Seasonal ARIMA 모형 추정

SARIMAX (http://www.statsmodels.org/dev/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html) 클래스 이용하면 Multiplicated SARIMA(p,d,q)x(P,D,Q,s) 모형에 대한 추정 및 예측이 가능하다. 클래스 인스턴스를 생성하기 위해서는 `order` 인수에 (p,d,q) 튜플을, `seasonal_order` 인수에 (P,D,Q,s) 튜플을 넣는다. `SARIMAX` 의 `fit` 메서드는 모수를 추정하여 그 결과를 `SARIMAXResult` 클래스 인스턴스로 반환한다.

In [1]:

```
def yearfraction2datetime(yearfraction, startyear=0):
    import datetime
    import dateutil
    year = int(yearfraction) + startyear
    month = int(round(12 * (yearfraction - year)))
    delta = dateutil.relativedelta.relativedelta(months=month)
    date = datetime.datetime(year, 1, 1) + delta
    return date
```

호흡기질환 사망자 수

In [2]:

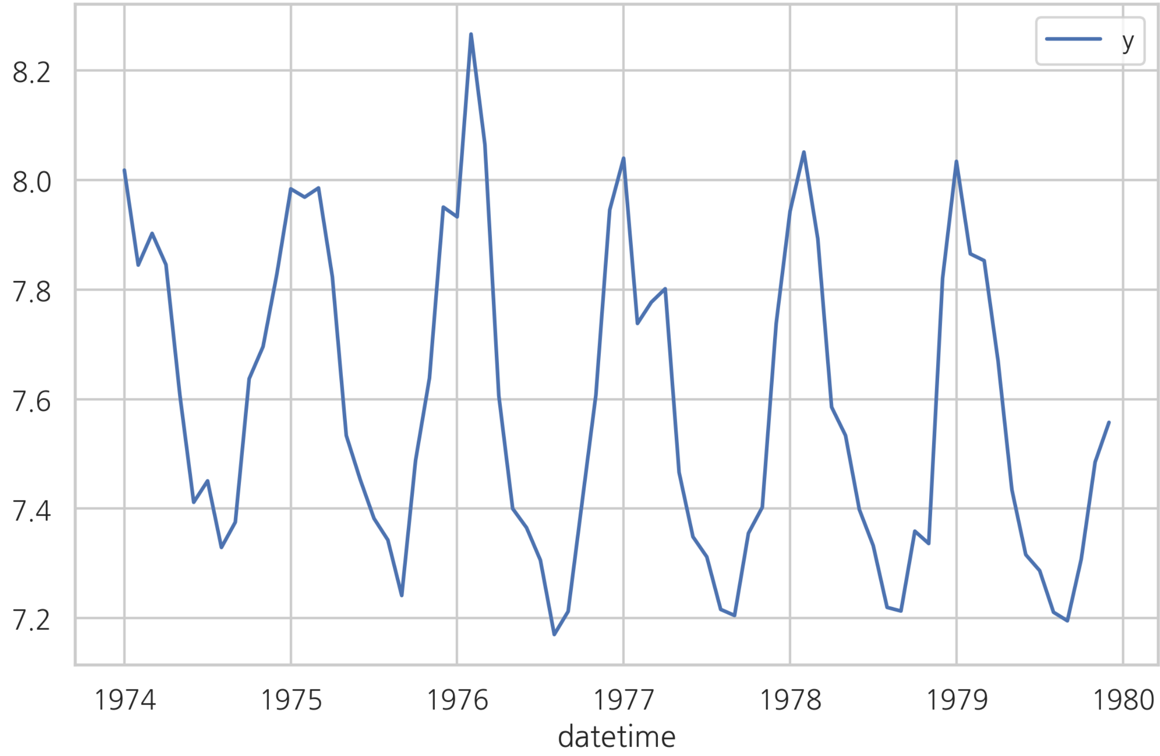
```
data = sm.datasets.get_rdataset("deaths", "MASS")
df = data.data
df["datetime"] = df.time.map(yearfraction2datetime)
df["month"] = df.datetime.dt.month
df["y"] = np.log(df.value)
df.tail()
```

Out[2]:

	time	value	datetime	month	y
67	1979.583333	1354	1979-08-01	8	7.210818
68	1979.666667	1333	1979-09-01	9	7.195187
69	1979.750000	1492	1979-10-01	10	7.307873
70	1979.833333	1781	1979-11-01	11	7.484930
71	1979.916667	1915	1979-12-01	12	7.557473

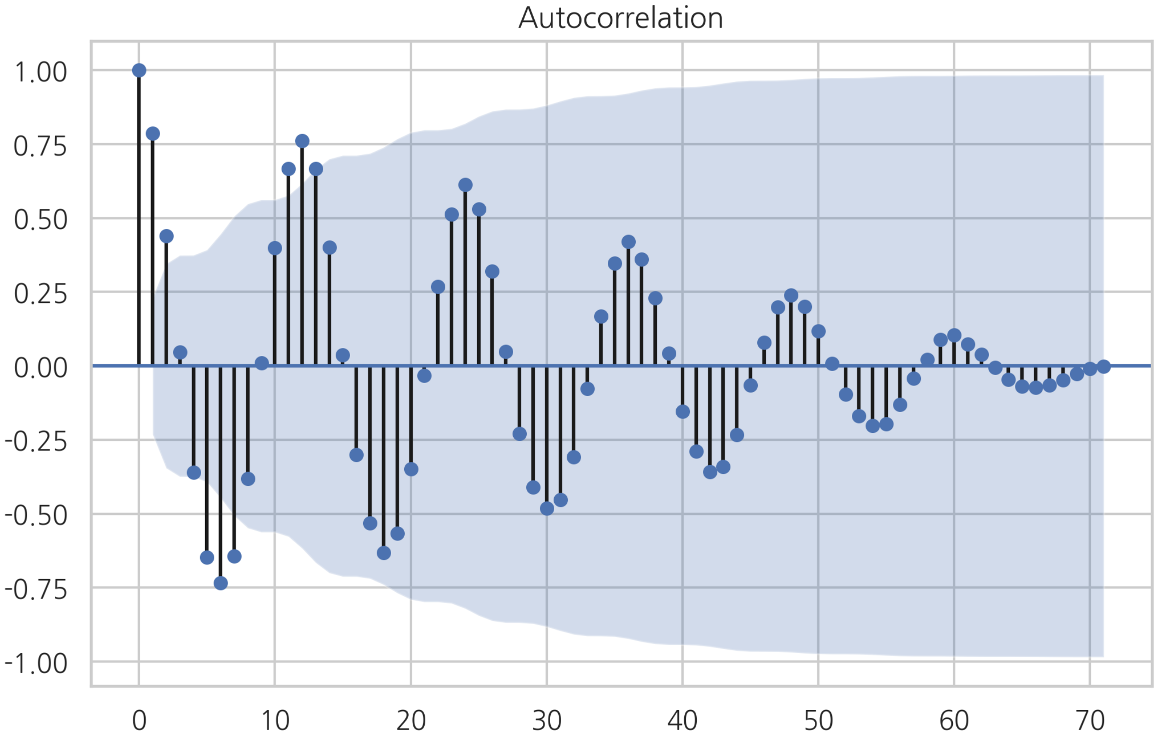
In [3]:

```
df.plot(x="datetime", y="y")
plt.show()
```



In [4]:

```
sm.tsa.graphics.plot_acf(df.y)
plt.show()
```



In [5]:

```
m = sm.tsa.SARIMAX(df.y, order=(1, 0, 0), seasonal_order=(1, 1, 1, 12))
r = m.fit()
print(r.summary())
```

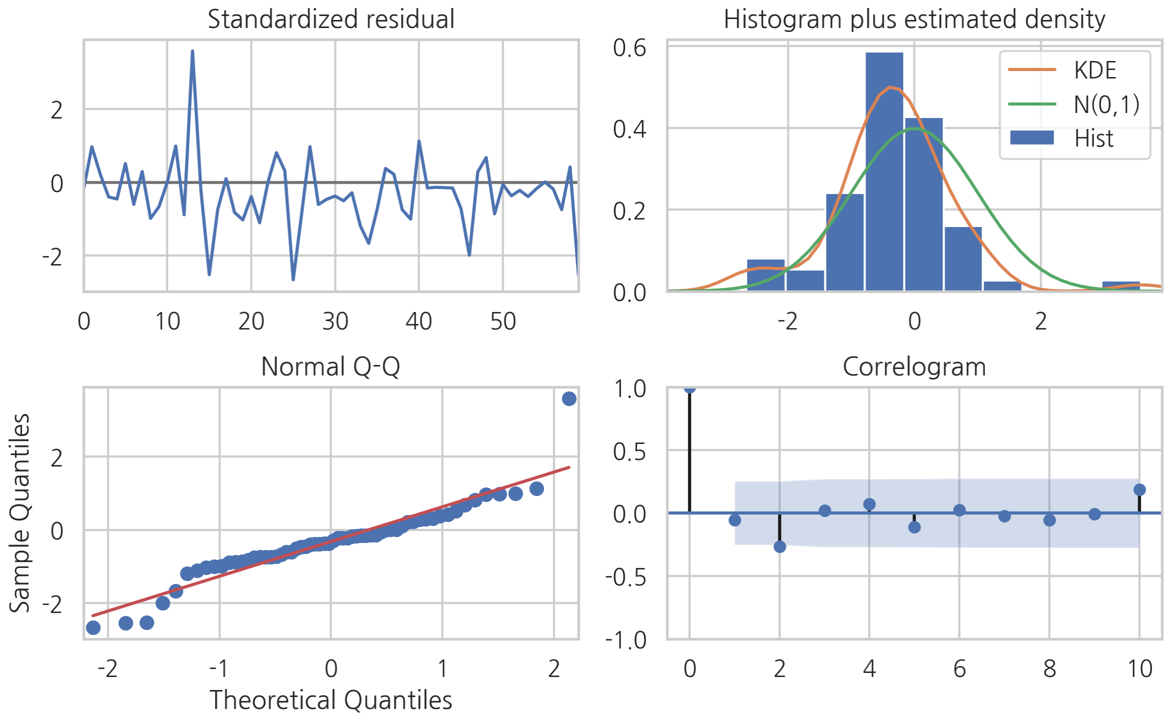
```
                                 Statespace Model Results                                 
==========================================================================================
======
Dep. Variable:                                  y   No. Observations:                     
72
Model:             SARIMAX(1, 0, 0)x(1, 1, 1, 12)   Log Likelihood                        
44.547
Date:                            Mon, 12 Nov 2018   AIC                                  -
81.094
Time:                                    22:16:09   BIC                                  -
72.716
Sample:                                         0   HQIC                                 -
77.817
                                             - 72                                         
Covariance Type:                              opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.4127      0.172      2.393      0.017       0.075       0.751
ar.S.L12      -0.3393      0.252     -1.348      0.178      -0.833       0.154
ma.S.L12      -0.4780      0.313     -1.529      0.126      -1.091       0.135
sigma2         0.0115      0.002      6.110      0.000       0.008       0.015
===================================================================================
Ljung-Box (Q):                       31.13   Jarque-Bera (JB):                43.74
Prob(Q):                              0.84   Prob(JB):                         0.00
Heteroskedasticity (H):               0.55   Skew:                             0.57
Prob(H) (two-sided):                  0.19   Kurtosis:                         7.02
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-ste
p).
```

잔차의 정규성과 자기상관계수 함수는 SARIMAXResult 클래스의 plot_diagnostics 메서드로 살펴볼 수 있다.

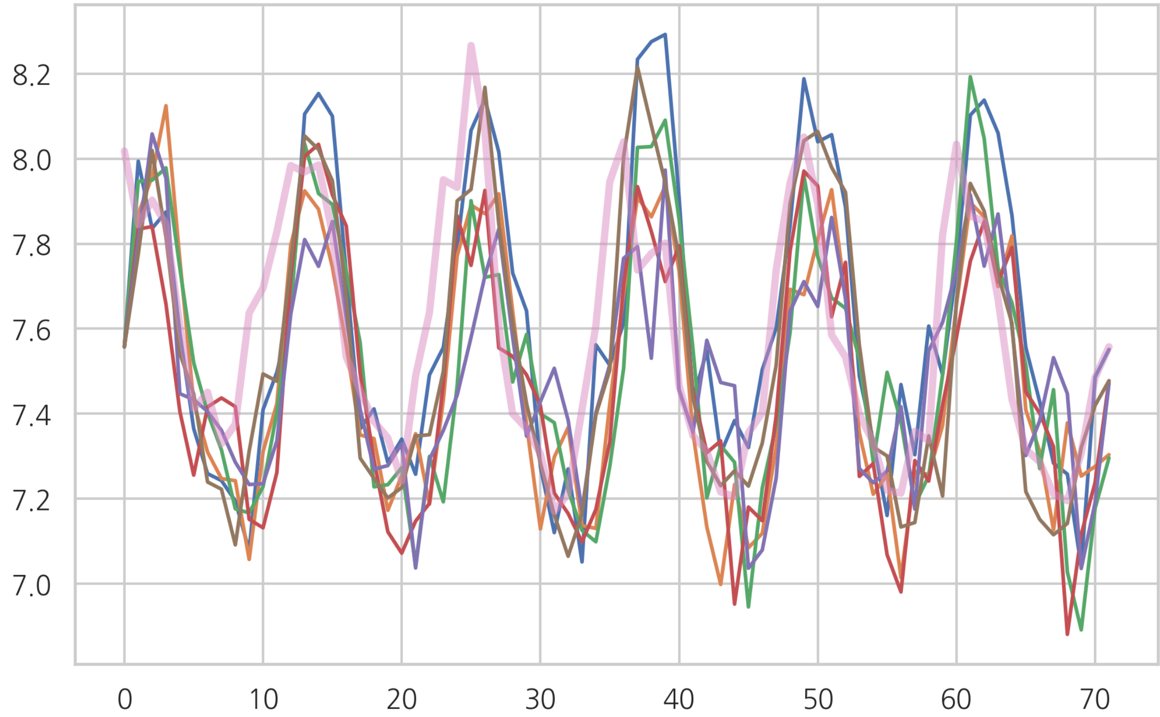
In [6]:

```
r.plot_diagnostics()
plt.tight_layout()
plt.show()
```



In [7]:

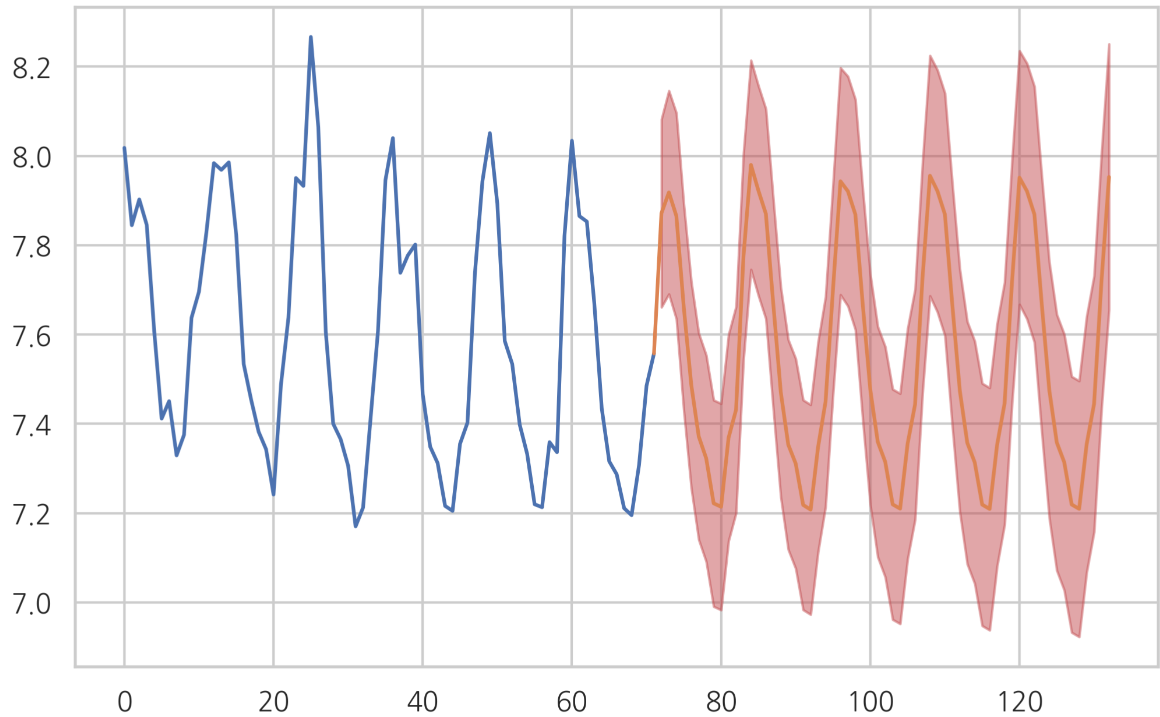
```
np.random.seed(0)
for i in range(6):
    plt.plot(r.simulate(len(df.y), initial_state=r.filtered_state[:, -1]))
plt.plot(df.y, lw=3, alpha=0.5)
plt.show()
```



In [8]:

```
horizon = 60
pred = r.get_prediction(start=len(df), end=len(df) + horizon)
s = df.y.copy()
s[:-1] = np.nan
s = np.hstack([s, pred.predicted_mean])
ci = pred.conf_int(alpha=0.05)

plt.plot(df.y)
plt.plot(s)
plt.fill_between(ci.index, ci["lower y"], ci["upper y"], color='r', alpha=0.5)
plt.show()
```



대기중 이산화탄소 농도 예측

In [9]:

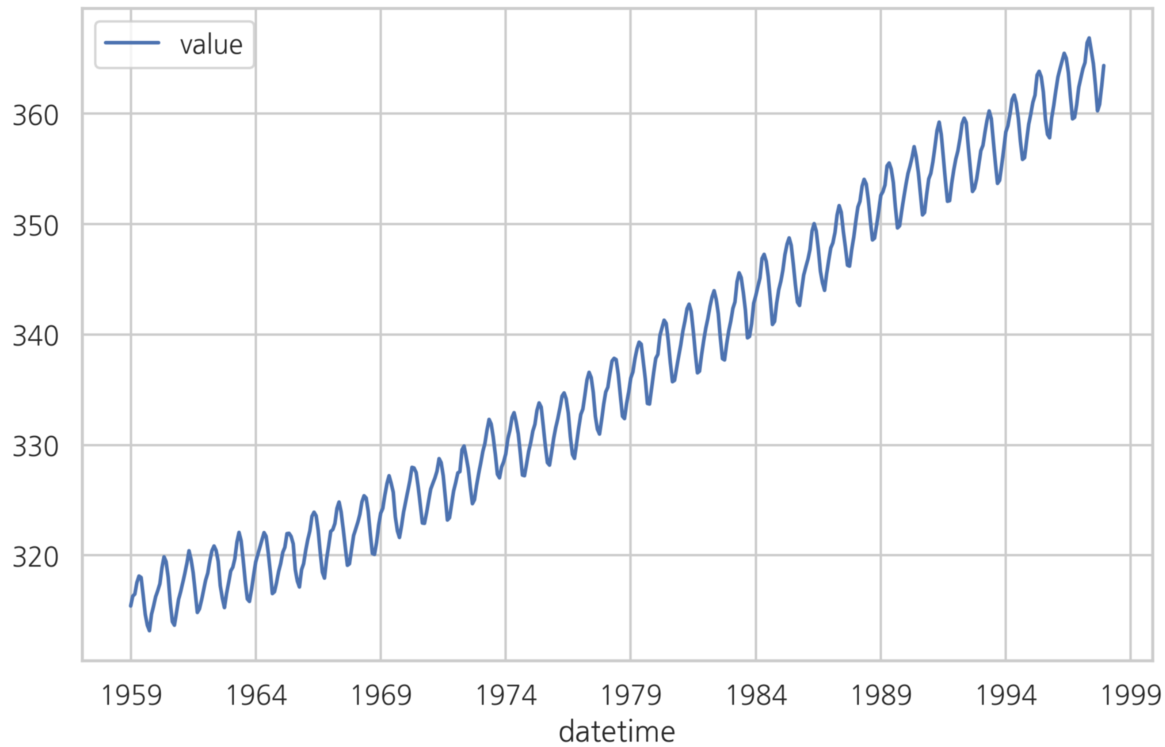
```
data = sm.datasets.get_rdataset("CO2")
df = data.data
df["datetime"] = df.time.map(yearfraction2datetime)
df["month"] = df.datetime.dt.month
df.tail()
```

Out[9]:

	time	value	datetime	month
463	1997.583333	362.57	1997-08-01	8
464	1997.666667	360.24	1997-09-01	9
465	1997.750000	360.83	1997-10-01	10
466	1997.833333	362.49	1997-11-01	11
467	1997.916667	364.34	1997-12-01	12

In [10]:

```
df.plot(x="datetime", y="value")
plt.show()
```



In [11]:

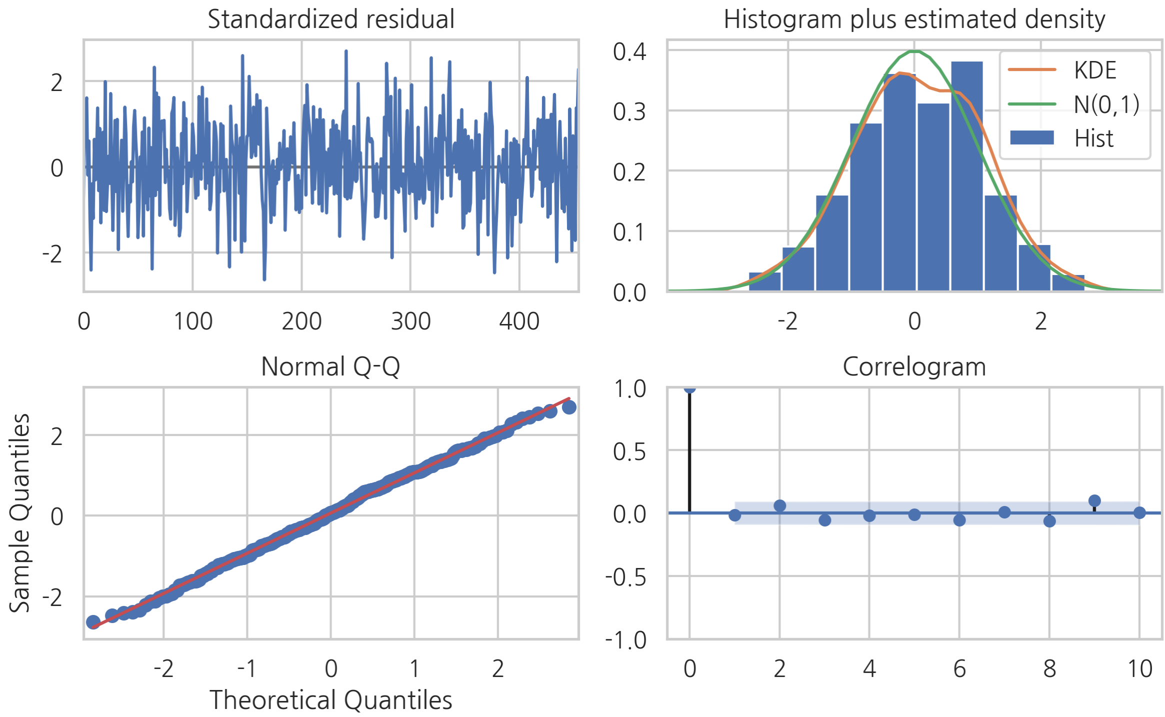
```
m = sm.tsa.SARIMAX(df.value, order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
r = m.fit()
print(r.summary())
```

```
                                 Statespace Model Results
==========================================================================================
======
Dep. Variable:                                value   No. Observations:
468
Model:             SARIMAX(1, 1, 1)x(1, 1, 1, 12)   Log Likelihood                      -
84.882
Date:                            Mon, 12 Nov 2018   AIC                                 1
79.763
Time:                                    22:16:16   BIC                                 2
00.365
Sample:                                         0   HQIC                                1
87.879
                                            - 468
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.2455      0.119      2.065      0.039       0.012       0.478
ma.L1         -0.5747      0.103     -5.596      0.000      -0.776      -0.373
ar.S.L12       0.0299      0.056      0.532      0.595      -0.080       0.140
ma.S.L12      -0.8582      0.033    -25.975      0.000      -0.923      -0.793
sigma2         0.0822      0.006     13.919      0.000       0.071       0.094
===================================================================================
Ljung-Box (Q):                       37.11   Jarque-Bera (JB):                 1.71
Prob(Q):                              0.60   Prob(JB):                         0.43
Heteroskedasticity (H):               0.93   Skew:                            -0.05
Prob(H) (two-sided):                  0.68   Kurtosis:                         2.72
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-ste
p).
```

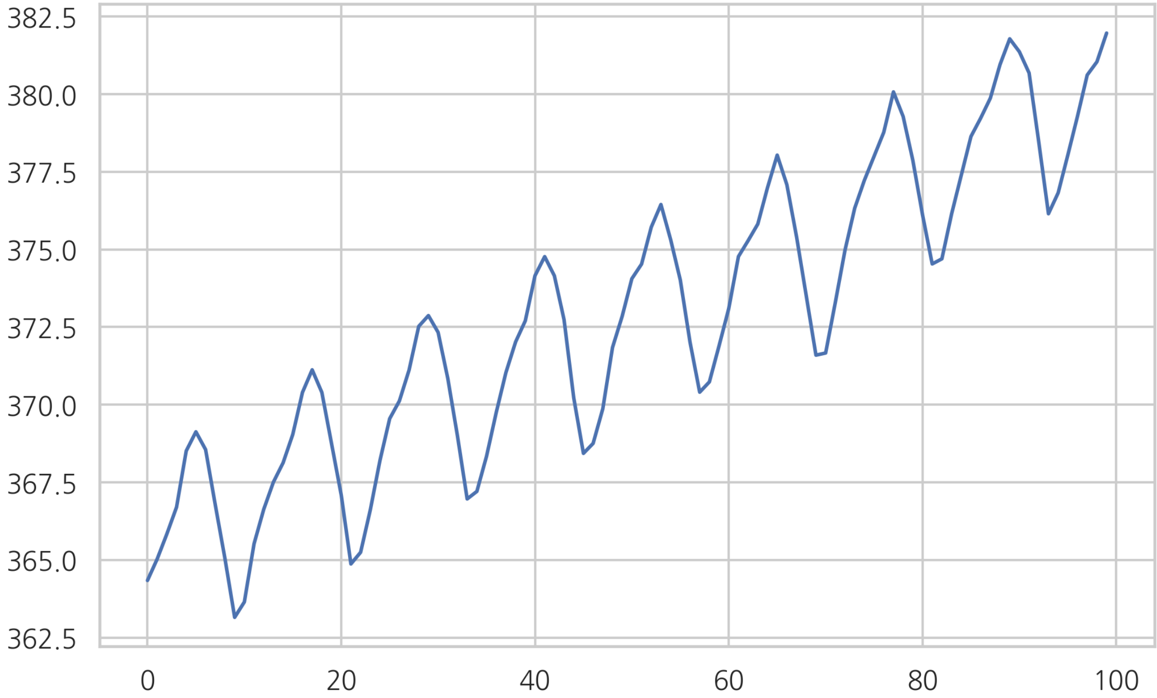
In [12]:

```
r.plot_diagnostics()
plt.tight_layout()
plt.show()
```



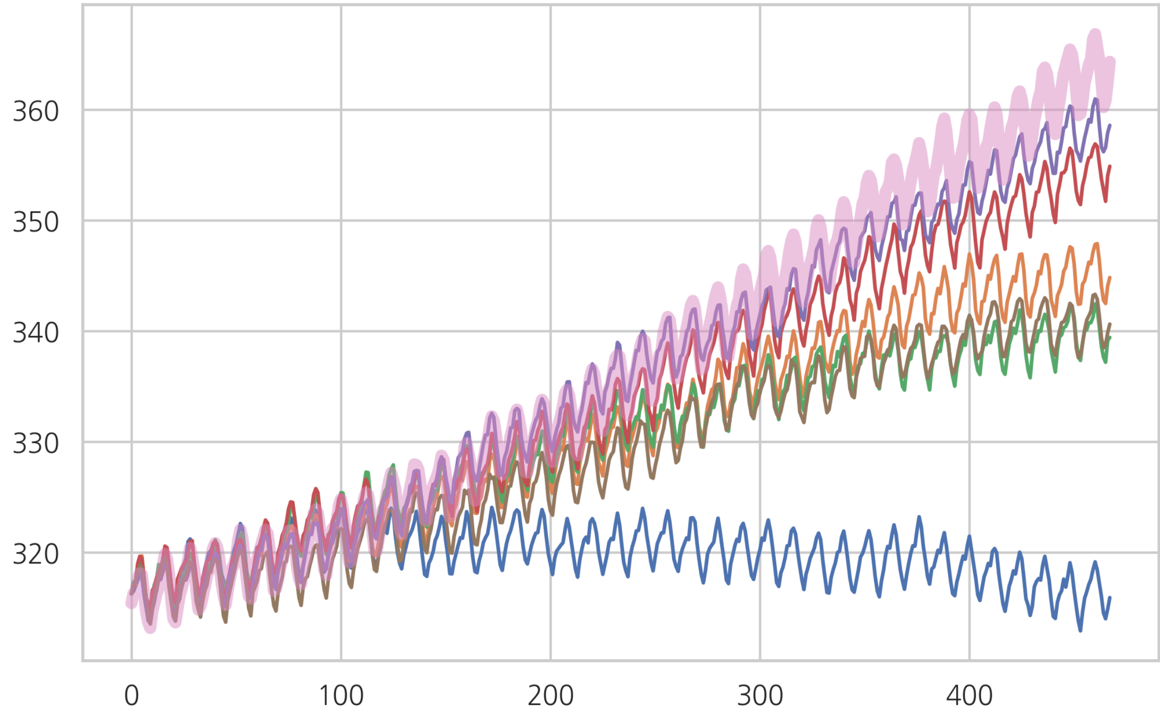
In [13]:

```
plt.plot(r.simulate(100, initial_state=r.filtered_state[:, -1]))
plt.show()
```



In [14]:

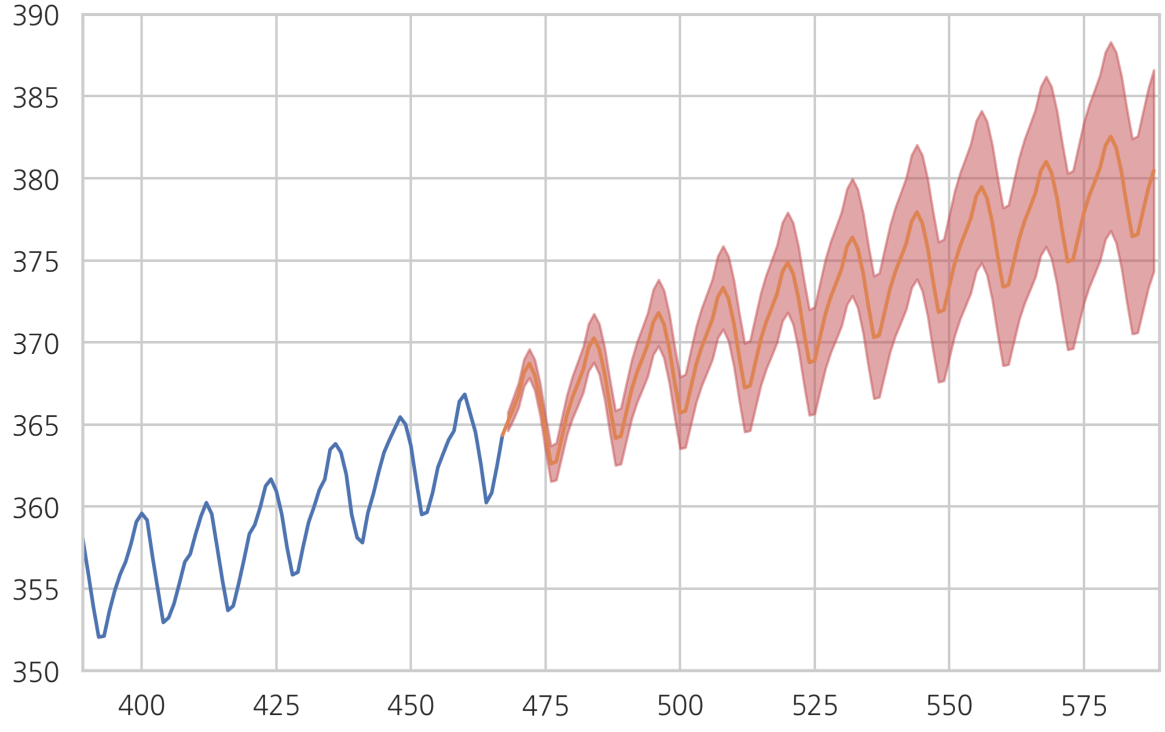
```
np.random.seed(0)
initial_state = r.filtered_state[:, 12]
for i in range(6):
    plt.plot(r.simulate(len(df.value), initial_state=initial_state))
plt.plot(df.value, lw=5, alpha=0.5)
plt.show()
```



In [15]:

```
horizon = 120
pred = r.get_prediction(start=len(df), end=len(df) + horizon)
s = df.value.copy()
s[:-1] = np.nan
s = np.hstack([s, pred.predicted_mean])
ci = pred.conf_int(alpha=0.05)

plt.plot(df.value)
plt.plot(s)
plt.fill_between(ci.index, ci["lower value"], ci["upper value"], color='r', alpha=0.5)
plt.xlim(len(s) - 200, len(s))
plt.ylim(350, 390)
plt.show()
```



항공운송

In [16]:

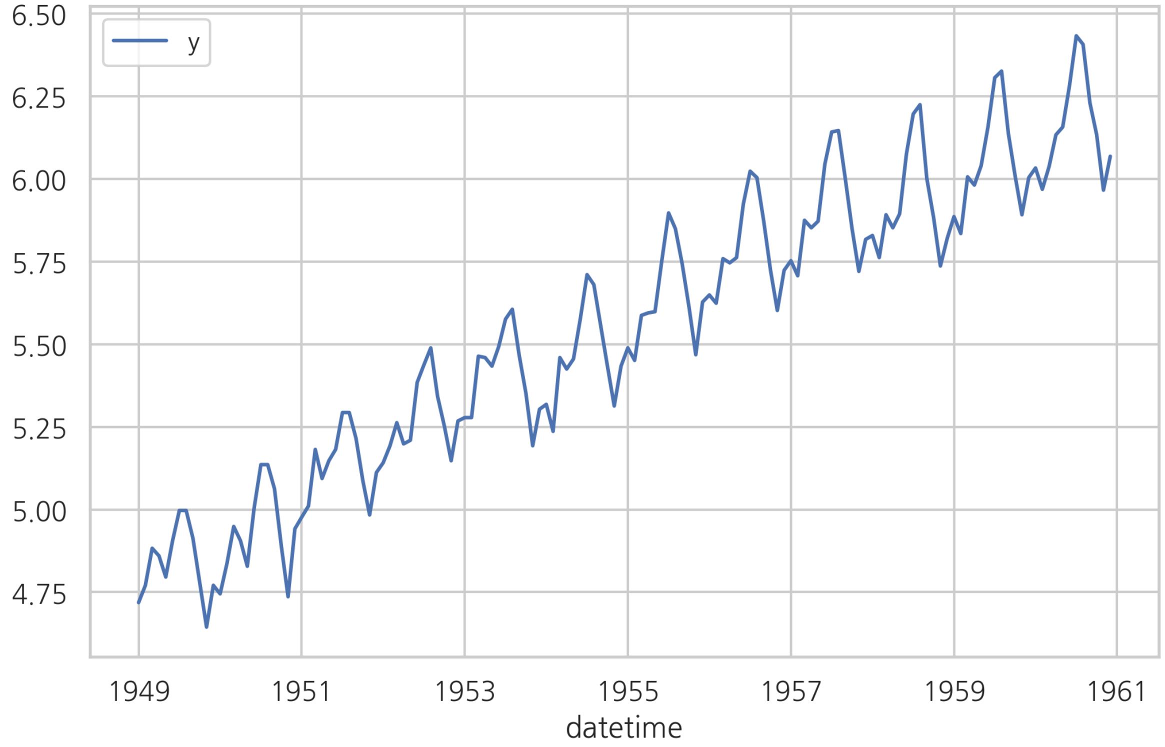
```
data = sm.datasets.get_rdataset("AirPassengers")
df = data.data
df["datetime"] = df.time.map(yearfraction2datetime)
df["y"] = np.log(df.value)
df["month"] = df.datetime.dt.month
df.tail()
```

Out[16]:

	time	value	datetime	y	month
139	1960.583333	606	1960-08-01	6.406880	8
140	1960.666667	508	1960-09-01	6.230481	9
141	1960.750000	461	1960-10-01	6.133398	10
142	1960.833333	390	1960-11-01	5.966147	11
143	1960.916667	432	1960-12-01	6.068426	12

In [17]:

```
df.plot(x="datetime", y="y")
plt.show()
```



In [18]:

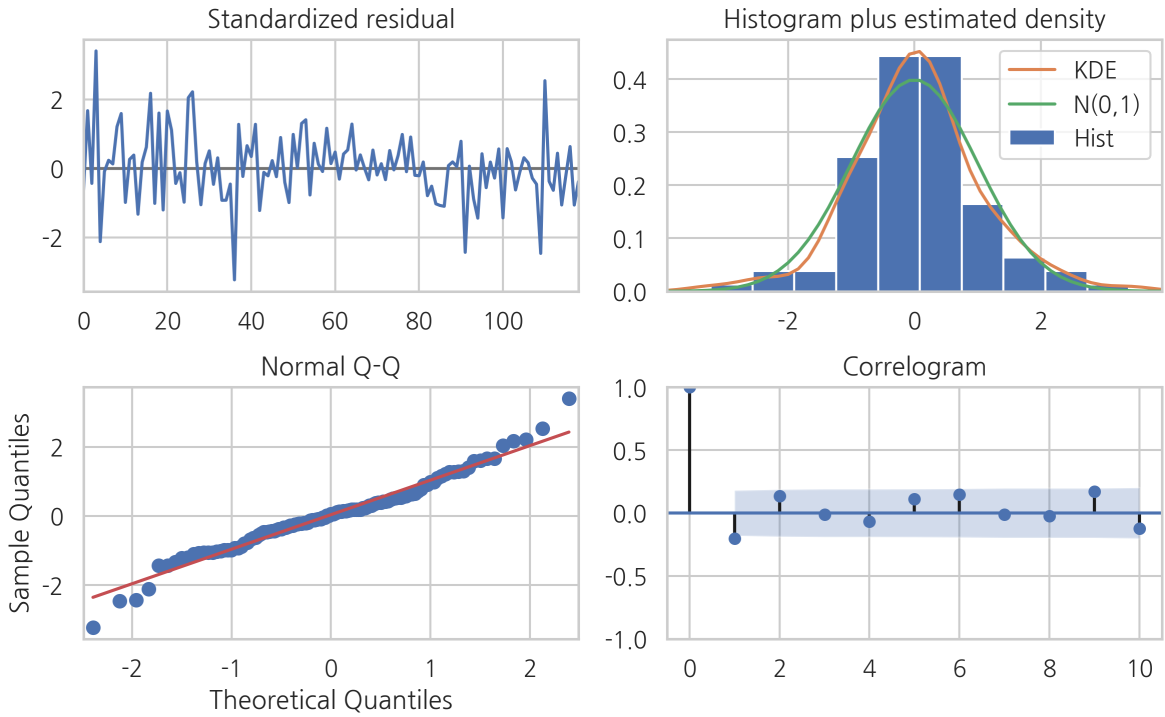
```
m = sm.tsa.SARIMAX(df.y, trend="c", order=(1, 0, 0), seasonal_order=(1, 1, 1, 12),
                   enforce_stationarity=False, enforce_invertibility=False)
r = m.fit()
print(r.summary())
```

```
                                 Statespace Model Results
==========================================================================================
======
Dep. Variable:                                  y   No. Observations:
144
Model:             SARIMAX(1, 0, 0)x(1, 1, 1, 12)   Log Likelihood                      2
19.171
Date:                            Mon, 12 Nov 2018   AIC                                -4
28.342
Time:                                    22:16:22   BIC                                -4
14.446
Sample:                                         0   HQIC                               -4
22.699
                                            - 144
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0285      0.008      3.766      0.000       0.014       0.043
ar.L1          0.7656      0.052     14.690      0.000       0.663       0.868
ar.S.L12      -0.0509      0.174     -0.293      0.769      -0.391       0.289
ma.S.L12      -0.5519      0.195     -2.825      0.005      -0.935      -0.169
sigma2         0.0014      0.000      9.116      0.000       0.001       0.002
===================================================================================
Ljung-Box (Q):                       55.97   Jarque-Bera (JB):                 9.86
Prob(Q):                              0.05   Prob(JB):                         0.01
Heteroskedasticity (H):               0.51   Skew:                             0.10
Prob(H) (two-sided):                  0.03   Kurtosis:                         4.40
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-ste
p).
```

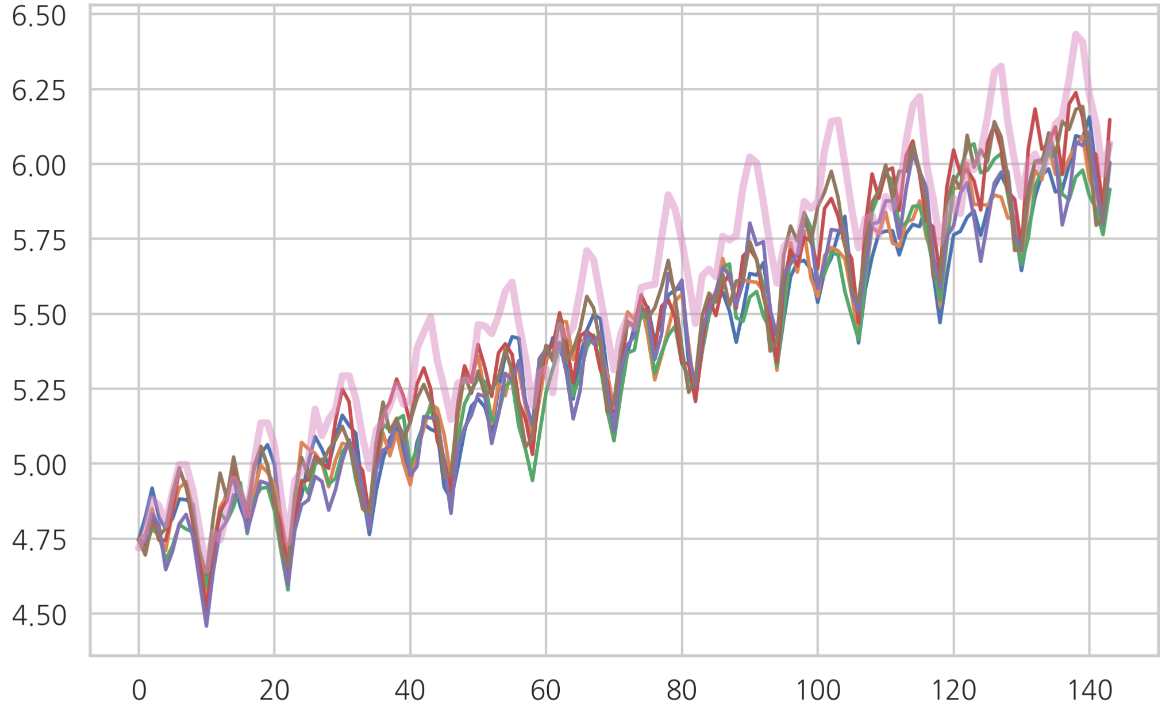
In [19]:

```
r.plot_diagnostics()
plt.tight_layout()
plt.show()
```



In [20]:

```
np.random.seed(0)
initial_state = r.filtered_state[:, 12]
for i in range(6):
    plt.plot(r.simulate(len(df.y), initial_state=initial_state))
plt.plot(df.y, lw=3, alpha=0.5)
plt.show()
```



In [21]:

```
horizon = 360
pred = r.get_prediction(start=len(df), end=len(df) + horizon)
s = df.y.copy()
s = np.hstack([s, pred.predicted_mean])
ci = pred.conf_int(alpha=0.05)

plt.plot(df.y)
plt.plot(s)
plt.fill_between(ci.index, ci["lower y"], ci["upper y"], color='r', alpha=0.5)
plt.xlim(len(s) - 500, len(s))
plt.show()
```