final

July 17, 2025

Author:Alejandro M.Ouslan

1 Probem

1.1 Problem A

[1]:
```
import polars as pl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import statsmodels.formula.api as smf
from statsmodels.tsa.ar_model import AutoReg
from statsmodels.tsa.stattools import acf
```

[2]:
```
df = pl.read_excel("data/us_macro_quarterly-1.xlsx")
df =df.with_columns(
        date=pl.col("column_0").str.replace(":0", "Q"),
        PCECTPI2=pl.col("PCECTPI").shift(1)

)
df = df.with_columns(
        infl=400*(pl.col("PCECTPI").log() - pl.col("PCECTPI2").log())
)
df
```

```
Could not determine dtype for column 10, falling back to string
Could not determine dtype for column 11, falling back to string
Could not determine dtype for column 12, falling back to string
Could not determine dtype for column 13, falling back to string
```

[2]: shape: (228, 16)

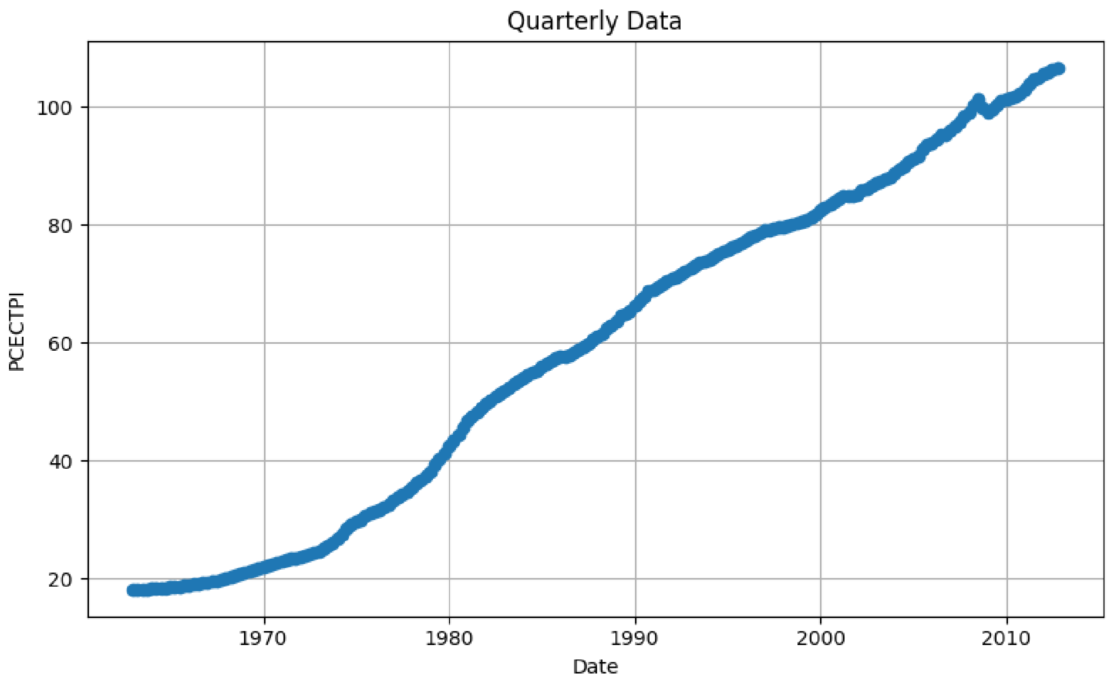
```
column_0   GDPC96     JAPAN_IP    PCECTPI   …  D_inf_lag   date     PCECTPI2
infl
---        ---        ---         ---          ---         ---      ---
---
```

```
str        f64        f64        f64        str        str      f64
f64


1957:01    2851.778   8.414363   16.449   …  null       1957Q1   null
null
1957:02    2845.453   9.097347   16.553   …  null       1957Q2   16.449
2.521068
1957:03    2873.169   9.042708   16.687   …  null       1957Q3   16.553
3.225048
1957:04    2843.718   8.796834   16.773   …  null       1957Q4   16.687
2.056191
1958:01    2770.0     8.632918   16.978   …  null       1958Q1   16.773
4.859175
…          …          …          …        …  …          …        …
…
2012:04    15539.628  94.258812  106.622  …  null       2012Q4   106.193
1.61267
2013:01    15583.948  94.72544   106.909  …  null       2013Q1   106.622
1.075254
2013:02    15679.677  95.992001  106.878  …  null       2013Q2   106.909
-0.116003
2013:03    15839.347  97.558537  107.387  …  null       2013Q3   106.878
1.900454
2013:04    15965.569  null       107.573  …  null       2013Q4   107.387
0.692222
```

[3]:
```python
data = df.to_pandas()
data['date'] = pd.PeriodIndex(data['date'], freq='Q')
data = data[(data["date"] >= "1963Q1") & (data["date"] <= "2012Q4")]
data.set_index('date', inplace=True)
```

1.2 Problem B

[4]:
```python
plt.figure(figsize=(8, 5))
plt.plot(data.index.to_timestamp(), data['PCECTPI'], marker='o')
plt.title('Quarterly Data')
plt.xlabel('Date')
plt.ylabel('PCECTPI')
plt.grid(True)
plt.tight_layout()
plt.show()
```



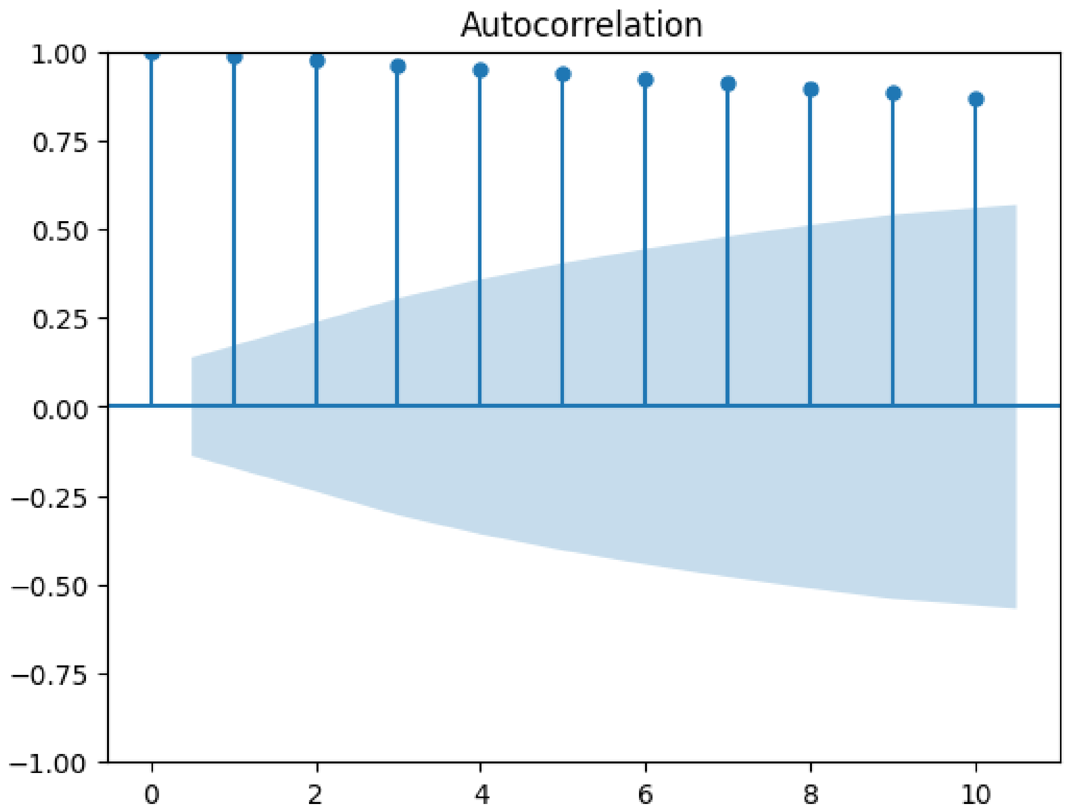
- The data seems to follow a trend but there is some stochastic elemnts

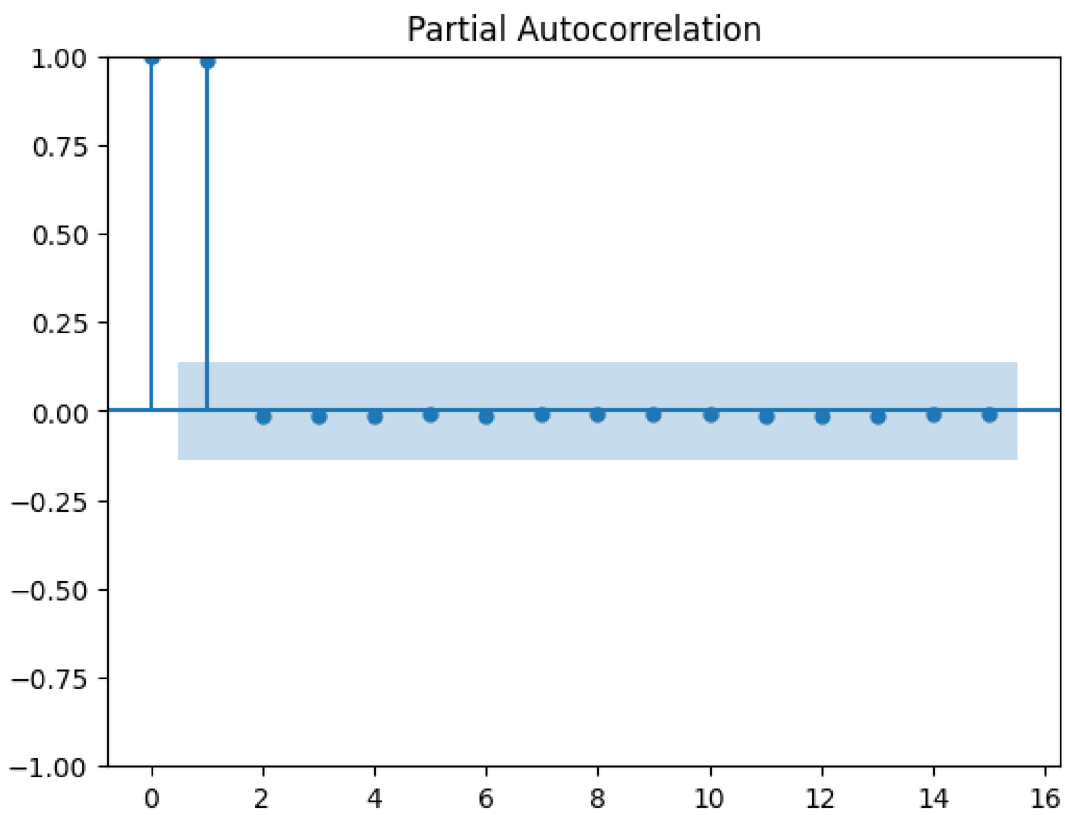
1.3 Probem C

[5]:
```
acf(data['PCECTPI'], nlags=4)
```

[5]:
```
array([1.        , 0.98759611, 0.97501513, 0.96230098, 0.94940062])
```

[6]:
```
fig = plot_acf(data['PCECTPI'], lags=10)
plt.show()
plot_pacf(data['PCECTPI'], lags=15)
plt.show()
```


Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10



1.4 Problem D

[7]:
```
data["pch"] = data["infl"].pct_change()
data["pch2"] = data["pch"].shift(1)
data
```

[7]:
```
        column_0     GDPC96     JAPAN_IP  PCECTPI      GS10       GS1      TB3MS  \
date
1963Q1  1963:01   3452.806   17.238516   18.069  3.893333  3.026667  2.906667
1963Q2  1963:02   3497.818   18.222013   18.095  3.963333  3.143333  2.940000
1963Q3  1963:03   3566.096   19.178191   18.181  4.033333  3.526667  3.293333
1963Q4  1963:04   3591.546   20.161688   18.248  4.120000  3.730000  3.496667
1964Q1  1964:01   3669.226   20.817353   18.336  4.180000  3.826667  3.530000
…         …         …         …        …         …         …         …
2011Q4  2011:04  15242.142  100.224981  104.880  2.046667  0.113333  0.013333
2012Q1  2012:01  15381.564  100.991584  105.471  2.036667  0.156667  0.066667
2012Q2  2012:02  15427.670   98.858428  105.750  1.823333  0.186667  0.086667
2012Q3  2012:03  15533.985   95.792017  106.193  1.643333  0.183333  0.103333
2012Q4  2012:04  15539.628   94.258812  106.622  1.706667  0.173333  0.086667

           UNRATE     EXUSUK    CPIAUCSL Inflation d_Inf D_inf_lag  PCECTPI2  \
```

```
date
1963Q1  5.766667  2.802933   30.476667      None  None      None    18.018
1963Q2  5.733333  2.800167   30.533333      None  None      None    18.069
1963Q3  5.500000  2.799367   30.720000      None  None      None    18.095
1963Q4  5.566667  2.797367   30.803333      None  None      None    18.181
1964Q1  5.466667  2.797767   30.930000      None  None      None    18.248
…           …         …           …          …    …       …        …
2011Q4  8.633333  1.572033  226.971333      None  None      None   104.529
2012Q1  8.233333  1.571733  228.269333      None  None      None   104.880
2012Q2  8.200000  1.582667  228.841000      None  None      None   105.471
2012Q3  8.033333  1.581367  230.029667      None  None      None   105.750
2012Q4  7.833333  1.606433  231.277000      None  None      None   106.193

            infl       pch      pch2
date
1963Q1  1.130602       NaN       NaN
1963Q2  0.575158 -0.491282       NaN
1963Q3  1.896574  2.297486 -0.491282
1963Q4  1.471357 -0.224203  2.297486
1964Q1  1.924342  0.307869 -0.224203
…           …         …         …
2011Q4  1.340918 -0.408764 -0.377228
2012Q1  2.247678  0.676223 -0.408764
2012Q2  1.056714 -0.529864  0.676223
2012Q3  1.672150  0.582406 -0.529864
2012Q4  1.612670 -0.035571  0.582406

[200 rows x 17 columns]
```

[8]:

```python
results  = smf.ols("pch ~ pch2", data=data).fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    pch   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.005
Method:                 Least Squares   F-statistic:                   0.02451
Date:                Thu, 17 Jul 2025   Prob (F-statistic):              0.876
Time:                        16:25:50   Log-Likelihood:                -443.61
No. Observations:                 198   AIC:                             891.2
Df Residuals:                     196   BIC:                             897.8
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.1253      0.163     -0.770      0.442      -0.446       0.196
pch2           0.0112      0.071      0.157      0.876      -0.130       0.152
```

```
==============================================================================
Omnibus:                      386.479   Durbin-Watson:                   1.992
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           155902.324
Skew:                         -10.778   Prob(JB):                         0.00
Kurtosis:                     138.767   Cond. No.                         2.28
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

- INFO: Missing the interpretation

1.5 Problem E

[9]:
```python
res = AutoReg(data['pch'].dropna(), lags=1).fit()
print(res.summary())
```

```
                            AutoReg Model Results
==============================================================================
Dep. Variable:                    pch   No. Observations:                  199
Model:                     AutoReg(1)   Log Likelihood                -443.612
Method:               Conditional MLE   S.D. of innovations              2.274
Date:                Thu, 17 Jul 2025   AIC                            893.224
Time:                        16:25:50   BIC                            903.089
Sample:                    09-30-1963   HQIC                           897.217
                         - 12-31-2012
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.1253      0.162     -0.774      0.439      -0.443       0.192
pch.L1         0.0112      0.071      0.157      0.875      -0.128       0.150
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           89.4274           +0.0000j           89.4274            0.0000
-----------------------------------------------------------------------------
```

[10]:
```python
res = AutoReg(data['pch'].dropna(), lags=2).fit()
print(res.summary())
```

```
                            AutoReg Model Results
==============================================================================
Dep. Variable:                    pch   No. Observations:                  199
Model:                     AutoReg(2)   Log Likelihood                -441.299
Method:               Conditional MLE   S.D. of innovations              2.273
Date:                Thu, 17 Jul 2025   AIC                            890.597
Time:                        16:25:50   BIC                            903.730
```

```
Sample:                    12-31-1963   HQIC                         895.913
                         - 12-31-2012
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.1376      0.162     -0.847      0.397      -0.456       0.181
pch.L1         0.0121      0.071      0.170      0.865      -0.127       0.151
pch.L2        -0.0005      0.071     -0.007      0.995      -0.140       0.139
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           12.3513          -43.5578j           45.2751           -0.2060
AR.2           12.3513          +43.5578j           45.2751            0.2060
-----------------------------------------------------------------------------
```

- INFO: Missing intrepretation

1.6 Problem F

[11]:
```python
for i in range(0,9):
    test = res = AutoReg(data['pch'].dropna(), lags=i).fit()
    print(f"lag {i} : {test.bic}")
```

```
lag 0 : 901.340096678343
lag 1 : 903.0892978392201
lag 2 : 903.7300014969487
lag 3 : 905.4679452316498
lag 4 : 907.0816117766385
lag 5 : 908.6810123639299
lag 6 : 910.2351510536212
lag 7 : 911.940062783817
lag 8 : 913.6236143527094
```

- Looking at the AIC the the chocen lag is 0

1.7 Program G

[12]:
```python
data
```

[12]:
```
        column_0    GDPC96   JAPAN_IP  PCECTPI      GS10       GS1     TB3MS  \
date
1963Q1   1963:01  3452.806  17.238516   18.069  3.893333  3.026667  2.906667
1963Q2   1963:02  3497.818  18.222013   18.095  3.963333  3.143333  2.940000
1963Q3   1963:03  3566.096  19.178191   18.181  4.033333  3.526667  3.293333
1963Q4   1963:04  3591.546  20.161688   18.248  4.120000  3.730000  3.496667
1964Q1   1964:01  3669.226  20.817353   18.336  4.180000  3.826667  3.530000
…             …         …          …        …         …         …         …
2011Q4   2011:04 15242.142 100.224981  104.880  2.046667  0.113333  0.013333
```

```
2012Q1  2012:01  15381.564  100.991584  105.471  2.036667  0.156667  0.066667
2012Q2  2012:02  15427.670   98.858428  105.750  1.823333  0.186667  0.086667
2012Q3  2012:03  15533.985   95.792017  106.193  1.643333  0.183333  0.103333
2012Q4  2012:04  15539.628   94.258812  106.622  1.706667  0.173333  0.086667

          UNRATE    EXUSUK    CPIAUCSL Inflation d_Inf D_inf_lag  PCECTPI2  \
date
1963Q1  5.766667  2.802933   30.476667      None  None      None    18.018
1963Q2  5.733333  2.800167   30.533333      None  None      None    18.069
1963Q3  5.500000  2.799367   30.720000      None  None      None    18.095
1963Q4  5.566667  2.797367   30.803333      None  None      None    18.181
1964Q1  5.466667  2.797767   30.930000      None  None      None    18.248
…            …         …           …       …    …       …       …
2011Q4  8.633333  1.572033  226.971333      None  None      None   104.529
2012Q1  8.233333  1.571733  228.269333      None  None      None   104.880
2012Q2  8.200000  1.582667  228.841000      None  None      None   105.471
2012Q3  8.033333  1.581367  230.029667      None  None      None   105.750
2012Q4  7.833333  1.606433  231.277000      None  None      None   106.193

            infl       pch      pch2
date
1963Q1  1.130602       NaN       NaN
1963Q2  0.575158 -0.491282       NaN
1963Q3  1.896574  2.297486 -0.491282
1963Q4  1.471357 -0.224203  2.297486
1964Q1  1.924342  0.307869 -0.224203
…            …         …         …
2011Q4  1.340918 -0.408764 -0.377228
2012Q1  2.247678  0.676223 -0.408764
2012Q2  1.056714 -0.529864  0.676223
2012Q3  1.672150  0.582406 -0.529864
2012Q4  1.612670 -0.035571  0.582406

[200 rows x 17 columns]
```

[13]:
```
y = data['pch'].dropna()
res = AutoReg(data['pch'].dropna(), lags=2, trend="n").fit()
forecast_ar = res.predict(start=res.model._hold_back, end=len(y)+1)
forecast_ar.tail(1)
```

[13]:
```
2013Q2   -0.000087
Freq: Q-DEC, dtype: float64
```

2 Problem

2.1 Problem A

- The formula interpretation is incorrect since a monthly percentage change in IP would use $\frac{(IP_t - IP_{t-1})}{IP_{t-1}}$, not $\frac{ln(IP_t)}{IP_{t-1}}$. What the current model does is calculate a ratio between the current and prior month and log's it, which isn't the monthly change.

2.2 Problem B

- $$Y = 0.787 + 0.052(101.359) + 0.185(101.034) + 0.234(100.374) + 0.164(101.196) = 64.83$$

2.3 Problem C

- Let N = 324 (27 years times 12 for total months) The formulas for AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) are:
- **AIC** = ln(SSR / N) + (2 * AR(X + 1)) / N
- **BIC** = ln(SSR / N) + (ln(N) * AR(X + 1)) / N

Where:
- SSR = Sum of Squared Residuals
- N = Number of observations
- AR = Autoregressive model order

Given: - SSR = 19,533
- N = 324
- AR = 1

Then:

- **BIC** = ln(19,533 / 324) + (ln(324) * 1) / 324
- **AIC** = ln(19,533 / 324) + 2 / 324

AR	SSR	BIC	AIC
0	19,533	4.116958907	4.105289946
1	18,643	4.088166106	4.064828183
2	17,377	4.035684659	4.000677774
3	16,285	3.988623406	3.941947560
4	15,842	3.978885409	3.920540602
5	15,824	3.995590344	3.925576575
6	15,824	4.013432145	3.931749415

The results only slightly differ when using AIC versus BIC, but not dramatically.

3 Probelem

3.1 Problem A

[14]:
```python
def simulate_ar3_process(beta, n, reps):
    means = []
    cov_lag1 = []
    cov_lag2 = []
    cov_lag3 = []
    var_list = []

    for _ in range(reps):
        epsilon = np.random.normal(0, 1, n)
        y = np.zeros(n)
        for t in range(3, n):
            y[t] = beta * y[t-3] + epsilon[t]

        means.append(np.mean(y))
        y_centered = y - np.mean(y)

        var_list.append(np.mean(y_centered ** 2))
        cov_lag1.append(np.mean(y_centered[1:] * y_centered[:-1]))  # lag 1
        cov_lag2.append(np.mean(y_centered[2:] * y_centered[:-2]))  # lag 2
        cov_lag3.append(np.mean(y_centered[3:] * y_centered[:-3]))  # lag 3

    mean_of_means = np.mean(means)
    std_of_means = np.std(means, ddof=1)

    avg_var = np.mean(var_list)
    avg_cov_lag1 = np.mean(cov_lag1)
    avg_cov_lag2 = np.mean(cov_lag2)
    avg_cov_lag3 = np.mean(cov_lag3)

    # Compute autocorrelations
    rho_1 = avg_cov_lag1 / avg_var
    rho_2 = avg_cov_lag2 / avg_var
    rho_3 = avg_cov_lag3 / avg_var

    return mean_of_means, std_of_means,avg_cov_lag1, avg_cov_lag2,␣
 ↪avg_cov_lag3, rho_1, rho_2, rho_3

# Example usage
beta = 0.7
n = 1000
reps = 10000

mean_estimate, std_estimate, cov1, cov2, cov3, acf1, acf2, acf3 =␣
 ↪simulate_ar3_process(beta, n, reps)
```

3.2 Problem B

```
[15]: print(f"Std dev of E[y_t]: {std_estimate:.6f}")
```

```
Std dev of E[y_t]: 0.103983
```

3.3 Problem C

```
[16]: print(f"Estimated Covariance lag 1: {cov1:.6f}")
      print(f"Estimated Covariance lag 2: {cov2:.6f}")
      print(f"Estimated Covariance lag 3: {cov3:.6f}")
```

```
Estimated Covariance lag 1: -0.011819
Estimated Covariance lag 2: -0.012232
Estimated Covariance lag 3: 1.353486
```

3.4 Problem D

```
[17]: print(f"Estimated Autocorrelation lag 1: {acf1:.6f}")
      print(f"Estimated Autocorrelation lag 2: {acf2:.6f}")
      print(f"Estimated Autocorrelation lag 3: {acf3:.6f}")
```

```
Estimated Autocorrelation lag 1: -0.006097
Estimated Autocorrelation lag 2: -0.006310
Estimated Autocorrelation lag 3: 0.698244
```

4 Problem 4

4.1 Problem A

- $y_t = c + \phi_1 y_{t-1} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon t - 2$

4.2 Problem B

- $\hat{y}_{t+1|t} = \beta y_{t-2}$

4.3 Problem C

- $\lim_{h\to\infty} \hat{y}_{t+h|t} = 0$

4.4 Problem D

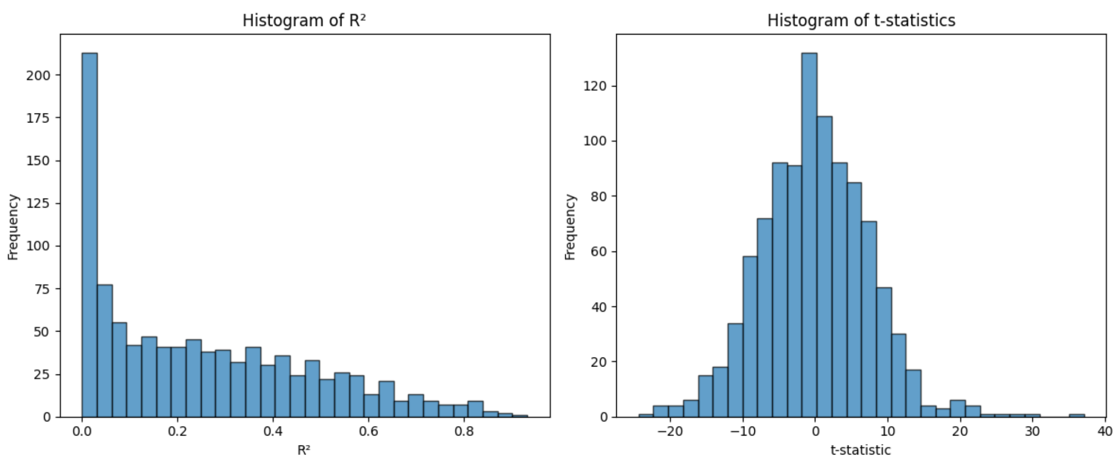
-$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_3 y_{t-3} + \epsilon_t$ - The model expresses the current data point as a linear combination of the last previous 3 values plus an error term

5 Problem

5.1 Problem A

[18]:
```
T = 100
num_simulations = 1000
rng = np.random.default_rng(787)

r_squared_vals = []
t_stats = []

for i in range(num_simulations):
    e = rng.normal(0, 1, T)
    a = rng.normal(0, 1, T)

    Y = np.zeros(T)
    X = np.zeros(T)

    Y[0] = e[0]
    X[0] = a[0]

    for t in range(1, T):
        Y[t] = Y[t - 1] + e[t]
        X[t] = X[t - 1] + a[t]

    df = pd.DataFrame({'Y': Y, 'X': X})
    results = smf.ols("Y ~ X", data=df).fit()

    r_squared_vals.append(results.rsquared)
    t_stats.append(results.tvalues['X'])

r_squared_vals = np.array(r_squared_vals)
t_stats = np.array(t_stats)


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

plt.subplot(1, 2, 1)
plt.hist(r_squared_vals, bins=30, edgecolor='k', alpha=0.7)
plt.title("Histogram of R²")
plt.xlabel("R²")
plt.ylabel("Frequency")

plt.subplot(1, 2, 2)
plt.hist(t_stats, bins=30, edgecolor='k', alpha=0.7)
plt.title("Histogram of t-statistics")
plt.xlabel("t-statistic")
plt.ylabel("Frequency")
```

```
plt.tight_layout()
plt.show()

r2_percentiles = np.percentile(r_squared_vals, [5, 50, 95])
t_stat_percentiles = np.percentile(t_stats, [5, 50, 95])


t_stat_exceeds_1_96 = np.mean(np.abs(t_stats) > 1.96)


print("R² percentiles (5%, 50%, 95%):", r2_percentiles)
print("t-statistic percentiles (5%, 50%, 95%):", t_stat_percentiles)
print(f"Fraction of |t| > 1.96: {t_stat_exceeds_1_96:.4f}")
```




```
R² percentiles (5%, 50%, 95%): [0.00156216 0.20518996 0.68473073]
t-statistic percentiles (5%, 50%, 95%): [-11.88825826  -0.2067511   11.36724655]
Fraction of |t| > 1.96: 0.7650
```