A

For Figure 5 A and B

```
import pandas as pd
import numpy as np
import matplotlib
import sys
import os
import statsmodels.formula.api as smf
import statsmodels.tsa.stattools as ts
import statsmodels.tsa.ar_model as ar
import statsmodels.tsa.arima_model as arm
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
from statsmodels.stats.diagnostic import acorr_ljungbox, lilliefors

import matplotlib.pyplot as plt
import matplotlib.dates

%matplotlib inline
```

Data reading and stationary processing

```
# Read the file and save it to df
filepath = r'/home/jupyter-g1901210103@stu/data/ETF300.csv'
df = pd.read_csv(filepath)
df.head()
```

	CSI300 Open	CSI300 High	CSI300 Low	CSI300 Close	CSI300 Volume	CSI300 Turnover
Date						
2019	3.7	3.7	3.7	3.7	0.0	0.0
2019	3.7	3.7	3.7	3.7	0.0	0.0
2019	3.7	3.7	3.7	3.7	0.0	0.0
2019	3.7	3.7	3.7	3.7	0.0	0.0
2019	3.7	3.7	3.7	3.7	0.0	0.0

```
# Select the preprocessing
# Take the closing price of the Shen Bao Index df['CSI300_Close_Close'] df.drop(columns=['Unit'])
df.drop(columns=['Unit'])
df.head()
```

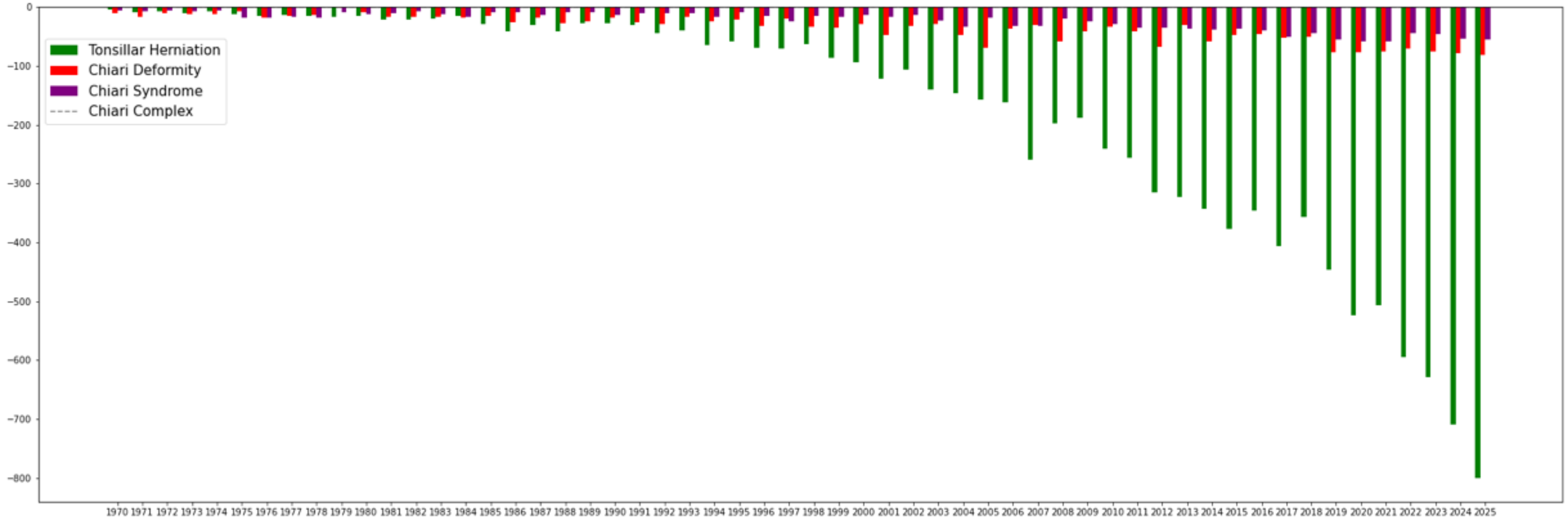
	CSI300 Open	CSI300 High	CSI300 Low	CSI300 Close	CSI300 Volume	CSI300 Turnover
Date						
2019	3.7	3.7	3.7	3.7	0.0	0.0
2019	3.7	3.7	3.7	3.7	0.0	0.0
2019	3.7	3.7	3.7	3.7	0.0	0.0
2019	3.7	3.7	3.7	3.7	0.0	0.0
2019	3.7	3.7	3.7	3.7	0.0	0.0

```
# Select the preprocessing
# Take the closing price of the Shen Bao Index df['CSI300_Close_Close'] df.sort_index()
df.sort_index()
df.head()
```

	CSI300 Open	CSI300 High	CSI300 Low	CSI300 Close	CSI300 Volume	CSI300 Turnover
Date						
2019	3.7	3.7	3.7	3.7	0.0	0.0
2019	3.7	3.7	3.7	3.7	0.0	0.0
2019	3.7	3.7	3.7	3.7	0.0	0.0
2019	3.7	3.7	3.7	3.7	0.0	0.0
2019	3.7	3.7	3.7	3.7	0.0	0.0

```
# Select the preprocessing
# Take the closing price of the Shen Bao Index df['CSI300_Close_Close'] df.sortindex()
df.sortindex()
df.head()
```

```
AttributeError                            Traceback (most recent call last)
```


Tonsillar Herniation
Chiari Deformity
Chiari Syndrome
Chiari Complex
0
−100
−200
−300
−400
−500
−600
−700
−800
1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025

In [ ]:

In [ ]:

B

For Figure 5 C and D

```
import pandas as pd
import numpy as np
import matplotlib
import os
import sys
import matplotlib.pyplot as plt
import statsmodels.tsa.ar_ar as ar
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error
from statsmodels.stattools import acorr_ljungbox # white noise test
from statsmodels.tsa.statespace import sarimax # Seasonality

import matplotlib.pyplot as plt
%matplotlib inline
```

```
# Input the data from our file
file_data = pd.read_csv('Tripod-prediction.xlsx')[[mo-4,]-set_index('Year')
file_data
```

	Chart 1 Mathematics	Chart 1 Mathematics	Chart 1.5 Mathematics	Chart 1 Mathematics	Chart 2 Mathematics	Chart 2 Mathematics
Term						
1991-1992	2.0	29.0	0.0	44.0	63.0	29.0
1990	0.0	72	0.0	0.0	1.0	0.0
1992	1.0	72	0.0	0.0	0.0	0.0
1993	0.0	72	0.0	1.0	0.0	1.0
1994	0.0	16.0	0.0	0.0	1.0	1.0
...	...	...	...	...	...	...
2018	21.0	1543.0	71.0	326.0	43.0	28.0
2019	25.0	1722.0	35.0	334.0	42.0	16.0
2020	47.0	1776.0	25.0	316.0	33.0	20.0
2021	38.0	1816.0	25.0	346.0	45.0	24.0
2022	40.0	1844.0	16.0	381.0	35.0	24.0

23 rows x 6 columns

```
# Attach the predictability
name1 = np.diff(file_data.loc[1970:, 'Chart 2 Mathematics'])
print("The p-value of this Best Best Test is",adfuller(name1,autolag='AIC')[1] )
```

```
The p-value of this Best Best Test is 2.0528231037425634e-
```

The data is stationary.

- The p-value is very closed to 0, so we can reject H0(There exists unit root).

Autoregressive Moving Average Models - ARMA(p, q)

```
# Find the best order of the log series of ar1
import warnings
warnings.filterwarnings("ignore")

import statsmodels.tsa.statespace as st

order_analyzer = st.arma_order_select_ic(name1, max_ar=7, max_ma=7, ic='aic')
print("The order of the model with smallest aic value is",order_analyzer.aic_min_order)
```

```
The order of the model with smallest aic value is (6, 6)
```

```
# Fit the data using ARMA(6,6)
import warnings
warnings.filterwarnings("ignore")

from statsmodels.tsa.arima_model import ARIMA
arma_mod = ARIMA(name1,order=(6,0,6))
parameters=arma_mod.fit()
print(parameters.aic)

# Attach whether residuals are white noise or not
pvalue_lb_test = acorr_ljungbox(arma_mod.resid,lags=[1], boxpierce=False,return_df=False)[1][0]

if pvalue_lb_test > 0.05:
    print('Passed the Lb equivilence level, we have reason to suspect the residuals are not white noise.')
else:
    print('The residuals exam like white noise.')
```

```
The residuals exam like white noise.
```

```
# Raise the forecast parts of ARMA model
arma_forecasts,arm95,ci95 = arma_mod.forecast(steps=5)
print("The forecast of one-step-ahead demand is: ",arma_forecasts)
```

```
The forecast of one-step-ahead demand is: [-0.10999961  0.00391025  0.14102844]
```

```
plt.figure(figsize=(20,15))
plt.plot(arma_mod.fittedvalues,arma_forecasts,label='forecast')
plt.plot(name1,label='original')
plt.legend(fontsize=15)
plt.show()
```

```
log_name1_1 = list(np.append(name1,arma_forecasts))
for i in range(len(name1),len(name1)+len(arma_forecasts),1):
# print(list_1 + pfile_data.loc[1970, 'Chart Mathematics'])
    exec('list_%s = []'%(i))
    exec('for j in range(len(log_name1_1)):\n\tprint(list_%s[j])'%(i,i,i))
    exec('list_%s.append(log_name1_1[j])'%(i,i))
    exec('list_%s = pd.DataFrame(list_%s,columns=[\'number\'])'%(i,i))
    exec('list_%s.to_csv(\'%s.csv\')'%(i,i))
    exec('list_%s.head()'%(i))
```

	number	year
0	19.0	2020
1	19.0	2021
2	19.0	2022
3	19.0	2023
4	19.0	2024

```
# Attach the predictability
name2 = np.diff(file_data.loc[1970:, 'Chart 1.5 Mathematics ')]
print("The p-value of this Best Best Test is",adfuller(name2,autolag='AIC')[1] )
```

```
The p-value of this Best Best Test is 0.00042478350972329
```

```
import warnings
warnings.filterwarnings("ignore")

import statsmodels.tsa.statespace as st

order_analyzer = st.arma_order_select_ic(name2, max_ar=7, max_ma=7, ic='aic')
print("The order of the model with smallest aic value is",order_analyzer.aic_min_order)
```

```
The order of the model with smallest aic value is (2, 2)
```

```
# Fit the data using ARMA(2,2)
import warnings
warnings.filterwarnings("ignore")

from statsmodels.tsa.arima_model import ARIMA
arma_mod = ARIMA(name2,order=(2,0,2))
parameters=arma_mod.fit()
print(parameters.aic)

# Attach whether residuals are white noise or not
pvalue_lb_test = acorr_ljungbox(arma_mod.resid,lags=[1], boxpierce=False,return_df=False)[1][0]

if pvalue_lb_test > 0.05:
    print('Passed the Lb equivilence level, we have reason to suspect the residuals are not white noise.')
else:
    print('The residuals exam like white noise.')

# Raise the forecast parts of ARMA model
arma_forecasts,arm95,ci95 = arma_mod.forecast(steps=5)
print("The forecast of one-step-ahead demand is: ",arma_forecasts)
```

```
The residuals exam like white noise.
The forecast of one-step-ahead demand is: [-0.07452716 -0.75470409 -0.23496366]
```

```
log_name2_1 = list(np.append(name2,arma_forecasts))
for i in range(len(name2),len(name2)+len(arma_forecasts),1):
# print(list_1 + pfile_data.loc[1970, 'Chart Mathematics '])
    exec('list_%s = []'%(i))
    exec('for j in range(len(log_name2_1)):\n\tprint(list_%s[j])'%(i,i,i))
    exec('list_%s.append(log_name2_1[j])'%(i,i))
    exec('list_%s = pd.DataFrame(list_%s,columns=[\'number\'])'%(i,i))
    exec('list_%s.to_csv(\'%s.csv\')'%(i,i))
    exec('list_%s.head()'%(i))
```

	number	year
0	19.0	2020
1	19.0	2021
2	19.0	2022
3	19.0	2023
4	19.0	2024

```
# Attach the predictability
name3 = np.diff(file_data.loc[1970:, 'Chart 1 Mathematics ')]
print("The p-value of this Best Best Test is",adfuller(name3,autolag='AIC')[1] )
```

```
The p-value of this Best Best Test is 2.8853033197304e-05
```

```
import warnings
warnings.filterwarnings("ignore")

import statsmodels.tsa.statespace as st

order_analyzer = st.arma_order_select_ic(name3, max_ar=7, max_ma=7, ic='aic')
print("The order of the model with smallest aic value is",order_analyzer.aic_min_order)
```

```
The order of the model with smallest aic value is (2, 2)
```

```
# Fit the data using ARMA(2,2)
import warnings
warnings.filterwarnings("ignore")

from statsmodels.tsa.arima_model import ARIMA
arma_mod = ARIMA(name3,order=(2,0,2))
parameters=arma_mod.fit()
print(parameters.aic)

# Attach whether residuals are white noise or not
pvalue_lb_test = acorr_ljungbox(arma_mod.resid,lags=[1], boxpierce=False,return_df=False)[1][0]

if pvalue_lb_test > 0.05:
    print('Passed the Lb equivilence level, we have reason to suspect the residuals are not white noise.')
else:
    print('The residuals exam like white noise.')

# Raise the forecast parts of ARMA model
arma_forecasts,arm95,ci95 = arma_mod.forecast(steps=5)
print("The forecast of one-step-ahead demand is: ",arma_forecasts)
```

```
The residuals exam like white noise.
The forecast of one-step-ahead demand is: [ 0.07452716 -1.75470409 -2.23496366]
```

```
log_name3_1 = list(np.append(name3,arma_forecasts))
for i in range(len(name3),len(name3)+len(arma_forecasts),1):
# print(list_1 + pfile_data.loc[1970, 'Chart 1 Mathematics '])
    exec('list_%s = []'%(i))
    exec('for j in range(len(log_name3_1)):\n\tprint(list_%s[j])'%(i,i,i))
    exec('list_%s.append(log_name3_1[j])'%(i,i))
    exec('list_%s = pd.DataFrame(list_%s,columns=[\'number\'])'%(i,i))
    exec('list_%s.to_csv(\'%s.csv\')'%(i,i))
    exec('list_%s.head()'%(i))
```

	number	year
0	19.0	2020
1	19.0	2021
2	19.0	2022
3	19.0	2023
4	19.0	2024

```
# Attach the predictability
name4 = np.diff(file_data.loc[1970:, 'Chart 2 Mathematics ')]
print("The p-value of this Best Best Test is",adfuller(name4,autolag='AIC')[1] )
```

```
The p-value of this Best Best Test is 7.3992001027904e-05
```

```
import warnings
warnings.filterwarnings("ignore")

import statsmodels.tsa.statespace as st

order_analyzer = st.arma_order_select_ic(name4, max_ar=7, max_ma=7, ic='aic')
print("The order of the model with smallest aic value is",order_analyzer.aic_min_order)
```

```
The order of the model with smallest aic value is (2, 2)
```

```
# Fit the data using ARMA(2,2)
import warnings
warnings.filterwarnings("ignore")

from statsmodels.tsa.arima_model import ARIMA
arma_mod = ARIMA(name4,order=(2,0,2))
parameters=arma_mod.fit()
print(parameters.aic)

# Attach whether residuals are white noise or not
pvalue_lb_test = acorr_ljungbox(arma_mod.resid,lags=[1], boxpierce=False,return_df=False)[1][0]

if pvalue_lb_test > 0.05:
    print('Passed the Lb equivilence level, we have reason to suspect the residuals are not white noise.')
else:
    print('The residuals exam like white noise.')

# Raise the forecast parts of ARMA model
arma_forecasts,arm95,ci95 = arma_mod.forecast(steps=5)
print("The forecast of one-step-ahead demand is: ",arma_forecasts)
```

```
The residuals exam like white noise.
The forecast of one-step-ahead demand is: [-0.07452716 -0.38617425  0.40797896]
```

```
log_name4_1 = list(np.append(name4,arma_forecasts))
for i in range(len(name4),len(name4)+len(arma_forecasts),1):
# print(list_1 + pfile_data.loc[1970, 'Chart 2 Mathematics '])
    exec('list_%s = []'%(i))
    exec('for j in range(len(log_name4_1)):\n\tprint(list_%s[j])'%(i,i,i))
    exec('list_%s.append(log_name4_1[j])'%(i,i))
    exec('list_%s = pd.DataFrame(list_%s,columns=[\'number\'])'%(i,i))
    exec('list_%s.to_csv(\'%s.csv\')'%(i,i))
    exec('list_%s.head()'%(i))
```

	number	year
0	19.0	2020
1	19.0	2021
2	19.0	2022
3	19.0	2023
4	19.0	2024

```
# Attach the predictability
name5 = np.diff(file_data.loc[1970:, 'Chart 1 Mathematics ')]
print("The p-value of this Best Best Test is",adfuller(name5,autolag='AIC')[1] )
```

```
The p-value of this Best Best Test is 2.3592001027904e-05
```

```
import warnings
warnings.filterwarnings("ignore")

import statsmodels.tsa.statespace as st

order_analyzer = st.arma_order_select_ic(name5, max_ar=7, max_ma=7, ic='aic')
print("The order of the model with smallest aic value is",order_analyzer.aic_min_order)
```

```
The order of the model with smallest aic value is (2, 2)
```

```
# Fit the data using ARMA(2,2)
import warnings
warnings.filterwarnings("ignore")

from statsmodels.tsa.arima_model import ARIMA
arma_mod = ARIMA(name5,order=(2,0,2))
parameters=arma_mod.fit()
print(parameters.aic)

# Attach whether residuals are white noise or not
pvalue_lb_test = acorr_ljungbox(arma_mod.resid,lags=[1], boxpierce=False,return_df=False)[1][0]

if pvalue_lb_test > 0.05:
    print('Passed the Lb equivilence level, we have reason to suspect the residuals are not white noise.')
else:
    print('The residuals exam like white noise.')

# Raise the forecast parts of ARMA model
arma_forecasts,arm95,ci95 = arma_mod.forecast(steps=5)
print("The forecast of one-step-ahead demand is: ",arma_forecasts)
```

```
The residuals exam like white noise.
The forecast of one-step-ahead demand is: [-0.07452716 -0.38617425  0.40797896]
```

```
log_name5_1 = list(np.append(name5,arma_forecasts))
for i in range(len(name5),len(name5)+len(arma_forecasts),1):
# print(list_1 + pfile_data.loc[1970, 'Chart 1 Mathematics '])
    exec('list_%s = []'%(i))
    exec('for j in range(len(log_name5_1)):\n\tprint(list_%s[j])'%(i,i,i))
    exec('list_%s.append(log_name5_1[j])'%(i,i))
    exec('list_%s = pd.DataFrame(list_%s,columns=[\'number\'])'%(i,i))
    exec('list_%s.to_csv(\'%s.csv\')'%(i,i))
    exec('list_%s.head()'%(i))
```

	number	year
0	12.0	2020
1	12.0	2021
2	12.0	2022
3	12.0	2023
4	12.0	2024

```
import warnings
warnings.filterwarnings("ignore")
import statsmodels.tsa.statespace as st
order_analyzer = st.arma_order_select_ic(name6, max_ar=7, max_ma=7, ic='aic')
print("The order of the model with smallest aic value is",order_analyzer.aic_min_order)
```

```
The order of the model with smallest aic value is (2, 2)
```

```
import warnings
warnings.filterwarnings("ignore")

from statsmodels.tsa.arima_model import ARIMA
arma_mod = ARIMA(name6,order=(2,0,2))
parameters=arma_mod.fit()
print(parameters.aic)

# Attach whether residuals are white noise or not
pvalue_lb_test = acorr_ljungbox(arma_mod.resid,lags=[1], boxpierce=False,return_df=False)[1][0]

if pvalue_lb_test > 0.05:
    print('Passed the Lb equivilence level, we have reason to suspect the residuals are not white noise.')
else:
    print('The residuals exam like white noise.')

# Raise the forecast parts of ARMA model
arma_forecasts,arm95,ci95 = arma_mod.forecast(steps=5)
print("The forecast of one-step-ahead demand is: ",arma_forecasts)
```

```
The residuals exam like white noise.
The forecast of one-step-ahead demand is: [-0.07452716 -0.38617425  0.40797896]
```

```
log_name6_1 = list(np.append(name6,arma_forecasts))
for i in range(len(name6),len(name6)+len(arma_forecasts),1):
# print(list_1 + pfile_data.loc[1970, 'Chart 1 Mathematics '])
    exec('list_%s = []'%(i))
    exec('for j in range(len(log_name6_1)):\n\tprint(list_%s[j])'%(i,i,i))
    exec('list_%s.append(log_name6_1[j])'%(i,i))
    exec('list_%s = pd.DataFrame(list_%s,columns=[\'number\'])'%(i,i))
    exec('list_%s.to_csv(\'%s.csv\')'%(i,i))
    exec('list_%s.head()'%(i))
```

	number	year
0	12.0	2020
1	12.0	2021
2	12.0	2022
3	12.0	2023
4	12.0	2024

```
# Attach the predictability
name7 = np.diff(file_data.loc[1970:, 'Chart 1 Mathematics ')]
print("The p-value of this Best Best Test is",adfuller(name7,autolag='AIC')[1] )
```

```
The p-value of this Best Best Test is 2.2592001027904e-05
```

```
import warnings
warnings.filterwarnings("ignore")
import statsmodels.tsa.statespace as st
order_analyzer = st.arma_order_select_ic(name7, max_ar=7, max_ma=7, ic='aic')
print("The order of the model with smallest aic value is",order_analyzer.aic_min_order)
```

```
The order of the model with smallest aic value is (2, 2)
```

```
import warnings
warnings.filterwarnings("ignore")

from statsmodels.tsa.arima_model import ARIMA
arma_mod = ARIMA(name7,order=(2,0,2))
parameters=arma_mod.fit()
print(parameters.aic)

# Attach whether residuals are white noise or not
pvalue_lb_test = acorr_ljungbox(arma_mod.resid,lags=[1], boxpierce=False,return_df=False)[1][0]

if pvalue_lb_test > 0.05:
    print('Passed the Lb equivilence level, we have reason to suspect the residuals are not white noise.')
else:
    print('The residuals exam like white noise.')

# Raise the forecast parts of ARMA model
arma_forecasts,arm95,ci95 = arma_mod.forecast(steps=5)
print("The forecast of one-step-ahead demand is: ",arma_forecasts)
```

```
The residuals exam like white noise.
The forecast of one-step-ahead demand is: [-0.07452716 -0.38617425  0.40797896]
```

```
log_name7_1 = list(np.append(name7,arma_forecasts))
for i in range(len(name7),len(name7)+len(arma_forecasts),1):
# print(list_1 + pfile_data.loc[1970, 'Chart 1 Mathematics '])
    exec('list_%s = []'%(i))
    exec('for j in range(len(log_name7_1)):\n\tprint(list_%s[j])'%(i,i,i))
    exec('list_%s.append(log_name7_1[j])'%(i,i))
    exec('list_%s = pd.DataFrame(list_%s,columns=[\'number\'])'%(i,i))
    exec('list_%s.to_csv(\'%s.csv\')'%(i,i))
    exec('list_%s.head()'%(i))
```

	number	year
0	12.0	2020
1	12.0	2021
2	12.0	2022
3	12.0	2023
4	12.0	2024

```
# Attach the predictability
name8 = np.diff(file_data.loc[1970:, 'Chart 2 Mathematics ')]
print("The p-value of this Best Best Test is",adfuller(name8,autolag='AIC')[1] )
```

```
The p-value of this Best Best Test is 2.2592001027904e-05
```

```
import warnings
warnings.filterwarnings("ignore")
import statsmodels.tsa.statespace as st
order_analyzer = st.arma_order_select_ic(name8, max_ar=7, max_ma=7, ic='aic')
print("The order of the model with smallest aic value is",order_analyzer.aic_min_order)
```

```
The order of the model with smallest aic value is (2, 2)
```

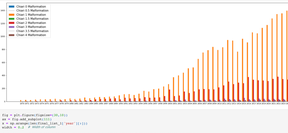
```
import warnings
warnings.filterwarnings("ignore")

from statsmodels.tsa.arima_model import ARIMA
arma_mod = ARIMA(name8,order=(2,0,2))
parameters=arma_mod.fit()
print(parameters.aic)

# Attach whether residuals are white noise or not
pvalue_lb_test = acorr_ljungbox(arma_mod.resid,lags=[1], boxpierce=False,return_df=False)[1][0]

if pvalue_lb_test > 0.05:
    print('Passed the Lb equivilence level, we have reason to suspect the residuals are not white noise.')
else:
    print('The residuals exam like white noise.')

# Raise the forecast parts of ARMA model
arma_forecasts,arm95,ci95 = arma_mod.forecast(steps=5)
print("The forecast of one-step-ahead demand is: ",arma_forecasts)
```

```
The residuals exam like white noise.
The forecast of one-step-ahead demand is: [-0.07452716 -0.38617425  0.40797896]
```

```
log_name8_1 = list(np.append(name8,arma_forecasts))
for i in range(len(name8),len(name8)+len(arma_forecasts),1):
# print(list_1 + pfile_data.loc[1970, 'Chart 2 Mathematics '])
    exec('list_%s = []'%(i))
    exec('for j in range(len(log_name8_1)):\n\tprint(list_%s[j])'%(i,i,i))
    exec('list_%s.append(log_name8_1[j])'%(i,i))
    exec('list_%s = pd.DataFrame(list_%s,columns=[\'number\'])'%(i,i))
    exec('list_%s.to_csv(\'%s.csv\')'%(i,i))
    exec('list_%s.head()'%(i))
```

	number	year
0	12.0	2020
1	12.0	2021
2	12.0	2022
3	12.0	2023
4	12.0	2024

```
fig = plt.figure(figsize=(20,15))
ax = fig.add_subplot(111)
x = np.arange(name1[name1.]).T
width = 0.2 # 设置柱子的宽度

ax.bar(x + width/2, [x for x in range(11)], width, label='Chart 1 Mathematics ')
ax.bar(x + width/2, [x for x in range(11)], width, label='Chart 1.5 Mathematics ',color='yellow')
ax.bar(x + width/2, [x for x in range(11)], width, label='Chart 1 Mathematics ')
ax.bar(x + width/2, [x for x in range(11)], width, label='Chart 2 Mathematics ')
ax.bar(x + width/2, [x for x in range(11)], width, label='Chart 1 Mathematics ')
ax.bar(x + width/2, [x for x in range(11)], width, label='Chart 2 Mathematics ')
ax.bar(x + width/2, [x for x in range(11)], width, label='Chart 1 Mathematics ',color='lightgreen')
ax.bar(x + width/2, [x for x in range(11)], width, label='Chart 2 Mathematics ',color='pink')
# ax.plot(x+ width/2, [x for x in range(11)], 'label=\'Chart 1.5 Mathematics ',color='pink',linewidth=2)

ax.set_xticks(x, labels=[list_1]*year'])

handles,labels = ax.get_legend_handles_labels()
handles [] = handles[0],handles[1],handles[2],handles[3],handles[4],handles[5],handles[6]
labels [] = labels[0],labels[1],labels[2],labels[3],labels[4],labels[5],labels[6]
ax.legend(handles,labels,loc='upper left', fontsize=8)

# ax.set_xticks(x, labels=[x])

# ax.legend()
plt.show()
```




```
fig = plt.figure(figsize=(20,15))
ax = fig.add_subplot(111)
x = np.arange(name1[name1.]).T
width = 0.2 # 设置柱子的宽度

ax.bar(x + width, [x for x in range(11)], width, label='Chart 1 Mathematics ',color='#1f77b4')
ax.bar(x + width, [x for x in range(11)], width, label='Chart 1.5 Mathematics ',color='green')
# ax.plot(x+ width/2, [x for x in range(11)], 'label=\'Chart 1.5 Mathematics ',color='pink',linewidth=2)

ax.set_xticks(x, labels=[list_1]*year'])

# ax.legend()
plt.show()
```

C

For Figure 5 E and F

In [1]:
```
import pandas as pd
import datetime
import numpy as np
import os
import xpt

import statsmodels.formula.api as smf
import statsmodels.tsa.api as smt
import statsmodels.api as sm
import scipy.stats as scs
import statsmodels.stats.diagnostic import acorr_ljungbox  # white noise test
from statsmodels.tsa.stattools import adfuller  #stationarity

import matplotlib.pyplot as plt
import matplotlib as mpl
```

In [7]:
```
#input the data from cev file
file_data = pd.read_excel("Figure3-prediction.xlsx").iloc[:-4,:].set_index('Term')
file_data
```

Out[7]:

Term	Acquired Chiari Malformation	Primary Chiari Malformation	Chiari-0Etiologies
1891-1950	0.0	5.0	0.0
1951	0.0	0.0	0.0
1952	0.0	0.0	0.0
1953	0.0	0.0	0.0
1954	0.0	0.0	0.0
...	...	...	...
2018	47.0	4.0	0.0
2019	41.0	6.0	2.0
2020	51.0	19.0	1.0
2021	57.0	14.0	0.0
2022	56.0	14.0	0.0

73 rows × 3 columns

In [8]:
```
#check the stationarity
term1 = np.diff(file_data.loc[1981:,'Acquired Chiari Malformation'])
# term1 = file_data.loc[1981:,'Secondary Chiari Malformation ']
print("The p-value of Unit Root Test is:",adfuller(term1,autolag='AIC')[1] )
```

```
The p-value of Unit Root Test is: 3.3421806174222e-18
```

The data is stationary:

- The p-value is very closed to 0, so we can reject H0(There exists unit root)

Autoregressive Moving Average Models - ARMA(p, q)

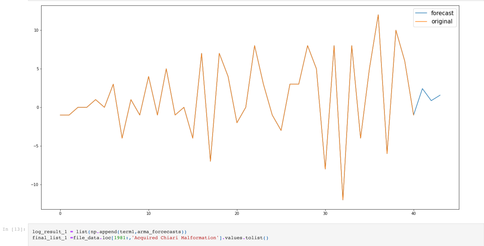
In [9]:
```
# Find the best order of the log return of P1

import warnings
warnings.filterwarnings('ignore')

import statsmodels.tsa.stattools as st


order_analyse = st.arma_order_select_ic(term1, max_ar=5, max_ma=5, ic=['aic'])
print("The order of the model with smallest aic value is",order_analyse.aic_min_order)
```

```
The order of the model with smallest aic value is: (1, 0)
```

In [10]:
```
import warnings
warnings.filterwarnings('ignore')

from statsmodels.tsa.arima_model import ARIMA
arma_mdl = ARIMA(term1,order=(1,0,1).fit()
params=arma_mdl.params

#check whether residuals are white noise or not
pvalue_lb_test = acorr_ljungbox(arma_mdl.resid,lags=[10], boxpierce=False,return_df=False)[1][0]

if pvalue_lb_test < 0.05:
    print('Passed on 5% significance level, we have reason to suspect the residuals are not white noise.')
else:
    print('The residuals seem like white noise.')

#show the forecast parts of ARMA model
arma_forecasts,err95,ci95 = arma_mdl.forecast(steps=3)
print('The forecast of one-step-ahead momentum is: ',arma_forecasts)
```

```
The residuals seem like white noise.
The forecast of one-step-ahead momentum is:  [2.42476853 0.86486605 1.57536004]
```

In [11]:
```
plt.figure(figsize=(20,10))
plt.plot(np.append(term1,arma_forecasts),label='forecast')
plt.plot(term1,label='original')
plt.legend(fontsize=15)
plt.show()
```



In [12]:
```
log_result_1 = list(np.append(term1,arma_forecasts))
final_list_1 =file_data.loc[1981:,'Acquired Chiari Malformation'].values.tolist()

for i in range(3):
    final_list_1.append(int(log_result_1[i-3]+final_list_1[i-1]))

final_list_1 = pd.DataFrame(final_list_1,columns=['number'])
final_list_1['year']= np.arange(1981,2026)
final_list_1.tail()
```

Out[12]:

	number	year
40	57.0	2021
41	56.0	2022
42	58.0	2023
43	56.0	2024
44	59.0	2025

In [13]:
```
#check the stationarity
term2 = np.diff(file_data.loc[1981:,'Primary Chiari Malformation'])
print("The p-value of Unit Root Test is:",adfuller(term2,autolag='AIC')[1] )
```

```
The p-value of Unit Root Test is: 0.003989422951318905
```

In [20]:
```
import warnings
warnings.filterwarnings('ignore')

import statsmodels.tsa.stattools as st


order_analyse = st.arma_order_select_ic(term2, max_ar=5, max_ma=5, ic=['bic'])
print("The order of the model with smallest bic value is",order_analyse.bic_min_order)
```

```
The order of the model with smallest aic value is: (0, 1)
```

In [21]:
```
import warnings
warnings.filterwarnings('ignore')

from statsmodels.tsa.arima_model import ARIMA
arma_mdl = ARIMA(term2,order=(0,0,1).fit()
params=arma_mdl.params

#check whether residuals are white noise or not
pvalue_lb_test = acorr_ljungbox(arma_mdl.resid,lags=[10], boxpierce=False,return_df=False)[1][0]

if pvalue_lb_test < 0.05:
    print('Passed on 5% significance level, we have reason to suspect the residuals are not white noise.')
else:
    print('The residuals seem like white noise.')

#show the forecast parts of ARMA model
arma_forecasts,err95,ci95 = arma_mdl.forecast(steps=3)
print('The forecast of one-step-ahead momentum is: ',arma_forecasts)
```

```
Passed on 5% significance level, we have reason to suspect the residuals are not white noise.
The forecast of one-step-ahead momentum is:  [-1.59475841  0.31137112  0.31137112]
```

In [22]:
```
log_result_2 = list(np.append(term2,arma_forecasts))
final_list_2 =file_data.loc[1981:,'Primary Chiari Malformation'].values.tolist()

for i in range(3):
    final_list_2.append(int(log_result_2[i-3]+final_list_2[i-1]))

final_list_2 = pd.DataFrame(final_list_2,columns=['number'])
final_list_2['year']= np.arange(1981,2026)
final_list_2.tail()
```

Out[22]:

	number	year
40	14.0	2021
41	14.0	2022
42	12.0	2023
43	12.0	2024
44	12.0	2025

In [24]:
```
#check the stationarity
term3 = np.diff(file_data.loc[1970:,'Chiari-0Etiologies '])
print("The p-value of Unit Root Test is",adfuller(term3,autolag='AIC')[1] )
```

```
The p-value of Unit Root Test is: 9.090230792764362372
```

In [25]:
```
import warnings
warnings.filterwarnings('ignore')

import statsmodels.tsa.stattools as st


order_analyse = st.arma_order_select_ic(term3, max_ar=5, max_ma=5, ic=['aic'])
print("The order of the model with smallest aic value is",order_analyse.aic_min_order)
```

```
The order of the model with smallest aic value is: (2, 0)
```

In [26]:
```
import warnings
warnings.filterwarnings('ignore')

from statsmodels.tsa.arima_model import ARIMA
arma_mdl = ARIMA(term3,order=(2,0,2).fit()
params=arma_mdl.params

#check whether residuals are white noise or not
pvalue_lb_test = acorr_ljungbox(arma_mdl.resid,lags=[10], boxpierce=False,return_df=False)[1][0]

if pvalue_lb_test < 0.05:
    print('Passed on 5% significance level, we have reason to suspect the residuals are not white noise.')
else:
    print('The residuals seem like white noise.')

#show the forecast parts of ARMA model
arma_forecasts,err95,ci95 = arma_mdl.forecast(steps=3)
print('The forecast of one-step-ahead momentum is: ',arma_forecasts)
```

```
The residuals seem like white noise.
The forecast of one-step-ahead momentum is:  [ 1.28548415  0.06465796 -0.71132457]
```

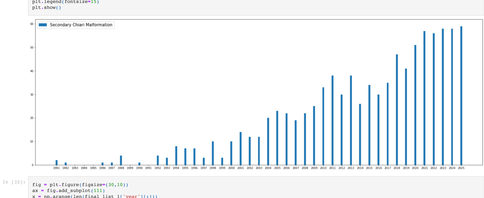
In [27]:
```
plt.figure(figsize=(20,10))
plt.plot(np.append(term3,arma_forecasts),label='forecast')
plt.plot(term3,label='original')
plt.legend(fontsize=15)
plt.show()
```

In [29]:
```
log_result_3 = list(np.append(term3,arma_forecasts))
final_list_3 =file_data.loc[1981:,'Chiari-0Etiologies '].values.tolist()

for i in range(3):
    final_list_3.append(int(log_result_3[i-3]+final_list_3[i-1]))

final_list_3 = pd.DataFrame(final_list_3,columns=['number'])
final_list_3['year']= np.arange(1981,2026)
final_list_3.tail()
```

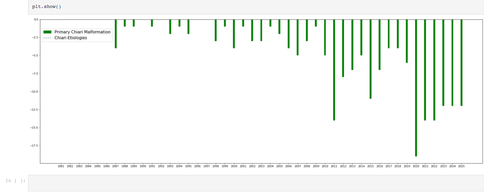
Out[29]:

	number	year
40	0.0	2021
41	0.0	2022
42	1.0	2023
43	1.0	2024
44	0.0	2025

In [30]:
```
plt.figure(figsize=(32,10))
x = np.arange(len(final_list_1['year']))
width = 0.2  # width of column

plt.bar(x, final_list_1['number'][:x], width, label='Secondary Chiari Malformation')
# plt.bar(x ,final_list_2['number'][:], width, label='Primary Chiari Malformation ')
# plt.bar(x + width, -final_list_3['number'][:], width, label='Chiari-like Malformation')
# plt.plot(x + width, [0 for i in range(44)],color='black')
# plt.plot(x + width, [5 for i in range(44)],color='black')

plt.xticks(x, labels=final_list_1['year'][:])
plt.legend(fontsize=15)
plt.show()
```



In [31]:
```
fig = plt.figure(figsize=(32,10))
ax = fig.add_subplot(111)
x = np.arange(len(final_list_3['year']))
width = 0.2  # width of column

ax.plot(x- width*0.5, [0 for i in range(45)], label='Chiari-0Etiologies ',color='grey',linestyle='--')
ax.bar(x, -final_list_2['number'][:], width, label='Primary Chiari Malformation ',color='green')


plt.xticks(x, labels=final_list_3['year'][:])

handles,labels = ax.get_legend_handles_labels()
handles = [ handles[1],handles[0]]
labels = [ labels[1],labels[0]]
ax.legend(handles,labels,loc='upper left', bbox_to_anchor=(0,0.95),fontsize=15)

plt.show()
```



In [ ]:

D

For Figure 5 G and H

In [13]:
```
import pandas as pd
import numpy as np
import numpy as np
import os
import sys

import statsmodels.formula.api as smf
import statsmodels.tsa.api as smt
import statsmodels.api as sm
import scipy.stats as scs
from statsmodels.stats.diagnostic import acorr_ljungbox  # white noise test
from statsmodels.tsa.stattools import adfuller  #stationarity

import matplotlib.pyplot as plt
import matplotlib as mpl
```

In [14]:
```
#input the data from our file
file_data = pd.read_excel("Figures-prediction.xlsx",).iloc[:-6,:].set_index('Term')
file_data
```

Out[14]:

	Benign Chiari Malformation	Simple Chiari Malformation	Complex Chiari Malformation	Syringomyelia-Chiari complex	Unnamed: 5
Term					
1891-1950	0.0	5.0	0.0	0.0	NaN
1951	0.0	0.0	0.0	0.0	NaN
1952	0.0	0.0	0.0	0.0	NaN
1953	0.0	0.0	0.0	0.0	NaN
1954	0.0	0.0	0.0	0.0	NaN
...	...	...	...	...	...
2018	23.0	6.0	6.0	8.0	NaN
2019	25.0	15.0	14.0	7.0	NaN
2020	23.0	10.0	14.0	6.0	NaN
2021	26.0	8.0	10.0	10.0	NaN
2022	30.0	9.0	18.0	6.0	NaN

73 rows × 5 columns

In [15]:
```
#check the stationarity
test2 = sm.diff(file_data.loc[1986:,'Benign Chiari Malformation'])
print("The p-value of Unit Root Test is:",adfuller(test2,autolag='AIC')[1] )
```

```
The p-value of Unit Root Test is: 2.4589540053980065e-10
```

The data is stationary:

- The p-value is very closed to 0, so we can reject H0(There exists unit root)

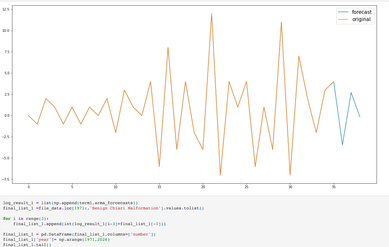
Autoregressive Moving Average Models - ARMA(p, q)

In [16]:
```
import warnings
warnings.filterwarnings('ignore')

import statsmodels.tsa.statistic as st

order_analyze = st.arma_order_select_ic(test2, max_ar=5, max_ma=5, ic='aic')
print('The order of the model with smallest aic value is',order_analyze.aic_min_order)
```

```
The order of the model with smallest aic value is: (1, 1)
```

In [17]:
```
import warnings
warnings.filterwarnings('ignore')

from statsmodels.tsa.arima_model import ARIMA
arma_mdl = ARIMA(test2,order=(1,0,1).fit()
print(arma_mdl.params)

#check whether residuals are white noise or not
pvalue_lb_test = acorr_ljungbox(arma_mdl.resid,lags=[10], boxpierce=False,return_df=False)[1][0]

if pvalue_lb_test < 0.05:
    print('Passed on 5% significance level, we have reason to suspect the residuals are not white noise.')
else:
    print('The residuals seem like white noise.')

#show the forecast parts of ARMA model
arma_forecasts,err95,ci95 = arma_mdl.forecast(steps=5)
print('The forecast of one-step-ahead momentum is: ',arma_forecasts)

The residuals seem like white noise.
The forecast of one-step-ahead momentum is: [2.14851958 2.71759959 -0.1619417 ]
```

In [18]:
```
plt.figure(figsize=(20,10))
plt.plot(np.append(test2,arma_forecasts),label='forecast')
plt.plot(test2,label='original')
plt.legend(fontsize=15)
plt.show()
```



In [19]:
```
long_result_1 = list(np.append(test2,arma_forecasts))
final_list_1 =file_data.loc[1971:,'Benign Chiari Malformation'].values.tolist()

for i in range(5):
    final_list_1.append(int(long_result_1[i-1]+final_list_1[-1]))

final_list_1 = pd.DataFrame(final_list_1,columns=['number'])
final_list_1['year'] = np.arange(1971,2026)
final_list_1.tail()
```

Out[19]:

	number	year
50	26.0	2021
51	30.0	2022
52	29.0	2023
53	28.0	2024
54	27.0	2025

In [20]:
```
#check the stationarity
test2 = sm.diff(file_data.loc[1981:,'Complex Chiari Malformation'])
print("The p-value of Unit Root Test is:",adfuller(test2,autolag='AIC')[1] )
```

```
The p-value of Unit Root Test is: 4.4355606562747e-11
```

In [21]:
```
import warnings
warnings.filterwarnings('ignore')

import statsmodels.tsa.statistic as st


order_analyze = st.arma_order_select_ic(test2, max_ar=5, max_ma=5, ic='aic')
print('The order of the model with smallest aic value is',order_analyze.aic_min_order)

The order of the model with smallest aic value is: (5, 2)
```

In [22]:
```
import warnings
warnings.filterwarnings('ignore')

from statsmodels.tsa.arima_model import ARIMA
arma_mdl = ARIMA(test2,order=(5,0,2).fit()
print(arma_mdl.params)

#check whether residuals are white noise or not
pvalue_lb_test = acorr_ljungbox(arma_mdl.resid,lags=[10], boxpierce=False,return_df=False)[1][0]

if pvalue_lb_test < 0.05:
    print('Passed on 5% significance level, we have reason to suspect the residuals are not white noise.')
else:
    print('The residuals seem like white noise.')

#show the forecast parts of ARMA model
arma_forecasts,err95,ci95 = arma_mdl.forecast(steps=5)
print('The forecast of one-step-ahead momentum is: ',arma_forecasts)

The residuals seem like white noise.
The forecast of one-step-ahead momentum is: [-4.40642414 -1.45453392 0.37905365]
```

In [23]:
```
long_result_2 = list(np.append(test2,arma_forecasts))
final_list_2 =file_data.loc[1971:,'Complex Chiari Malformation'].values.tolist()

for i in range(5):
    final_list_2.append(int(long_result_2[i-1]+final_list_2[-1]))

final_list_2 = pd.DataFrame(final_list_2,columns=['number'])
final_list_2['year'] = np.arange(1971,2026)
final_list_2.tail()
```

Out[23]:

	number	year
50	16.0	2021
51	16.0	2022
52	14.0	2023
53	15.0	2024
54	16.0	2025

In [24]:
```
#check the stationarity
test2 = sm.diff(file_data.loc[1971:,'Simple Chiari Malformation'])
print("The p-value of Unit Root Test is:",adfuller(test2,autolag='AIC')[1] )
```

```
The p-value of Unit Root Test is: 1.2720300353429Se-12
```

In [25]:
```
import warnings
warnings.filterwarnings('ignore')
import statsmodels.tsa.statistic as st
order_analyze = st.arma_order_select_ic(test2, max_ar=5, max_ma=5, ic='aic')
print('The order of the model with smallest aic value is',order_analyze.aic_min_order)

The order of the model with smallest aic value is: (2, 2)
```

In [26]:
```
import warnings
warnings.filterwarnings('ignore')

from statsmodels.tsa.arima_model import ARIMA
arma_mdl = ARIMA(test2,order=(2,0,2).fit()
print(arma_mdl.params)

#check whether residuals are white noise or not
pvalue_lb_test = acorr_ljungbox(arma_mdl.resid,lags=[10], boxpierce=False,return_df=False)[1][0]

if pvalue_lb_test < 0.05:
    print('Passed on 5% significance level, we have reason to suspect the residuals are not white noise.')
else:
    print('The residuals seem like white noise.')

#show the forecast parts of ARMA model
arma_forecasts,err95,ci95 = arma_mdl.forecast(steps=5)
print('The forecast of one-step-ahead momentum is: ',arma_forecasts)

The residuals seem like white noise.
The forecast of one-step-ahead momentum is: [1.25808220 0.6410253 0.30930422]
```

In [27]:
```
plt.figure(figsize=(20,10))
plt.plot(np.append(test2,arma_forecasts),label='forecast')
plt.plot(test2,label='original')
plt.legend(fontsize=15)
plt.show()
```

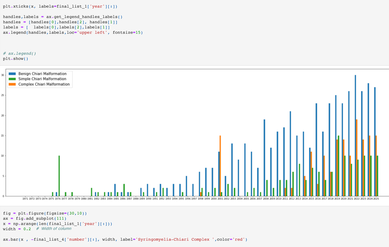
In [28]:
```
long_result_2 = list(np.append(test2,arma_forecasts))
final_list_2 =file_data.loc[1971:,'Simple Chiari Malformation'].values.tolist()

for i in range(5):
    final_list_2.append(int(long_result_2[i-1]+final_list_2[-1]))

final_list_2 = pd.DataFrame(final_list_2,columns=['number'])
final_list_2['year'] = np.arange(1971,2026)
final_list_2.tail()
```

Out[28]:

	number	year
50	8.0	2021
51	6.0	2022
52	7.0	2023
53	8.0	2024
54	8.0	2025

In [29]:
```
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111)
x = np.arange(final_list_1['year'][:])
width = 0.2 # width of column

ax.bar(x - width*1, final_list_1['number'][:], width, label='Benign Chiari Malformation')
ax.bar(x , final_list_2['number'][:], width, label='Complex Chiari Malformation ')
ax.bar(x+ width*1, final_list_3['number'][:], width, label='Simple Chiari Malformation')


plt.xticks(x, labels=final_list_1['year'][:])

handles,labels = ax.get_legend_handles_labels()
handles = [handles[0],handles[1], handles[2]]
labels = [ labels[0],labels[1],labels[2]]
ax.legend(handles,labels,loc='upper left', fontsize=15)


# ax.legend()
plt.show()
```

In [30]:
```
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111)
x = np.arange(final_list_1['year'][:])
width = 0.2 # width of column

ax.bar(x , -final_list_4['number'][:], width, label='Syringomyelia-Chiari Complex ',color='red')

plt.xticks(x, labels=final_list_1['year'][:])

# handles,labels = ax.get_legend_handles_labels()
# handles = [handles[0],handles[1], handles[2]]
# labels = [ labels[0],labels[1],labels[2]]
# ax.legend(handles,labels,loc='upper left', fontsize=15)


ax.legend()
plt.show()
```



In [31]:
```
```

E

For Figure 5 I

In [1]:
```python
import pandas as pd
import datetime
import numpy as np
import os
import sys

import statsmodels.formula.api as smf
import statsmodels.tsa.api as smt
import statsmodels.api as sm
import scipy.stats as scs
from statsmodels.stats.diagnostic import acorr_ljungbox  # white noise test
from statsmodels.tsa.stattools import adfuller   #stationarity

import matplotlib.pyplot as plt
import matplotlib as mpl
```

In [2]:
```python
#input the data from csv file
file_data = pd.read_excel("Figure5-prediction.xlsx").iloc[:-6,:].set_index('Term')
file_data
```

Out[2]:

	Chiari-Like Malformation
Term	
1891-1950	1.0
1951	0.0
1952	0.0
1953	0.0
1954	0.0
...	...
2018	43.0
2019	40.0
2020	33.0
2021	37.0
2022	44.0

73 rows × 1 columns

In [3]:
```python
#check the stationarity
term1 = np.diff(file_data.loc[1986:,'Chiari-Like Malformation'])
print("The p-value of Unit Root Test is:",adfuller(term1,autolag='AIC')[1] )
```

```
The p-value of Unit Root Test is: 5.5190635734093215e-18
```

The data is stationary:

- The p-value is very closed to 0, so we can reject H0(There exists unit root)

Autoregressive Moving Average Models - ARMA(p, q)

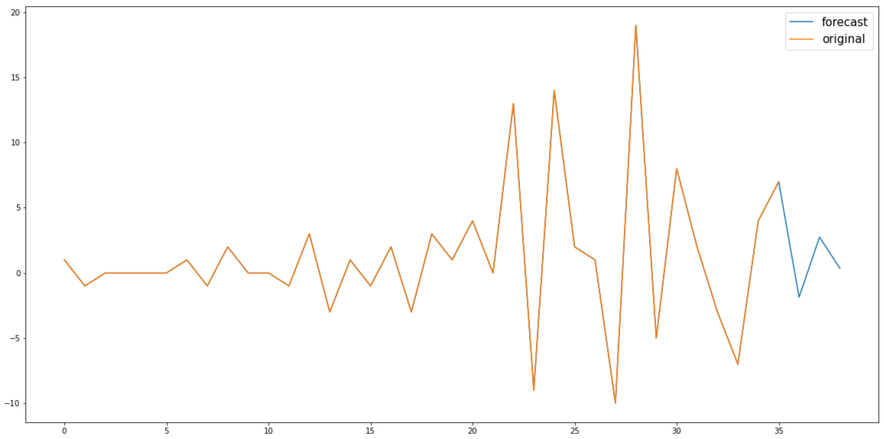
In [6]:
```python
import warnings
warnings.filterwarnings("ignore")

import statsmodels.tsa.stattools as st

order_analyze = st.arma_order_select_ic(term1, max_ar=5, max_ma=5, ic=['bic'])
print("The order of the model with smallest aic value is:",order_analyze.bic_min_order)
```

```
The order of the model with smallest aic value is: (1, 0)
```

In [7]:
```python
import warnings
warnings.filterwarnings("ignore")

from statsmodels.tsa.arima_model import ARIMA
arma_mdl = ARIMA(term1,order=(1,0,0)).fit()
params=arma_mdl.params

#check whether residuals are white noise or not
pvalue_LB_test = acorr_ljungbox(arma_mdl.resid,lags=[10], boxpierce=False,return_df=False)[1][0]

if pvalue_LB_test < 0.05:
    print('Basesd on 5% siginificance level, we have reason to suspect the residuals are not white noise.')
else:
    print('The residuals seem like white noise.')

#show the forecast parts of ARMA model
arma_forcecasts,err95,ci95 = arma_mdl.forecast(steps=3)
print("The forecast of one-step-ahead momentum is: ",arma_forcecasts)
```

```
The residuals seem like white noise.
The forecast of one-step-ahead momentum is:  [-1.86420322  2.7454502   0.34829118]
```

In [8]:
```python
plt.figure(figsize=(20,10))
plt.plot(np.append(term1,arma_forcecasts),label='forecast')
plt.plot(term1,label='original')
plt.legend(fontsize=15)
plt.show()
```

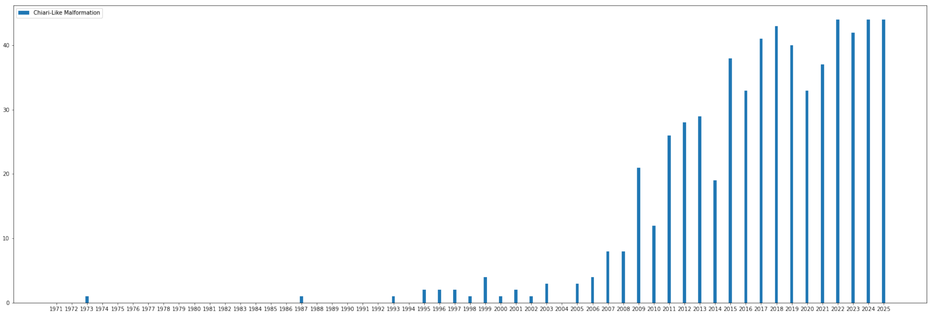


In [9]:
```python
log_result_1 = list(np.append(term1,arma_forcecasts))
final_list_1 =file_data.loc[1971:,'Chiari-Like Malformation'].values.tolist()

for i in range(3):
    final_list_1.append(int(log_result_1[i-3]+final_list_1[-1]))

final_list_1 = pd.DataFrame(final_list_1,columns=['number'])
final_list_1['year']= np.arange(1971,2026)
final_list_1.tail()
```

Out[9]:

	number	year
50	37.0	2021
51	44.0	2022
52	42.0	2023
53	44.0	2024
54	44.0	2025

In [10]:
```python
fig = plt.figure(figsize=(30,10))
ax = fig.add_subplot(111)
x = np.arange(len(final_list_1['year'][:]))
width = 0.2   # Width of column

ax.bar(x, final_list_1['number'][:], width, label='Chiari-Like Malformation')


plt.xticks(x, labels=final_list_1['year'][:])



ax.legend(loc='upper left')
plt.show()
```



In [ ]: