International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2025: 7.089

Time Series Modelling of Monthly Temperature Deviations Using ARIMA Techniques in Python

P. Ramakrishna Reddy[1], M. Naresh[2], S. Asif Alisha[3], P. Maheswari[4], A. Srinivasulu[5], Hema Sekhar Palla[6], B. Sarojamma[7], S. Venkatramana Reddy[8]

[1]Department of Statistics, S V University, Tirupati

[2]Department of Statistics, S V University, Tirupati

[3]Department of Statistics, Mohan Babu University, Tirupati

[4]Department of Statistics, Government Degree college, Nagari

[5]Department of Statistics, Vikrama Simhapuri University, Nellore

[6]Department of Statistics, Shri Gnanambica Degree College, Madanapalle Chittoor

[7]Department of Statistics, S V University, Tirupati

[8]Department of Physics, S. V. University, Tirupati
Corresponding Author Email: *drsvreddy123[at]gmail.com*

Abstract: ***Monthly temperature deviations from the 1951 to 1980 baseline were examined for the period 2019 to 2023 using time series modelling in a Python environment. Several ARIMA specifications, including ARIMA (1,0,0), ARIMA (0,1,1), and ARIMA (1,1,1), were estimated to understand patterns in atmospheric variability and to identify a model that captures temporal dependence effectively. Model adequacy was evaluated through Akaike and Bayesian information criteria, along with autocorrelation and partial autocorrelation diagnostics and Ljung–Box testing. Descriptive statistics, seasonal grouping, heat mapping, and rolling mean analysis supported the empirical assessment of monthly fluctuations. Comparative evaluation using root mean square error indicated that ARIMA (1,0,0) provided the most reliable representation of the observed temperature deviations. The findings contribute to statistical forecasting practices for climatic variables and demonstrate the applicability of computational tools in environmental data analysis.***

Keywords: temperature deviation, ARIMA modelling, time series forecasting, atmospheric variability, Python analysis

1. Introduction

Generally, temperature plays an important role for farmers and human life. By using Python software for atmospheric variables like temperature the models are listed. Stijn Heldens et.al [1], says that it is mainly required of researchers to investigate the scientific fields of their primary expertise. Steven D. Meyers et.al [2] explains the marine sector for several decades, the number and size of commercially operated marine boats on the world's oceans and inside seas have been increasing. Tom De Smedt et. al [3] discussed that the World Wide Web, often known as "Web as Corpus," is a vast collection of linguistic data that has gained popularity over the past several years as a useful tool for activities such as opinion mining, machine translation, and trend identification. Ceyhun Ozgur et.al [4] introduces the usefulness of MatLab, R and Python in teaching setting as compared in their work. In this case, they have attempted to determine which language is the most appropriate for instructing college students in statistics and OR. Noel M O'Boyle et.al [5] discussed, the Cheminformaticians to construct one-time scripts to do basic statistics and prepare data for analysis. For these routine jobs, the languages like Perl, Python, Ruby are perfect and unfortunately, they are orders of magnitude slower than complied languages like c++. William F. Holmgren et.al [6] introduces a well-known MATLAB package for solar modelling and analysis is called PVLIB Toolbox. Alan W Sousa da Silva et.al [7] explains, a wrapper script for the ANTECHAMBER tool, known as ACPYPE, makes it easier to generate small molecule topologies and parameters for a range of molecular dynamics systems. Nicholas K. Sauter et.al [8] says that the software development may be accelerated by using modular and reusable code. Skipper Seabold et.al [9] introduced a Python package for economic and statistical analysis is called Statsmodels. The target audience comprises of econometricians, theoretical and practical statisticians, Python developers from many fields utilise statistical models.

2. Methodology

For atmospheric variable like temperature from 2019 to 2023, we are fitted ARIMA (1,0,0) ARIMA (0,1,1) and ARIMA (1,1,1) models using Python software. ARIMA(p,d,q) is combinaton of ‘p’ order Auto Regression, Integration of order ‘d’ and moving average of order ‘q’. Akaike information criterion (AIC) and Baysian information Criterion (BIC) used for AIRMA model selection. AIC helps for the best relative fit. BIC finds true model among relative fit. Auto correlation function (ACF) explains about correlation between a time series and its lagged values for moving average part in ARIMA, where as partial auto correlation function (PACF) is correlation of lagged values and intermediate lags of autoregressive part of ARIMA.

Python Pogrom

```
1.import numpy as np
import pandas as pd
2.from statsmodels.tsa.stattools import adfuller

result = adfuller(ts)
print("ADF Statistic:", result[0])
print("p-value:", result[1])
3.from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(ts, order=(1,0,0))
model_fit = model.fit()

print(model_fit.summary())
4.import pandas as pd

df = pd.read_excel(r"C:\Users\DELL\Desktop\Atmospheric data by nasa 2018-2023.xlsx")
monthly_cols = ['Jan','Feb','Mar','Apr','May','Jun',
            'Jul','Aug','Sep','Oct','Nov','Dec']

df_monthly = df[['Year'] + monthly_cols]

ts_df = df_monthly.melt(
    id_vars='Year',
    var_name='Month',
    value_name='Value'
)
ts_df['Date'] = pd.to_datetime(
    ts_df['Year'].astype(str) + '-' + ts_df['Month'],
    format='%Y-%b'
)

ts_df = ts_df.sort_values('Date')
ts_df.set_index('Date', inplace=True)

ts = ts_df['Value']
10. import matplotlib.pyplot as plt

plt.figure(figsize=(10,4))
plt.plot(ts)
plt.title("Monthly Time Series (2019–2023)")
plt.show()
5.from statsmodels.tsa.arima.model import ARIMA
```

```
model = ARIMA(ts, order=(1,1,1))
model_fit = model.fit()

print(model_fit.summary())
Same procedure repeats for ARIMA(0,1,1), ARIMA(1,0,0)
6. forecast = model_fit.forecast(steps=12)
plt.figure(figsize=(10,4))
plt.plot(ts, label='Observed')
plt.plot(forecast, label='Forecast', color='red')
plt.legend()
plt.show()
7. import matplotlib.pyplot as plt

plt.figure(figsize=(12,5))
plt.plot(ts, marker='o')
plt.title("Monthly Atmospheric Time Series (2019–2023)")
plt.xlabel("Year")
plt.ylabel("Value")
plt.grid(True)
plt.show()
8. plt.figure(figsize=(12,5))
ts_df.boxplot(column='Value', by='Month_num')
plt.title("Monthly Variability")
plt.suptitle("")
plt.xlabel("Month")
plt.ylabel("Value")
plt.show()
9. rolling_mean = ts.rolling(window=12).mean()

plt.figure(figsize=(12,5))
plt.plot(ts, label="Original")
plt.plot(rolling_mean, label="12-Month Rolling Mean", linewidth=3)
plt.title("Trend Using Rolling Mean")
plt.legend()
plt.grid(True)
plt.show()
10. import numpy as np

def smape(actual, predicted):
    actual = np.array(actual)
    predicted = np.array(predicted)
    return 100 * np.mean(
        2 * np.abs(predicted - actual) /
        (np.abs(actual) + np.abs(predicted))
    )
```

```
    )
smape_value = smape(test.values, forecast.values)
print("sMAPE:", smape_value)
import matplotlib.pyplot as plt

plt.figure(figsize=(12,5))
plt.plot(test.index, test, label='Actual')
plt.plot(test.index, forecast, label='Forecast', linestyle='--')
plt.title("ARIMA Forecast vs Actual (sMAPE Evaluation)")
plt.xlabel("Date")
plt.ylabel("Value")
plt.legend()
plt.grid(True)
plt.show()
11. def smape(actual, predicted):
    return 100 * np.mean(
        2 * np.abs(predicted - actual) /
        (np.abs(actual) + np.abs(predicted))
    )

smape_value = smape(test.values, forecast.values)

print("RMSE :", rmse)
print("sMAPE:", smape_value)
import matplotlib.pyplot as plt

plt.figure(figsize=(12,5))

# Actual vs Forecast
plt.plot(test.index, test, label='Actual', marker='o')
plt.plot(test.index, forecast, label='Forecast', linestyle='--', marker='x')

# Title with metrics
plt.title(f"ARIMA Forecast Evaluation\nRMSE = {rmse:.3f}, sMAPE = {smape_value:.2f}%")

plt.xlabel("Date")
plt.ylabel("Value")
plt.legend()
plt.grid(True)
plt.show()
```

Empirical investigations:

Below is the temperature data deviations from the corresponding 1951-1980 means of 2019 to 2023 imported from excel to Python for 12 months, J-D means January to December, D-N means December to Novenber , DJF means December, January and Februrary, MAM means march, April, May, JJA means June July August and SON means September October and November. This data was taken from the website [10].

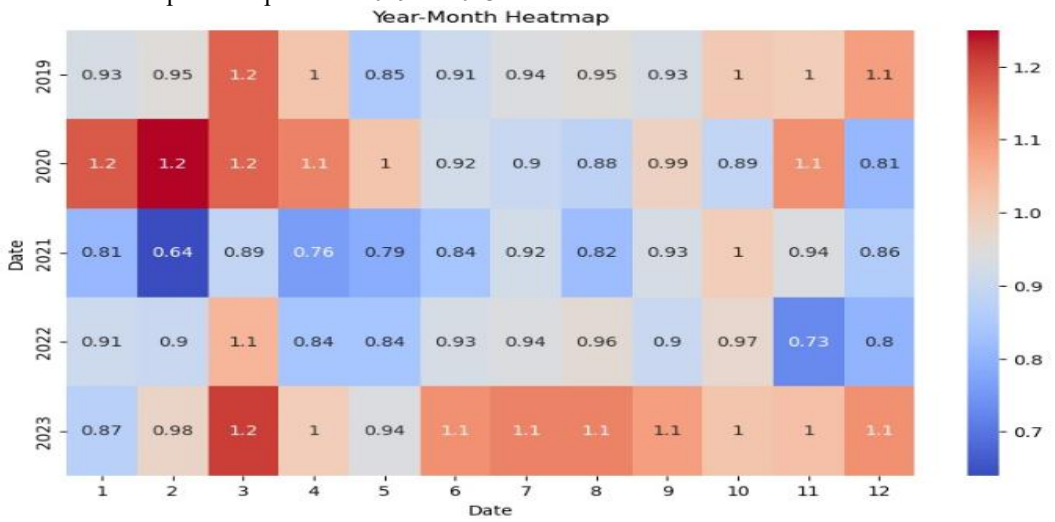
```
import numpy as np
import pandas as pd

data = pd.read_excel(r"C:\Users\DELL\Desktop\Atmospheric data by nasa 2018-2023.xlsx")
data.head()
```

Table 1

	Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	J-D	D-N	DJF	MAM	JJA	SON
0	2019	0.93	0.95	1.17	1.02	0.85	0.91	0.94	0.95	0.93	1.01	1.00	1.09	0.98	0.97	0.93	1.01	0.93	0.98
1	2020	1.18	1.25	1.17	1.13	1.02	0.92	0.90	0.88	0.99	0.89	1.11	0.81	1.02	1.04	1.17	1.11	0.90	0.99
2	2021	0.81	0.64	0.89	0.76	0.79	0.84	0.92	0.82	0.93	1.00	0.94	0.86	0.85	0.85	0.76	0.81	0.86	0.96
3	2022	0.91	0.90	1.05	0.84	0.84	0.93	0.94	0.96	0.90	0.97	0.73	0.80	0.90	0.90	0.89	0.91	0.94	0.87
4	2023	0.87	0.98	1.21	1.00	0.94	1.11	1.13	1.13	1.09	1.02	1.03	1.11	1.05	1.05	0.88	1.05	1.12	1.05

Year-month-wise heat map for temperature 2019 to 2023 is as shown in table - 2



Descriptive statistics for data is listed in table – 3

```
[5]: data.describe()
```

[5]:

	Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	J-D	D-N	DJF
count	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.00000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000
mean	2021.000000	0.940000	0.944000	1.098000	0.950000	0.888000	0.942000	0.966000	0.94800	0.968000	0.978000	0.962000	0.934000	0.960000	0.962000	0.926000
std	1.581139	0.141774	0.217555	0.130843	0.148324	0.091488	0.100349	0.093167	0.11649	0.075631	0.052631	0.143422	0.153395	0.083367	0.087006	0.150433
min	2019.000000	0.810000	0.640000	0.890000	0.760000	0.790000	0.840000	0.900000	0.82000	0.900000	0.890000	0.730000	0.800000	0.850000	0.850000	0.760000
25%	2020.000000	0.870000	0.900000	1.050000	0.840000	0.840000	0.910000	0.920000	0.88000	0.930000	0.970000	0.940000	0.810000	0.900000	0.900000	0.880000
50%	2021.000000	0.910000	0.950000	1.170000	1.000000	0.850000	0.920000	0.940000	0.95000	0.930000	1.000000	1.000000	0.860000	0.980000	0.970000	0.890000
75%	2022.000000	0.930000	0.980000	1.170000	1.020000	0.940000	0.930000	0.940000	0.96000	0.990000	1.010000	1.030000	1.090000	1.020000	1.040000	0.930000
max	2023.000000	1.180000	1.250000	1.210000	1.130000	1.020000	1.110000	1.130000	1.13000	1.090000	1.020000	1.110000	1.110000	1.050000	1.050000	1.170000

The Different ARIMA models of order (1,1,1), (1,0,0) and (0,1,1) coefficients, AIC, BIC and Ljung-Box test for lack of fit is given in table – 4.

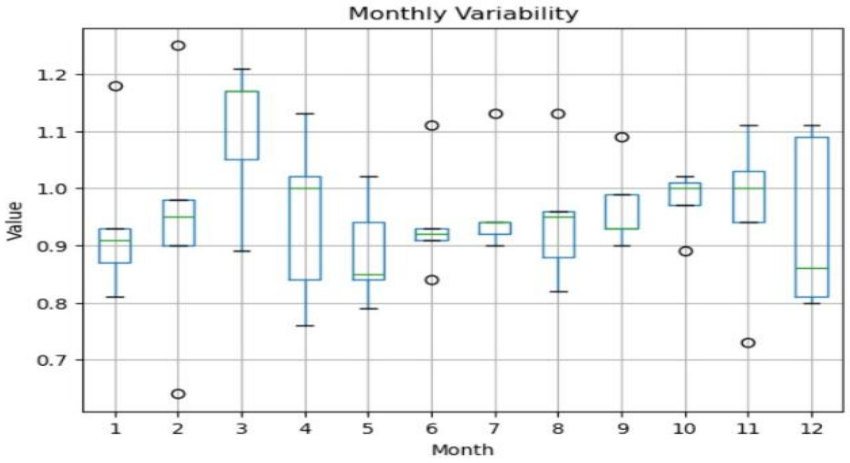
```
==============================================================================
Dep. Variable:                  Value   No. Observations:                   60
Model:                 ARIMA(1, 1, 1)   Log Likelihood                48.904
Date:                Mon, 05 Jan 2026   AIC                            -91.808
Time:                        09:25:48   BIC                            -85.575
Sample:                  01-01-2019   HQIC                           -89.375
                       - 12-01-2023
Covariance Type:                 opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.4549      0.188      2.418      0.016       0.086       0.824
ma.L1         -0.8566      0.137     -6.270      0.000      -1.124      -0.589
sigma2         0.0110      0.002      5.608      0.000       0.007       0.015
===================================================================================
Ljung-Box (L1) (Q):                   0.12   Jarque-Bera (JB):                 0.30
Prob(Q):                              0.73   Prob(JB):                         0.86
Heteroskedasticity (H):               1.17   Skew:                            -0.15
Prob(H) (two-sided):                  0.72   Kurtosis:                         3.19
===================================================================================
```

Dep. Variable:	Value	No. Observations:	60
Model:	ARIMA(0, 1, 1)	Log Likelihood	47.725
Date:	Mon, 05 Jan 2026	AIC	-91.449
Time:	14:12:38	BIC	-87.294
Sample:	01-01-2019 - 12-01-2023	HQIC	-89.827
Covariance Type:	opg		

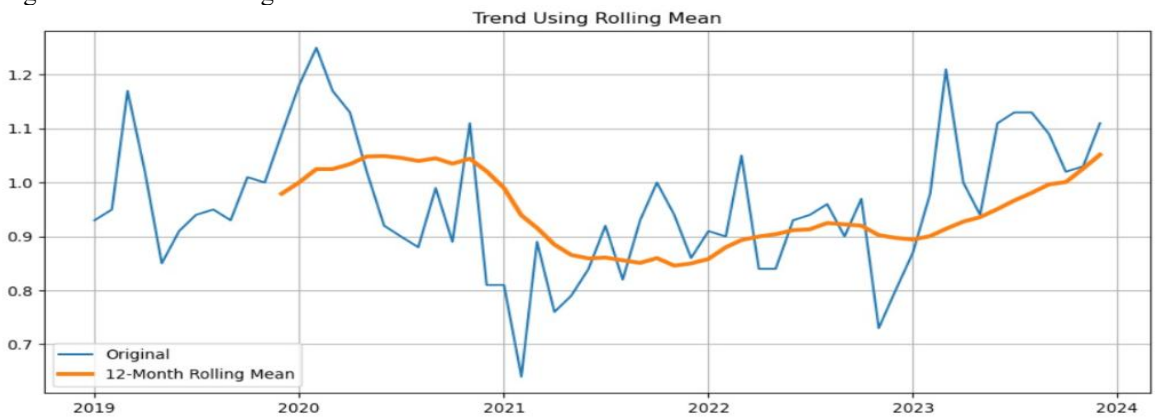
	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	-0.4634	0.115	-4.039	0.000	-0.688	-0.239
sigma2	0.0116	0.002	5.339	0.000	0.007	0.016

Ljung-Box (L1) (Q):	0.26	Jarque-Bera (JB):	0.05
Prob(Q):	0.61	Prob(JB):	0.98
Heteroskedasticity (H):	1.05	Skew:	0.07
Prob(H) (two-sided):	0.91	Kurtosis:	2.98

Monthly variability of temperature deviations from the corresponding 1951-1980 mean for months for January to December using Box-polt technique is given figer 1.



Figer 1, Figer 2 represents rolling mean of 12 months for data of 2019 to 2023 for temperature data, Blue line indicates original data and orange line indicates rolling mean model.



Root mean square error for ARIMA (1,1,1), ARIMA (1,0,0) and ARIMA (0,1,1) temperature data of 2019 to 2023 is listed in table - 5.

Model	RMSE value
ARIMA (1,1,1)	0.3115
ARIMA (1,0,0)	0.1544
ARIMA (0,1,1)	0.2550

3. Summary and Conclusion

For temperature data of monthly deviations from the corresponding 1951-1980 mean from 2019 to 2023 is fitted with different ARIMA models like ARIMA (1,1,1), ARIMA (1,0,0) and ARIMA (0,1,1). Model best fit is measured using AIC and BIC. Mean squared error criteria is used to check which model is the best for data. RMSE for different models are listed in table – 5.

By observing above table we conclude that for the temperature data, ARIMA (1,0,0) is the best model. (After observation of the RMSE values).

References

[1] Stijn Heldens, Alessio Sclocco, Henk Dreuning, Ben van Werkhoven, Pieter Hijma, Jason Maassen, Rob V. van Nieuwpoort, **"litstudy: A Python package for literature reviews"**, Published by Elsevier Ltd, 2352-7110/© 2022.

[2] Steven D. Meyers, Laura Azevedo, Mark E. Luther, **"A Scopus-based bibliometric study of maritime research involving the Automatic Identification System"**, Transportation Research Interdisciplinary Perspectives, Published by Elsevier Ltd, 2590-1982/© 2021.

[3] Tom De Smedt and Walter Daelemans, **"Pattern for Python"**, Journal of Machine Learning Research 13 (2012) 2063-2067.

[4] Ceyhun Ozgur, Taylor Colliau, Grace Rogers, Zachariah Hughes4, Elyse "Bennie" Myer-Tyson, **"MatLab vs. Python vs. R"**, Journal of data science: JDS · July 2017, 355-372.

[5] Noel M O'Boyle, Chris Morley and Geoffrey R Hutchison, **"Pybel: a Python wrapper for the Open Babel cheminformatics toolkit"**, Chemistry Central Journal 2008, 2:5.

[6] William F. Holmgren, Robert W. Andrews, Antonio T. Lorenzo, Joshua S. Stein, **"PVLIB Python 2015"**, SAND2015-6181C.

[7] Alan W Sousa da Silva and Wim F Vranken, **"ACPYPE - AnteChamber PYthon Parser interface"**, BMC Research Notes 2012, 5:367.

[8] Nicholas K. Sauter, Johan Hattne, Ralf W. Grosse Kunstleve and Nathaniel Echols, **"New Python-based methods for data processing"**, Acta Cryst. (2013). D69, 1274–1282, Acta Crystallographica, Section D, Biological Crystallography, ISSN 0907-4449.

[9] Skipper Seabold and Josef Perktold, **"Statsmodels: Econometric and Statistical Modeling with Python"**, PROC. OF THE 9th PYTHON IN SCIENCE CONF. (SCIPY 2010).

[10] https://www.kaggle.com/datasets/sujaykapadnis/global-surface-temperatures/data