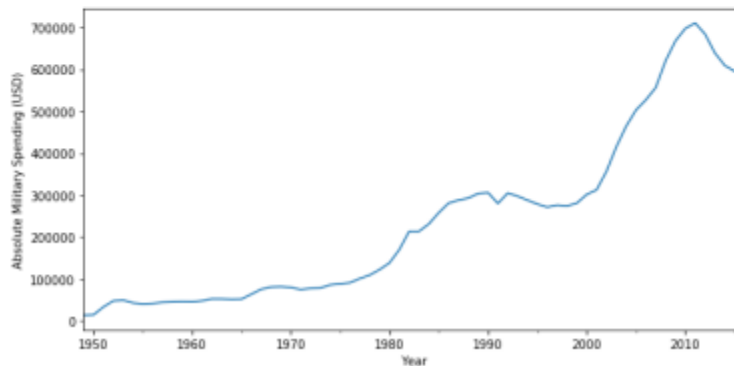
1. Absolute Military Spending ARIMA

```
In [34]: from pandas import Series
import numpy as np
from matplotlib import pyplot as plt
from statsmodels.tsa.stattools import acf, pacf
import statsmodels.tsa.stattools as ts
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima_model import ARIMA
from scipy import stats
import scipy
import pandas as pd
import math
from statsmodels.graphics.api import qqplot

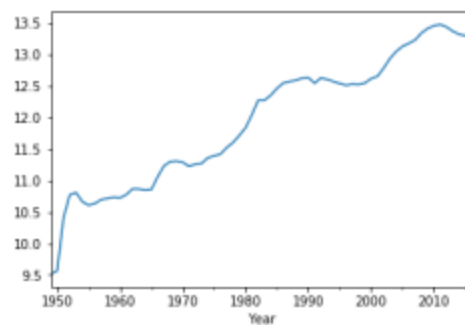
from sklearn.metrics import mean_squared_error
from numpy import std, mean, sqrt
from statsmodels.stats import diagnostic as diag
```

Exploratory Analysis

```
In [35]: #plotting the raw time series data
series = Series.from_csv('military.csv', header=0)
plt.ylabel('Absolute Military Spending (USD)')
series.plot(figsize=(10,5))
plt.show()
```



```
In [36]: #Log-form manipulation to inform on relative changes and improve the efficiency of the model.
series = series.apply(np.log)
series.plot()
plt.show()
```

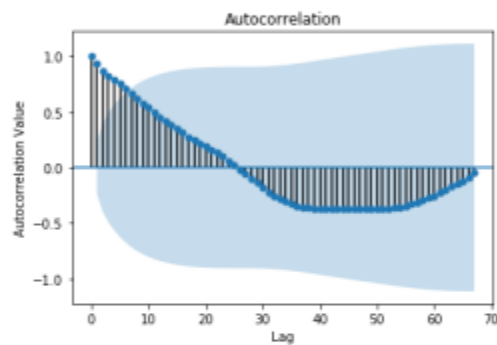


Stationarity Tests

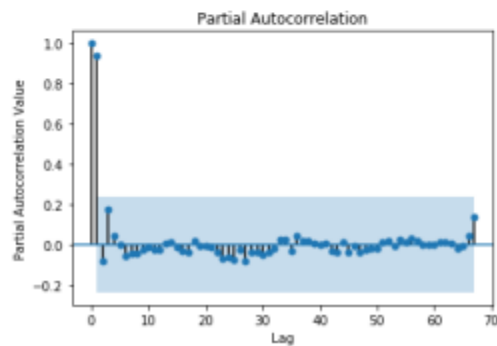
```
In [37]: #ACF and PACF tests for stationarity.
#The highlighted regions depict a 95% confidence interval,
#under the assumption that the time series is iid, suggesting that correlation
#values outside of this code are very likely a correlation and not a statistic
#al fluke.
#Confidence intervals for ACF are drawn as a cone.

plot_acf(series)
plt.xlabel('Lag')
plt.ylabel('Autocorrelation Value')
plt.show()

#steady decrease downwards
## If one or more lags pierce those dashed lines, then the lag(s) is significant
ntly different from zero and the series is not white noise.
```



```
In [38]: plot_pacf(series)
plt.xlabel('Lag')
plt.ylabel('Partial Autocorrelation Value')
plt.show()
```



Auamented Dickey-Fuller Test

```
In [21]: result = ts.adfuller(x, 2, regression='c') #maximum lag which is included in test
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: -0.546798
p-value: 0.882556
Critical Values:
1%: -3.535
5%: -2.907
10%: -2.591
```

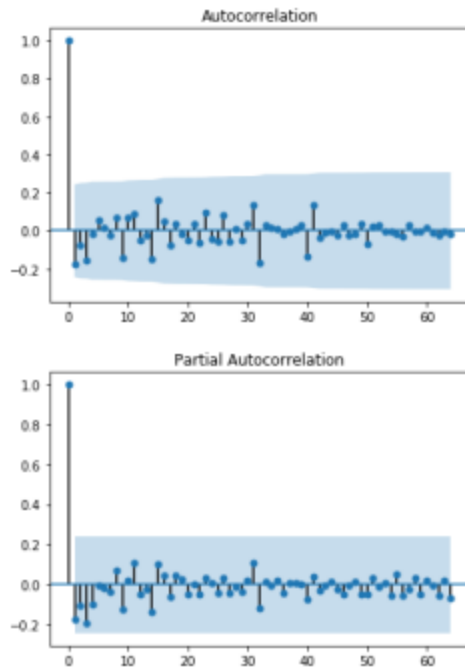
```
In [98]: x_diff = np.diff(x)
x_diff = np.diff(x_diff)
x_diff = np.diff(x_diff)
result = ts.adfuller(x_diff, 2, regression='c') #maximum lag which is included
in test
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: -9.102921
p-value: 0.000000
Critical Values:
1%: -3.541
5%: -2.909
10%: -2.592
```

```
In [108]: # acf and pacf
plot_acf(x_diff)
plot_pacf(x_diff)

# hist plot
count, division = np.histogram(x_diff)
plt.show()

#histogram
count, division = np.histogram(x_diff)
plt.show()
```



Practical signifiance

```
In [77]: #Practical signifiance for difference in means between training and test dat
a.

size = len(x_diff)//2
train, test= x_diff[:size], x_diff[size:]

def cohen_d(x,y):
    nx = len(x)
    ny = len(y)
    dof = nx + ny - 2
    return (mean(x) - mean(y)) / sqrt(((nx-1)*std(x, ddof=1) ** 2 + (ny-1)*std
(y, ddof=1) ** 2) / dof)

print(cohen_d(test,train))

0.0177790794304
```

Model Identification

```
In [78]: #Source: https://stackoverflow.com/questions/30901460/non-invertible-of-a-arim
a-model

p_values = [0, 1, 2, 4]
d_values = range(0, 3)
q_values = range(0, 3)

pdq = []
stderr = []

for p in p_values:
    for d in d_values:
        for q in q_values:
            order = (p,d,q)
            pdq.append(order)
            history = [x for x in train]
            predictions = []
            try:
                for t in range(len(test)):
                    model = ARIMA(history, order)
                    model_fit = model.fit(dispatch=0)
                    output = model_fit.forecast()
                    yhat = output[0]
                    predictions.append(yhat)
                    obs = test[t]
                    history.append(obs)
                error = mean_squared_error(test, predictions)
                stderr.append(error)

            except: # ignore the error and go on

                pass
```

```
In [81]: keys = pdq
values = stderr
d = dict(zip(keys, values))
print (d)

{(0, 0, 0): 2.9270637251015354e-06, (0, 0, 1): 3.9560322657175338e-06, (0, 0,
2): 2.8977320818170993e-06, (0, 1, 0): 1.1617086909392845e-05, (0, 1, 1): 3.0
984474614688966e-06, (0, 1, 2): 7.7248472306088377e-06, (0, 2, 0): 3.84947578
28279521e-05, (0, 2, 1): 1.2555802040464861e-05, (0, 2, 2): 3.383762245727189
1e-06, (1, 0, 0): 2.5902093332326673e-06, (1, 0, 1): 6.8999283052727381e-06,
(1, 0, 2): 2.0930473825749393e-05, (1, 1, 0): 3.351523538509614e-06, (1, 1,
1): 4.4089501631587672e-06, (1, 1, 2): 1.863421723412184e-05, (1, 2, 0): 1.44
54719318780986e-05}
```

```
In [102]: min(d, key=d.get)
```

```
Out[102]: (1, 0, 0)
```

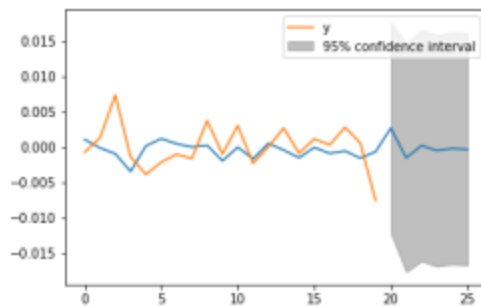
```
In [103]: #note that ARIMA model will fit to the differenced data
```

```
model = ARIMA(train, order=(1,0,0))
model_fit = model.fit(dispatch=False)
```

```
In [104]: forecast, stderr, conf = model_fit.forecast() #The forecast() function allows
the confidence interval to be specified.
#The alpha argument on the forecast() function specifies the confidence level.
It is set by default to alpha=0.05, which is a 95% confidence interval. This
is a sensible and widely used confidence interval.
#An alpha of 0.05 means that the ARIMA model will estimate the upper and lower
values around the forecast where there is a only 5% of the time the real valu
e will not be in that range.
print('Expected: %.3f' % test[0])
print('Forecast: %.3f' % forecast)
print('Standard Error: %.3f' % stderr)
intervals = [0.2, 0.1, 0.05, 0.01]
for a in intervals:
    forecast, stderr, conf = model_fit.forecast(alpha=a)
    print('%.1f%% Confidence Interval: %.3f between %.3f and %.3f' % ((1-a)
)*100, forecast, conf[0][0], conf[0][1]))

Expected: 0.003
Forecast: 0.003
Standard Error: 0.008
80.0% Confidence Interval: 0.003 between -0.007 and 0.012
90.0% Confidence Interval: 0.003 between -0.010 and 0.015
95.0% Confidence Interval: 0.003 between -0.012 and 0.018
99.0% Confidence Interval: 0.003 between -0.017 and 0.022
```

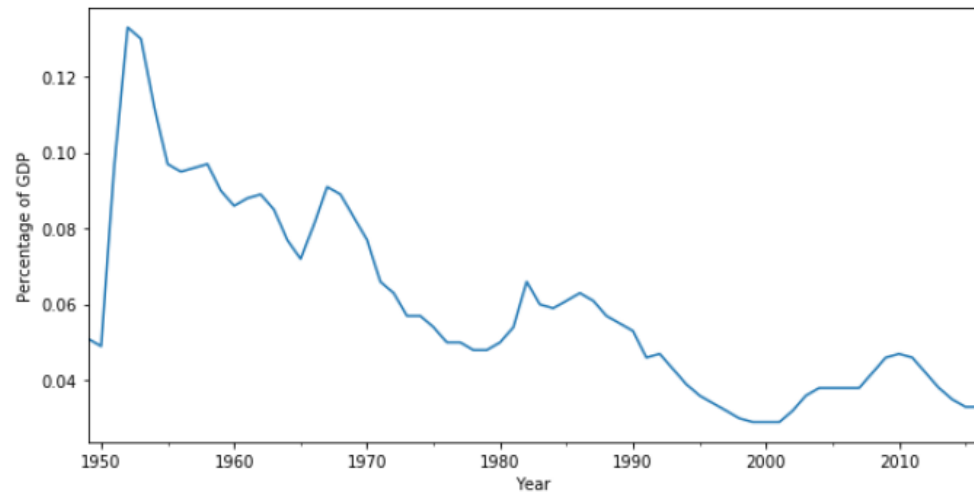
```
In [107]: model_fit.plot_predict(len(train)-20, len(train)+5)
plt.show()
```



2. Percentage GDP ARIMA

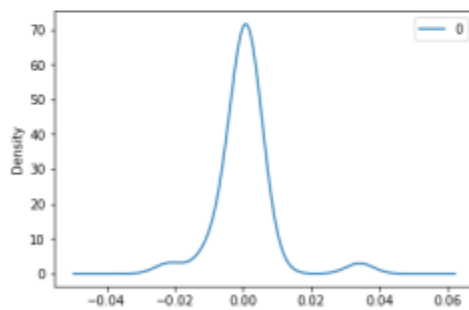
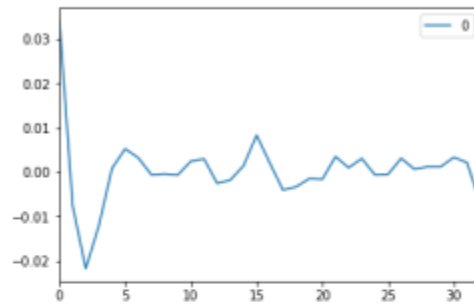
Exploratory Analysis

```
In [51]: series = Series.from_csv('gdp.csv', header=0)
plt.ylabel('Percentage of GDP')
series.plot(figsize=(10,5))
plt.show()
```



Model Diagnostics

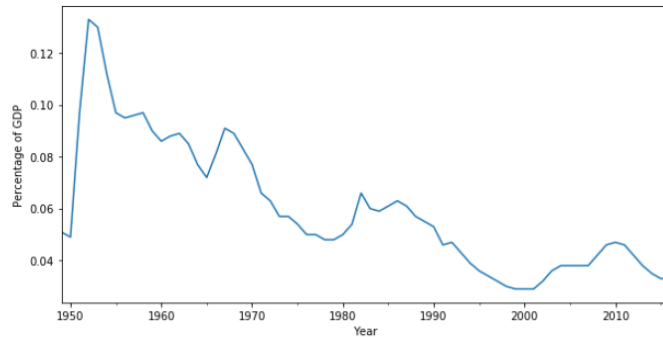
```
In [109]: # plot residual errors  
residuals = pd.DataFrame(model_fit.resid)  
residuals.plot()  
residuals.plot(kind='kde')  
plt.show()  
print(residuals.describe())
```



```
count  33.000000  
mean    0.000427  
std     0.008103  
min    -0.021742  
25%    -0.001621  
50%     0.000853  
75%     0.002958  
max     0.034111
```


Exploratory Analysis

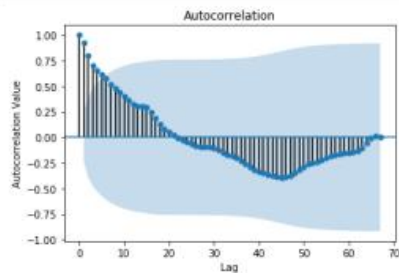
```
In [51]: series = Series.from_csv('gdp.csv', header=0)
plt.ylabel('Percentage of GDP')
series.plot(figsize=(10,5))
plt.show()
```



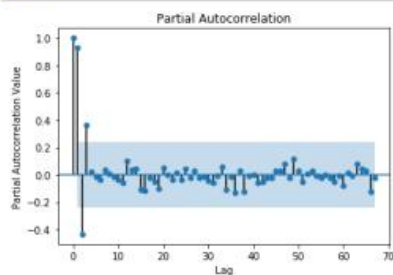
Stationarity Tests

```
In [52]: #ACF and PACF tests for stationarity.
#The highlighted regions depict a 95% confidence interval,
#under the assumption that the time series is iid, suggesting that correlation
#values outside of this code are very likely a correlation and not a statistic
#al fluke.
#Confidence intervals for ACF are drawn as a cone.

plot_acf(series)
plt.xlabel('Lag')
plt.ylabel('Autocorrelation Value')
plt.show()
```



```
In [53]: plot_pacf(series)
plt.xlabel('Lag')
plt.ylabel('Partial Autocorrelation Value')
plt.show()
```



```
In [56]: result = ts.adfuller(x,1, regression = 'c')
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: -2.302020
p-value: 0.171317
Critical Values:
1%: -3.534
5%: -2.906
10%: -2.591
```

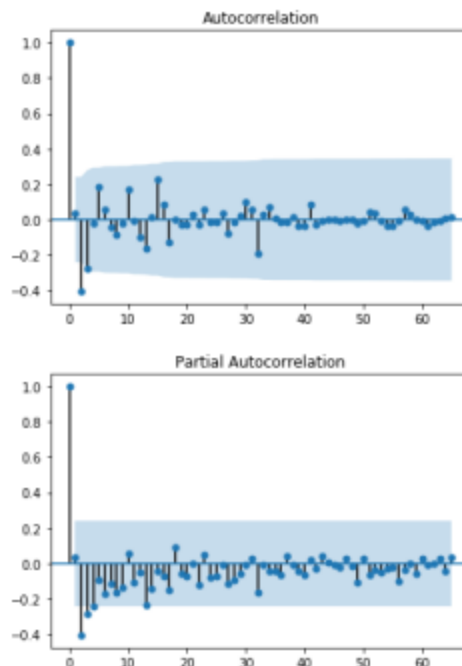
```
In [60]: x_diff = np.diff(x)
x_diff = np.diff(x_diff)
result = ts.adfuller(x_diff,1,regression='c') #maximum lag which is included i
n test
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: -9.425920
p-value: 0.000000
Critical Values:
1%: -3.537
5%: -2.908
10%: -2.591
```

```
In [99]: # acf and pacf
plot_acf(x_diff)
plot_pacf(x_diff)

# hist plot
count, division = np.histogram(x_diff)
plt.show()

#histogram
count, division = np.histogram(x_diff)
plt.show()
```



Practical Significance

```
In [67]: size = len(x_diff)//2
train, test= x_diff[:size], x_diff[size:]

#test for practical significance

def cohen_d(x,y):
    nx = len(x)
    ny = len(y)
    dof = nx + ny - 2
    return (mean(x) - mean(y)) / sqrt(((nx-1)*std(x, ddof=1) ** 2 + (ny-1)*std
(y, ddof=1) ** 2) / dof)

print(cohen_d(test,train))

#
0.0315758778594
```

Model Identification

```
In [68]: #finding optimal parameters for ARIMA model

"""Split the dataset into training and test sets.
Walk the time steps in the test dataset.
Train an ARIMA model.
Make a one-step prediction.
Store prediction; get and store actual observation.
Calculate error score for predictions compared to expected values."""

#Source: https://machinelearningmastery.com/grid-search-arima-hyperparameters-with-python/

# evaluate an ARIMA model for a given order (p,d,q)
def evaluate_arima_model(X, arima_order):
    size = len(X)//2
    # prepare training dataset
    train, test= X[:size], X[size:]
    history = [x for x in train]
    # make predictions
    predictions = list()
    for t in range(len(test)):
        model = ARIMA(history, order=arima_order)
        model_fit = model.fit(disp=0)
        yhat = model_fit.forecast()[0]
        predictions.append(yhat)
        history.append(test[t])
    # calculate out of sample error
    error = mean_squared_error(test, predictions)
    return error

def evaluate_models(dataset, p_values, d_values, q_values):
    dataset = dataset.astype('float32')
    best_score, best_cfg = float("inf"), None
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p,d,q)
                try:
                    mse = evaluate_arima_model(dataset, order)
                    if mse < best_score:
                        best_score, best_cfg = mse, order
                    print('ARIMA%s MSE=%.3f' % (order,mse))
                except:
                    continue
    print('Best ARIMA%s MSE=%.3f' % (best_cfg, best_score))

p_values = [0, 1, 2, 4, 6, 8, 10]
d_values = range(0,3)
q_values = range(0, 3)

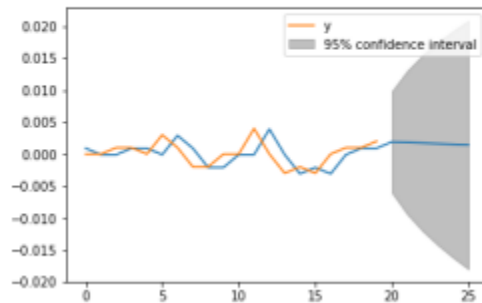
evaluate_models(series.values, p_values, d_values, q_values)
```

```
In [90]: model = ARIMA(test, order=(0,1,0))
         model_fit = model.fit(dispatch=False)
```

```
In [93]: forecast, stderr, conf = model_fit.forecast() #The forecast() function allows
         the confidence interval to be specified.
         #The alpha argument on the forecast() function specifies the confidence level.
         It is set by default to alpha=0.05, which is a 95% confidence interval. This
         is a sensible and widely used confidence interval.
         #An alpha of 0.05 means that the ARIMA model will estimate the upper and lower
         values around the forecast where there is a only 5% of the time the real valu
         e will not be in that range.
         print('Expected: %.3f' % test[0])
         print('Forecast: %.3f' % forecast)
         print('Standard Error: %.3f' % stderr)
         intervals = [0.2, 0.1, 0.05, 0.01]
         for a in intervals:
             forecast, stderr, conf = model_fit.forecast(alpha=a)
             print('%.1f%% Confidence Interval: %.3f between %.3f and %.3f' % ((1-a)
             )*100, forecast, conf[0][0], conf[0][1]))
```

Expected: 0.005
Forecast: 0.002
Standard Error: 0.004
80.0% Confidence Interval: 0.002 between -0.003 and 0.007
90.0% Confidence Interval: 0.002 between -0.005 and 0.009
95.0% Confidence Interval: 0.002 between -0.006 and 0.010
99.0% Confidence Interval: 0.002 between -0.009 and 0.012

```
In [98]: model_fit.plot_predict(len(train)-20, len(train)+5)
         plt.show()
```

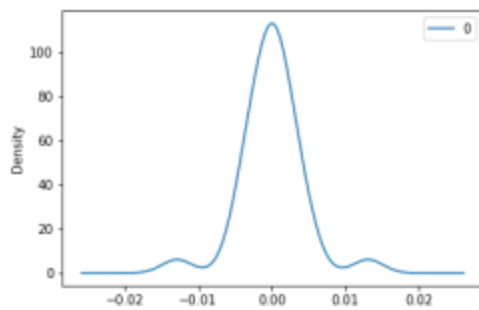
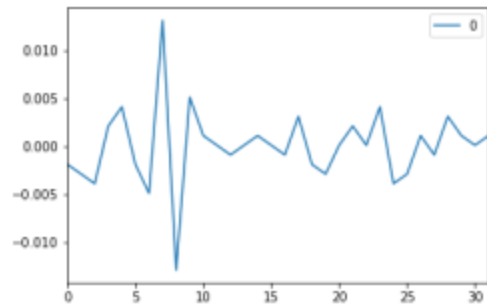


Model Diagnostics

```
In [95]: #After fitting the model, we should check whether the model is appropriate.
         #Let's analyze the residuals and investigate the autocorrelation of the ARIMA
         model
         residuals = pd.DataFrame(model_fit.resid)
```

```
In [100]: # plot residual erros
```

```
residuals.plot()  
residuals.plot(kind='kde')  
plt.show()  
print(residuals.describe())
```



```
count  3.200000e+01  
mean    8.321877e-11  
std     4.114171e-03  
min    -1.290624e-02  
25%    -1.906255e-03  
50%     9.374924e-05  
75%     1.343755e-03  
max     1.309375e-02
```