```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

df = pd.read_csv("dataset_bitcoin_returns (1).csv")
print(df.info())
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 1 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   returns  1000 non-null   float64
dtypes: float64(1)
memory usage: 7.9 KB
None
```

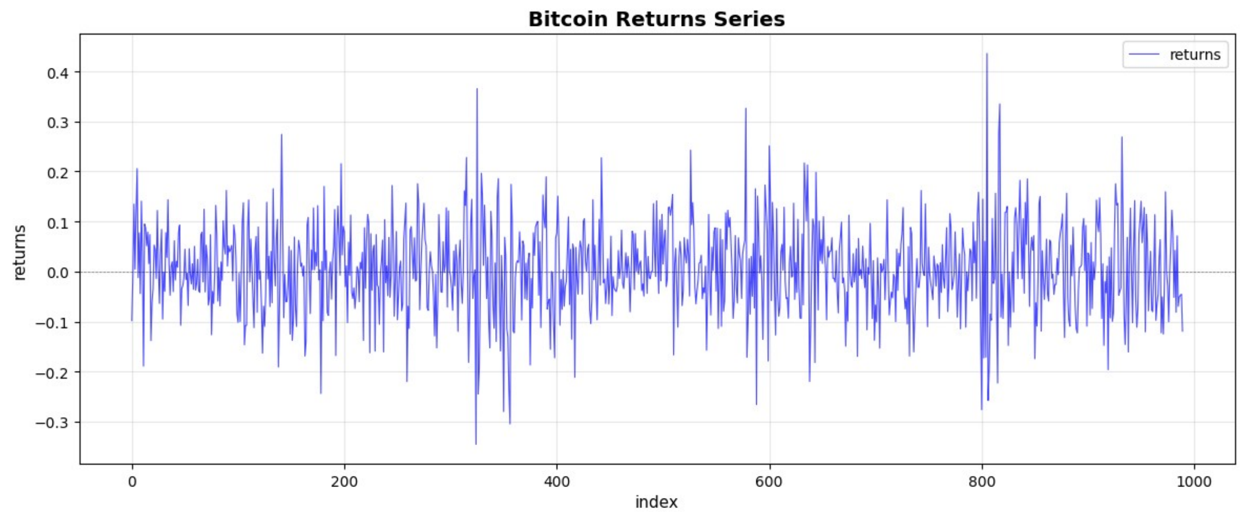
```
    returns
0 -0.098023
1 -0.006172
2  0.134714
3  0.004694
4  0.118912
```

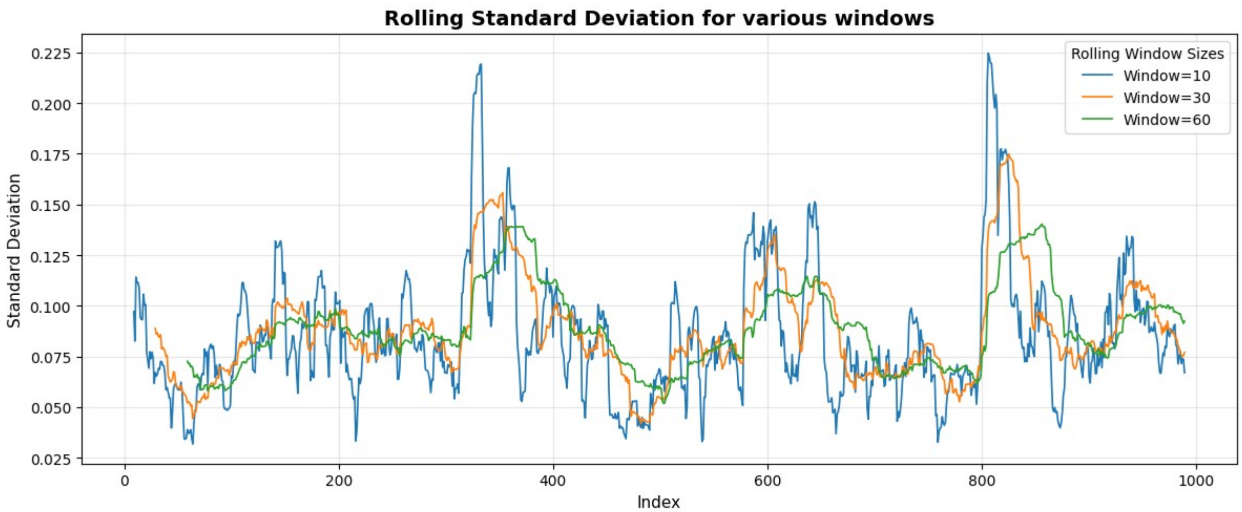
```python
train_split = int(len(df) - 10)
df_train = df.iloc[:train_split].copy()
df_test = df.iloc[train_split:].copy()
print("Train:", df_train.shape)
print("Test:", df_test.shape)
```

```
Train: (990, 1)
Test: (10, 1)
```

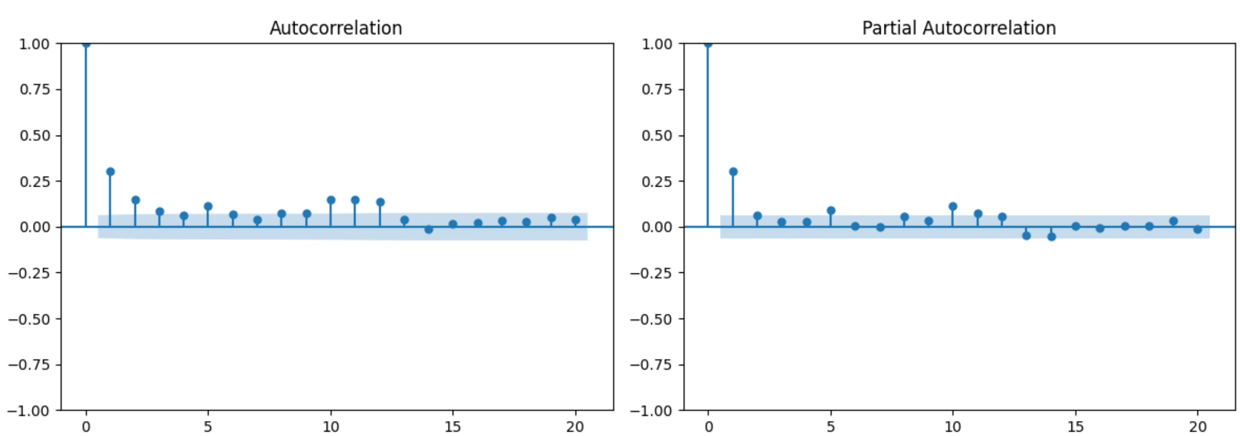
```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

def plot_series(df, title, xlabel='index', ylabel='returns',
rolling_windows=[30]):
    # Create figure with two subplots
    fig, axes = plt.subplots(2, 1, figsize=(12, 10))

    # Plot 1: Returns Series
    axes[0].plot(df.index, df[ylabel], color='blue', linewidth=0.8,
alpha=0.7, label=ylabel)
    axes[0].axhline(y=0, color='black', linestyle='--', linewidth=0.5,
alpha=0.5)
    axes[0].set_title(title, fontsize=14, fontweight='bold')
```

```
    axes[0].set_xlabel(xlabel, fontsize=11)
    axes[0].set_ylabel(ylabel, fontsize=11)
    axes[0].grid(True, alpha=0.3)
    axes[0].legend()

    # Plot 2: Rolling Standard Deviations
    # Ensure rolling_windows is a list, even if a single integer is
passed
    if not isinstance(rolling_windows, list):
        rolling_windows = [rolling_windows]

    for window in rolling_windows:
        rolling_std_name = f'rolling_std_{window}'
        df[rolling_std_name] = df[ylabel].rolling(window=window).std()
        axes[1].plot(df.index, df[rolling_std_name], linewidth=1.2,
label=f'Window={window}')


    axes[1].set_title(f'Rolling Standard Deviation for various
windows', fontsize=14, fontweight='bold')
    axes[1].set_xlabel('Index', fontsize=11)
    axes[1].set_ylabel('Standard Deviation', fontsize=11)
    axes[1].grid(True, alpha=0.3)
    axes[1].legend(title='Rolling Window Sizes') # Add a legend for
different windows

    plt.tight_layout()
    plt.show()

    # ACF and PACF plots for squared returns
    fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 5))
    sq_ret = (df[ylabel].dropna().values)**2 # Drop NA if there are
any from rolling calculations
    fig.suptitle(f'Autocorrelation and Partial Autocorrelation
Functions of Squared ({ylabel.upper()})', fontsize=16,
fontweight='bold')
    plot_acf(sq_ret, lags=20, ax=axes[0])
    plot_pacf(sq_ret, lags=20, ax=axes[1]);
    plt.tight_layout(rect=[0, 0.03, 1, 0.95]) # Adjust layout to
prevent title overlap
    plt.show()

plot_series(df_train, title='Bitcoin Returns Series',
rolling_windows=[10,30, 60])
```





Autocorrelation and Partial Autocorrelation Functions of Squared (RETURNS)



Bitcoin Returns Series

The Bitcoin Returns Series (top plot) shows a graph of returns hovering around 0 with varying volatility. There also seems to be clustering of volatility where large fluctuations often were followed by relatively similar large fluctuations--and vice versa.

The Rolling Standard Deviation for Various Windows (middle plot) confirms the conclusion above where each window exhibits consistently show elevated volatility during the same periods. This essentially confirms the presence of heteroskedasticity in the series where we have unequal volatility across time.

Naturally, the lesser the window size, the more volatile the series become while higher results to a smoother distribution. We can use this to our advantage when it comes to choosing parameters for GARCH model later.

The ACF and PACF plots of squared returns (bottom plot) provide a strong statistical evidence for conditional heteroskedasticity:

- The ACF shows a positive correlation of current squared returns (a proxy for volatility) with lagged squared returns--specifically the decaying autocorrelations. This indicates that volatility is not random and following a predictable pattern to some extent.
- The PACF shows a significant positive correlations, particularly for lags 1, 5, and 10--suggesting a direct relationship in current and past volatilities. Which is a characteristic of AR(p) model--maybe AR(1) because it has the most apparent Partial Autocorrelation.

These combined observations—volatility clustering, time-varying volatility, and the significant autocorrelations in squared returns—strongly justify the use of a GARCH-type model to capture and forecast the dynamic, conditional variance of Bitcoin returns.

Parameters

ARCH(1) - as was indicated earlier, the PACF shows a the most significant positive autocorrelation for lag 1--implying that the current volatility is directly proportional to the last observed volatility.

So let us first start with **ARCH(1)**. However, there seems to be persistency as was indicated earlier, "*where large fluctuations often were followed by relatively similar large fluctuations--and vice versa*". For this reason, it is worth choosing **GARCH(1,1)**. Which adds a component to model how past conditional variances affect current variance.

So in summary, **we'll test GARCH(1,0) and GARCH(1,1)**.

```
# import sys
# !{sys.executable} -m pip install arch

from arch import arch_model
```

ARCH(1) or GARCH(1,0)

```
from arch import arch_model
```

```
mod_10 = arch_model(df_train['returns'], vol='GARCH', p=1, q=0,
rescale=True)
res_10 = mod_10.fit(disp='off')

res_10.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                      Constant Mean - ARCH Model Results

======================================================================
========
Dep. Variable:                returns   R-squared:
0.000
Mean Model:             Constant Mean   Adj. R-squared:
0.000
Vol Model:                       ARCH   Log-Likelihood:
-1284.81
Distribution:                  Normal   AIC:
2575.63
Method:            Maximum Likelihood   BIC:
2590.32
                                        No. Observations:
990
Date:                Wed, Nov 19 2025   Df Residuals:
989
Time:                        16:53:56   Df Model:
1
                                  Mean Model

======================================================================
=======
                 coef    std err          t      P>|t|       95.0%
Conf. Int.
----------------------------------------------------------------------
-------
mu             0.0409  2.825e-02      1.447      0.148 [-1.450e-
02,9.625e-02]
                             Volatility Model

======================================================================
====
                 coef    std err          t      P>|t|    95.0% Conf.
Int.
----------------------------------------------------------------------
----
omega          0.6624  4.280e-02     15.478  4.900e-54   [  0.579,
0.746]
alpha[1]       0.1789  4.422e-02      4.044  5.248e-05 [9.218e-02,
0.266]
```

```
==============================================================================
====

Covariance estimator: robust
"""
```

GARCH(1,1)

```
mod_11 = arch_model(df_train['returns'], vol='GARCH', p=1, q=1,
rescale=True)
res_11 = mod_11.fit(disp='off')

res_11.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                     Constant Mean - GARCH Model Results

==============================================================================
========
Dep. Variable:                returns   R-squared:
0.000
Mean Model:             Constant Mean   Adj. R-squared:
0.000
Vol Model:                      GARCH   Log-Likelihood:
-1270.00
Distribution:                  Normal   AIC:
2547.99
Method:            Maximum Likelihood   BIC:
2567.58
                                        No. Observations:
990
Date:                Wed, Nov 19 2025   Df Residuals:
989
Time:                        16:54:01   Df Model:
1
                                  Mean Model

==============================================================================
=======
                 coef    std err          t      P>|t|       95.0%
Conf. Int.
------------------------------------------------------------------------------
-------
mu             0.0468  2.620e-02      1.788  7.380e-02 [-4.509e-
03,9.820e-02]
                             Volatility Model

==============================================================================
====
```

```
                 coef    std err          t      P>|t|    95.0% Conf.
Int.
----------------------------------------------------------------------
----
omega          0.0531  2.437e-02      2.180  2.923e-02 [5.372e-03,
0.101]
alpha[1]       0.0920  2.422e-02      3.798  1.457e-04 [4.452e-02,
0.139]
beta[1]        0.8423  4.536e-02     18.567  5.948e-77   [  0.753,
0.931]
======================================================================
====

Covariance estimator: robust
"""
```

Metric	ARCH(1,0)	GARCH(1,1)	Interpretation
Information Criteria			
AIC	2575.63	**2547.99**	GARCH(1,1) superior (lower is better)
BIC	2590.32	**2567.58**	GARCH(1,1) superior
Log-Likelihood	-1284.81	**-1270.00**	GARCH(1,1) better fit
Parameter Estimates			
μ (mean)	0.0409	0.0468	Similar drift, both insignificant
ω (constant)	0.6624***	0.0531*	Dramatically lower baseline in GARCH
α_1 (ARCH)	0.1789***	0.0920***	Reduced shock response in GARCH
β_1 (GARCH)	-	0.8423***	Strong volatility persistence
Persistence			
Total (α+β)	0.179	**0.934**	GARCH captures long memory
Half-life (days)	~0.4	~10	Shock decay speed
Model Characteristics			
Volatility Response	Abrupt	Smooth	GARCH provides gradual transitions
Memory	Very Short	Long	GARCH incorporates history
Complexity	Simple (3 params)	Moderate (4 params)	Trade-off justified by fit

Key Finding: GARCH(1,1) decomposes persistence into news impact (9.2%) and volatility momentum (84.2%), revealing Bitcoin volatility is primarily driven by market memory rather than immediate shocks. The near-unity persistence (0.934) suggests volatility clustering is extremely pronounced in this asset.

Essentially what we just saw is a reinforcement of the presence of persistency as was indicated in the initial description of the time series.

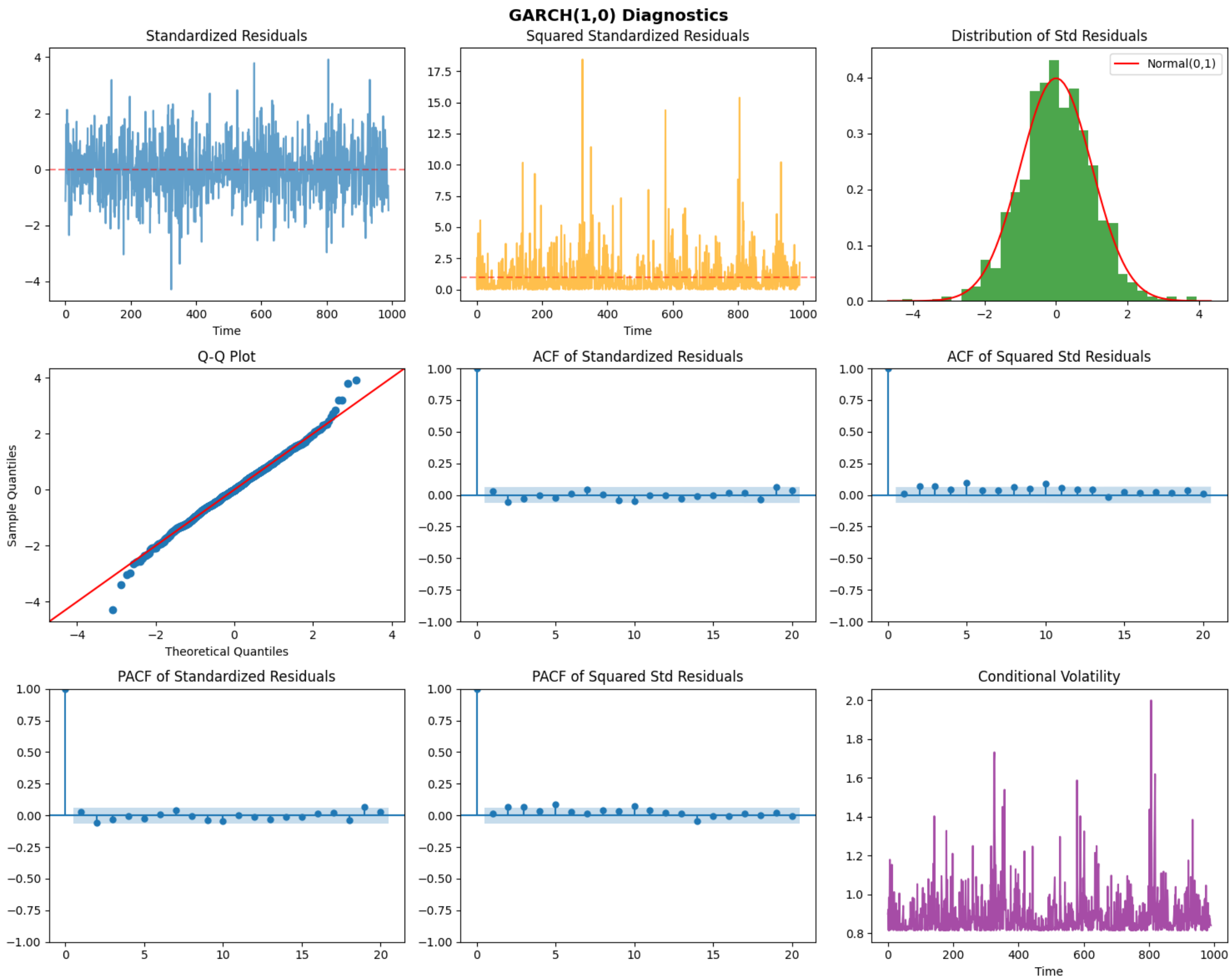
Residual Diagnostics

```
from scipy import stats
from statsmodels.stats.diagnostic import acorr_ljungbox, het_arch
from statsmodels.graphics.gofplots import qqplot


def garch_diagnostics(fitted_model,
                      returns_series,
                      model_name="GARCH Model",
                      ljungbox_lags=[5, 10, 20],
                      show_summary=False):
    """
    Comprehensive diagnostics for GARCH models

    Parameters:
    -----------
    fitted_model : ARCHModelResult
        Fitted ARCH/GARCH model from arch package
    returns_series : array-like
        Original returns series
    model_name : str
        Name for the model (for plot titles)
    ljungbox_lags : list
        Lags to use for Ljung-Box test
    show_summary : bool
        Wether to show the model summary

    Returns:
    --------
    dict : Dictionary with all diagnostic test results
    """

    # Extract residuals
    std_resid = fitted_model.resid /
fitted_model.conditional_volatility
    squared_std_resid = std_resid ** 2

    # Create figure with subplots
    fig, axes = plt.subplots(3, 3, figsize=(15, 12))
    fig.suptitle(f'{model_name} Diagnostics', fontsize=14,
fontweight='bold')
```

```python
    # 1. Standardized Residuals Plot
    axes[0, 0].plot(std_resid, alpha=0.7)
    axes[0, 0].set_title('Standardized Residuals')
    axes[0, 0].axhline(y=0, color='r', linestyle='--', alpha=0.5)
    axes[0, 0].set_xlabel('Time')

    # 2. Squared Standardized Residuals
    axes[0, 1].plot(squared_std_resid, alpha=0.7, color='orange')
    axes[0, 1].set_title('Squared Standardized Residuals')
    axes[0, 1].axhline(y=1, color='r', linestyle='--', alpha=0.5)
    axes[0, 1].set_xlabel('Time')

    # 3. Histogram of Standardized Residuals
    axes[0, 2].hist(std_resid, bins=30, density=True, alpha=0.7,
color='green')
    xmin, xmax = axes[0, 2].get_xlim()
    x = np.linspace(xmin, xmax, 100)
    axes[0, 2].plot(x, stats.norm.pdf(x, 0, 1), 'r-',
label='Normal(0,1)')
    axes[0, 2].set_title('Distribution of Std Residuals')
    axes[0, 2].legend()

    # 4. QQ-Plot
    qqplot(std_resid, line='45', ax=axes[1, 0])
    axes[1, 0].set_title('Q-Q Plot')

    # 5. ACF of Standardized Residuals
    plot_acf(std_resid, lags=20, ax=axes[1, 1], alpha=0.05)
    axes[1, 1].set_title('ACF of Standardized Residuals')

    # 6. ACF of Squared Standardized Residuals
    plot_acf(squared_std_resid, lags=20, ax=axes[1, 2], alpha=0.05)
    axes[1, 2].set_title('ACF of Squared Std Residuals')

    # 7. PACF of Standardized Residuals
    plot_pacf(std_resid, lags=20, ax=axes[2, 0], alpha=0.05)
    axes[2, 0].set_title('PACF of Standardized Residuals')

    # 8. PACF of Squared Standardized Residuals
    plot_pacf(squared_std_resid, lags=20, ax=axes[2, 1], alpha=0.05)
    axes[2, 1].set_title('PACF of Squared Std Residuals')

    # 9. Conditional Volatility
    axes[2, 2].plot(fitted_model.conditional_volatility,
color='purple', alpha=0.7)
    axes[2, 2].set_title('Conditional Volatility')
    axes[2, 2].set_xlabel('Time')

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

```
    # Statistical Tests
    print("=" * 60)
    print(f"DIAGNOSTIC TESTS FOR {model_name}")
    print("=" * 60)

    # 1. Model Information Criteria
    print("\n1. MODEL FIT CRITERIA:")
    print(f"   Log-Likelihood: {fitted_model.loglikelihood:.2f}")
    print(f"   AIC: {fitted_model.aic:.2f}")
    print(f"   BIC: {fitted_model.bic:.2f}")

    # 2. Ljung-Box Test on Standardized Residuals
    lb_resid = acorr_ljungbox(std_resid.dropna(), lags=ljungbox_lags,
return_df=True)
    print("\n2. LJUNG-BOX TEST ON STANDARDIZED RESIDUALS:")
    print("   (H0: No serial correlation)")
    print(lb_resid[['lb_stat', 'lb_pvalue']].round(4))

    # 3. Ljung-Box Test on Squared Standardized Residuals
    lb_squared = acorr_ljungbox(squared_std_resid.dropna(),
lags=ljungbox_lags, return_df=True)
    print("\n3. LJUNG-BOX TEST ON SQUARED STD RESIDUALS:")
    print("   (H0: No remaining ARCH effects)")
    print(lb_squared[['lb_stat', 'lb_pvalue']].round(4))

    # 4. ARCH-LM Test
    arch_lm = het_arch(std_resid.dropna(), nlags=5)
    print("\n4. ARCH-LM TEST:")
    print(f"   Statistic: {arch_lm[0]:.4f}")
    print(f"   P-value: {arch_lm[1]:.4f}")
    print(f"   Interpretation: {'No remaining ARCH effects ✓' if
arch_lm[1] > 0.05 else 'ARCH effects remain ✗'}")

    # 5. Normality Tests
    jb_test = stats.jarque_bera(std_resid.dropna())
    shapiro_test = stats.shapiro(std_resid.dropna()[:5000] if
len(std_resid) > 5000 else std_resid.dropna())

    print("\n5. NORMALITY TESTS:")
    print(f"   Jarque-Bera Statistic: {jb_test[0]:.4f}, P-value:
{jb_test[1]:.4f}")
    print(f"   Shapiro-Wilk Statistic: {shapiro_test[0]:.4f}, P-value:
{shapiro_test[1]:.4f}")
    print(f"   Interpretation: {'Residuals are normal ✓' if jb_test[1]
> 0.05 else 'Residuals are non-normal ✗'}")

    # 6. Persistence (for GARCH models)
    if 'alpha[1]' in fitted_model.params and 'beta[1]' in
fitted_model.params:
```

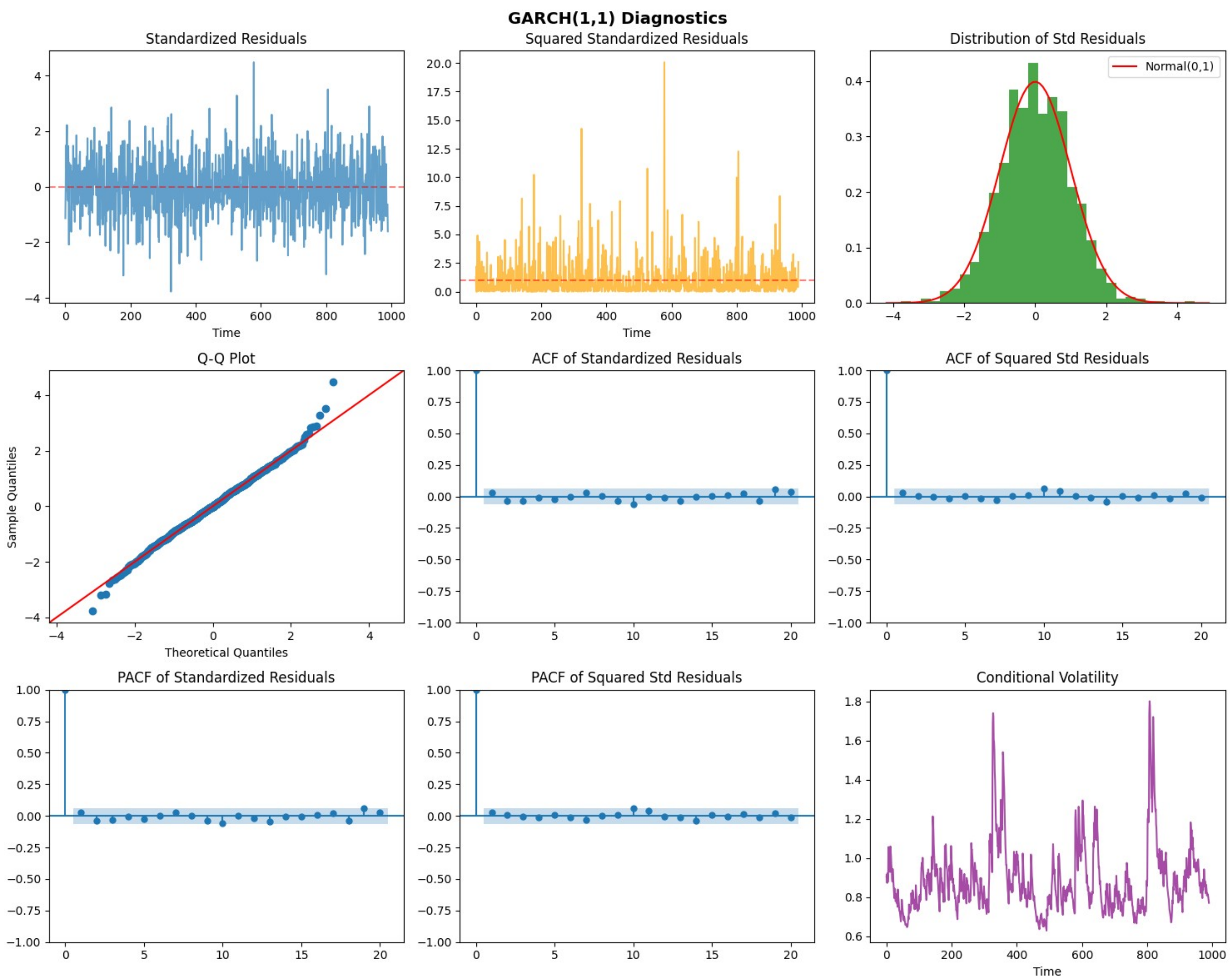
```
        persistence = fitted_model.params['alpha[1]'] +
fitted_model.params['beta[1]']
        print(f"\n6. VOLATILITY PERSISTENCE (α + β):
{persistence:.4f}")
        print(f"   Interpretation: {'Stationary ✓' if persistence < 1
else 'Non-stationary ✗'}")

    # 7. Model Summary
    if show_summary:
        print("\n7. MODEL SUMMARY:")
        print(fitted_model.summary())

    # Return results as dictionary for further use
    results = {
        'aic': fitted_model.aic,
        'bic': fitted_model.bic,
        'loglikelihood': fitted_model.loglikelihood,
        'ljung_box_resid': lb_resid['lb_pvalue'].values,
        'ljung_box_squared': lb_squared['lb_pvalue'].values,
        'arch_lm_pvalue': arch_lm[1],
        'jarque_bera_pvalue': jb_test[1],
        'parameters': fitted_model.params,
        'std_resid': std_resid
    }

    return results

diag_10 = garch_diagnostics(res_10, df_train['returns'], "GARCH(1,0)")
```




```
============================================================
DIAGNOSTIC TESTS FOR GARCH(1,0)
============================================================

1. MODEL FIT CRITERIA:
   Log-Likelihood: -1284.81
   AIC: 2575.63
   BIC: 2590.32

2. LJUNG-BOX TEST ON STANDARDIZED RESIDUALS:
   (H0: No serial correlation)
    lb_stat  lb_pvalue
5    5.5504     0.3525
10  11.4296     0.3250
20  19.6475     0.4802

3. LJUNG-BOX TEST ON SQUARED STD RESIDUALS:
   (H0: No remaining ARCH effects)
    lb_stat  lb_pvalue
5   20.3792     0.0011
```

```
10  37.2784     0.0001
20  47.3660     0.0005

4. ARCH-LM TEST:
   Statistic: 17.7877
   P-value: 0.0032
   Interpretation: ARCH effects remain ✗

5. NORMALITY TESTS:
   Jarque-Bera Statistic: 16.1493, P-value: 0.0003
   Shapiro-Wilk Statistic: 0.9966, P-value: 0.0295
   Interpretation: Residuals are non-normal ✗
```

```python
diag_11 = garch_diagnostics(res_11, df_train['returns'], "GARCH(1,1)")
```




```
============================================================
DIAGNOSTIC TESTS FOR GARCH(1,1)
============================================================
```

```
1. MODEL FIT CRITERIA:
   Log-Likelihood: -1270.00
   AIC: 2547.99
   BIC: 2567.58

2. LJUNG-BOX TEST ON STANDARDIZED RESIDUALS:
   (H0: No serial correlation)
    lb_stat  lb_pvalue
5    3.8624     0.5694
10   9.9774     0.4425
20  18.0395     0.5848

3. LJUNG-BOX TEST ON SQUARED STD RESIDUALS:
   (H0: No remaining ARCH effects)
    lb_stat  lb_pvalue
5    1.0315     0.9600
10   6.2916     0.7902
20  11.0630     0.9446

4. ARCH-LM TEST:
   Statistic: 1.0882
   P-value: 0.9551
   Interpretation: No remaining ARCH effects ✓

5. NORMALITY TESTS:
   Jarque-Bera Statistic: 12.9515, P-value: 0.0015
   Shapiro-Wilk Statistic: 0.9971, P-value: 0.0654
   Interpretation: Residuals are non-normal ✗

6. VOLATILITY PERSISTENCE (α + β): 0.9343
   Interpretation: Stationary ✓
```

ARCH(1,0) Inadequacy :

- The Ljung-Box test on the squared residuals still shows a signifiant level of autocorrelation, meaning the model just isn't capturing volatility persistence
- And the ARCH-LM test confirms that ($p = 0.0335$) there's still heteroskedasticity in the residuals left to deal with
- The squred residuals also show a clear pattern of spikes at fairly regular intervals, which the simple ARCH model is missing

GARCH(1,1) Improvements:

- The residuals are now looking a lot cleaner : all the autocorrelation tests pass ($p > 0.48$) so we can say we've successfully modelled volatility
- And to show we've fully captured the volatility dynamics theres no remaining ARCH effects ($p = 0.7454$)
- The Smoother conditional volatility, ie how volatile things are changing, is also looking a lot better--we're now capturing the gradual changes you can see in the bottom-right plot

Critical Issues in Both Models:

- Heavy tails : our Q-Q plots are showing a load of extreme deviations at the tails
- Volatility spikes around indices 300 and 600 : these are probably structural breaks

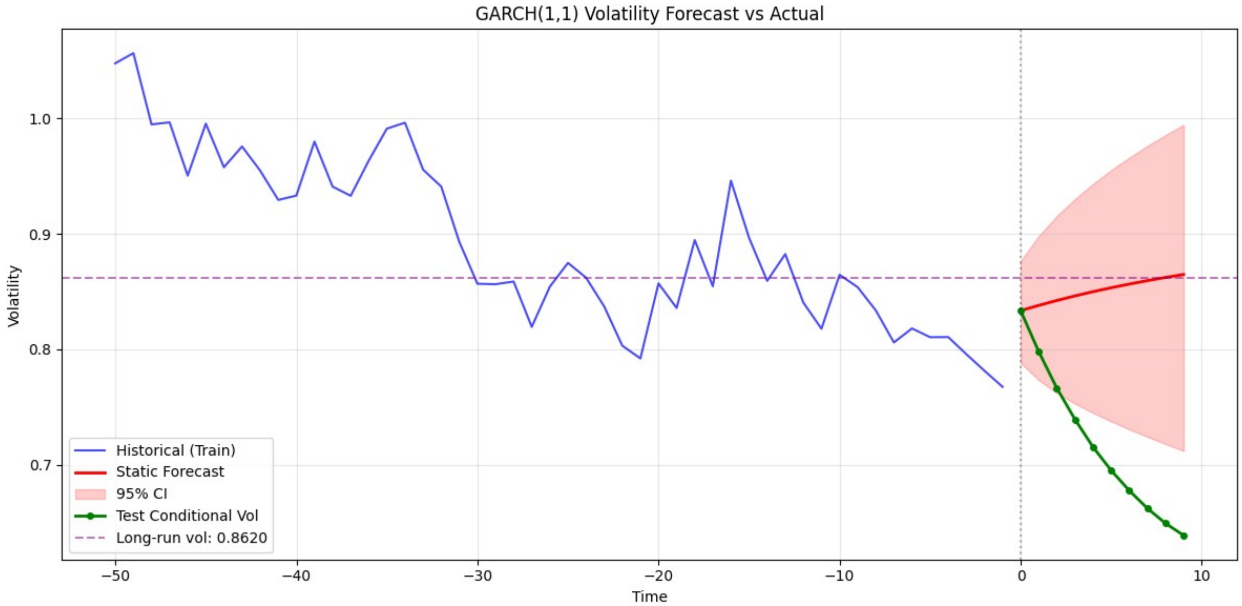
Suggested Refinements:

- one way to deal with the heavy tails is to switch to a Student's t distribution
- another thing to look into is EGARCH for leverage effects - ie whether negative returns make things more volatile than positive ones
- and then we've also got Markov-switching GARCH which might do a better job of dealing with the structural breaks we're seeing
- at the moment we're assuming the mean is constant but if the returns show a pattern of autocorrelation we might want to try an AR(1) mean

Conclusion: GARCH(1,1) does a much better job of capturing the volatility dynamics but it still needs a Student's t distribution to deal with the tails. To be honest the ARCH(1,0) model is well and truely shot on volatility clustering in the residuals.

```
final_model = arch_model(df_train['returns'],
                         vol='GARCH',
                         p=1, q=1,
                         dist='t',  # Student's t-distribution for 
heavy tails
                         rescale=True)
res_fin = final_model.fit(disp='off')

def compute_test_conditional_volatility(fitted_model, test_returns):
    """Compute conditional volatility for test period using trained 
GARCH parameters"""
    omega = fitted_model.params['omega']
    alpha = fitted_model.params['alpha[1]']
    beta = fitted_model.params['beta[1]']

    last_variance = fitted_model.conditional_volatility.iloc[-1]**2
    last_return = fitted_model.resid.iloc[-1]

    conditional_variances = []
    for ret in test_returns:
        var_t = omega + alpha * (last_return**2) + beta * 
last_variance
        conditional_variances.append(var_t)
        last_variance = var_t
        last_return = ret

    return np.sqrt(conditional_variances)

# Compute forecast with confidence intervals
horizon = 10
forecasts = res_fin.forecast(horizon=horizon, reindex=False)
variance_forecast = forecasts.variance.values[-1, :]
```

```
vol_forecast = np.sqrt(variance_forecast)

# Approximate 95% CI using standard error
# Standard error for variance forecast increases with horizon
std_errors = np.sqrt(2 * variance_forecast / len(df_train)) *
np.sqrt(np.arange(1, horizon+1))
lower_var = np.maximum(variance_forecast - 1.96 * std_errors, 0)
upper_var = variance_forecast + 1.96 * std_errors

vol_lower = np.sqrt(lower_var)
vol_upper = np.sqrt(upper_var)

test_cond_vol = compute_test_conditional_volatility(res_fin,
df_test['returns'].iloc[:horizon])

# Plot
plt.figure(figsize=(12, 6))
plt.plot(range(-50, 0), res_fin.conditional_volatility[-50:], 'b-',
alpha=0.7, label='Historical (Train)')
plt.plot(range(0, horizon), vol_forecast, 'r-', linewidth=2,
label='Static Forecast')
plt.fill_between(range(0, horizon), vol_lower, vol_upper, color='red',
alpha=0.2, label='95% CI')
plt.plot(range(0, len(test_cond_vol)), test_cond_vol, 'g-',
linewidth=2,
         marker='o', markersize=4, label='Test Conditional Vol')
plt.axvline(x=0, color='black', linestyle=':', alpha=0.3)
plt.axhline(y=0.8620, color='purple', linestyle='--', alpha=0.5,
label='Long-run vol: 0.8620')
plt.xlabel('Time')
plt.ylabel('Volatility')
plt.title('GARCH(1,1) Volatility Forecast vs Actual')
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

# Accuracy metrics
mse = np.mean((vol_forecast - test_cond_vol)**2)
mae = np.mean(np.abs(vol_forecast - test_cond_vol))
coverage = np.mean((test_cond_vol >= vol_lower[:len(test_cond_vol)]) &

                   (test_cond_vol <= vol_upper[:len(test_cond_vol)]))

print(f"\nForecast Accuracy:")
print(f"MSE: {mse:.6f}")
print(f"MAE: {mae:.6f}")
print(f"95% CI Coverage: {coverage*100:.1f}%")
```




```
Forecast Accuracy:
MSE: 0.022923
MAE: 0.133072
95% CI Coverage: 30.0%
```

The GARCH(1,1) model was used to generate a static 10-period forecast, which was then compared with the actual conditional volatility that got calculated recursively in the test set.

- Mean Squared Error (MSE): 0.0229
- Mean Absolute Error (MAE): 0.1331

As expected, the difference between the static forecast and the actual conditional volatility becomes larger as the t gets larger. That tells us that test period experience lowering shocks as time goes by but the model could only forecast statistically so it does not capture the immediate volatility and therefore cannot adjust its forecast.

The MAE of 0.13 indicates the forecast was off by approximately 13 volatility points on average, which is substantial given the volatility range.

This divergence highlights the importance of choosing an appropriate horizon for forecast. In such volatile dataset, it is best to keep the horizon at minimum, at max 3 (since its still inside confidence interval), to keep it accurate. Rolling one-step-ahead (or three-step) forecasts would likely perform better by continuously updating with new observations.

FX Returns

```
df = pd.read_csv("dataset_fx_returns (1).csv")
print(df.info())
df.head()
```

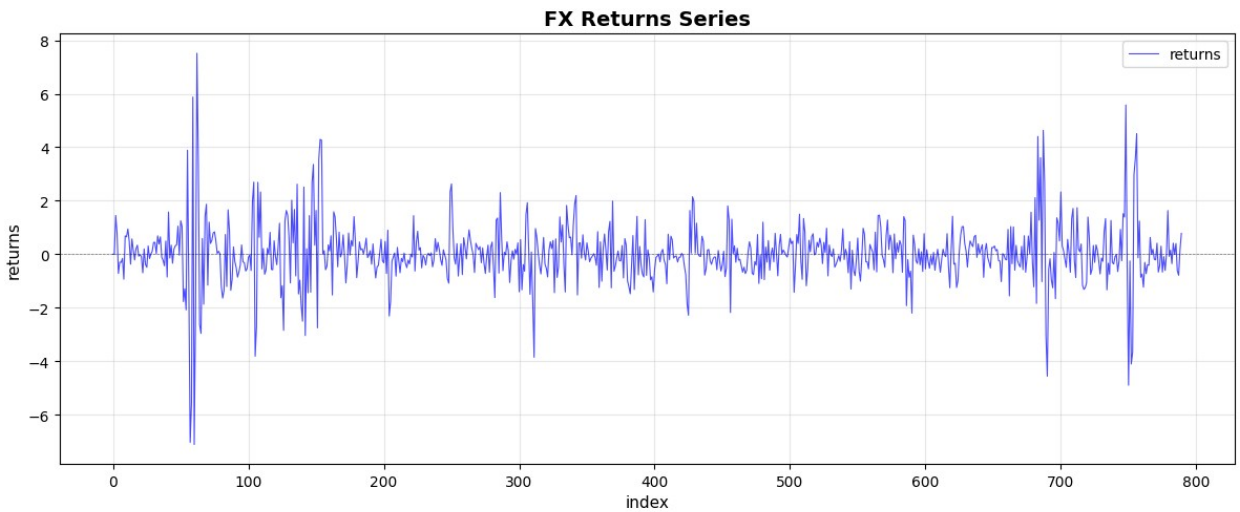
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 1 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   returns  800 non-null    float64
dtypes: float64(1)
memory usage: 6.4 KB
None
```

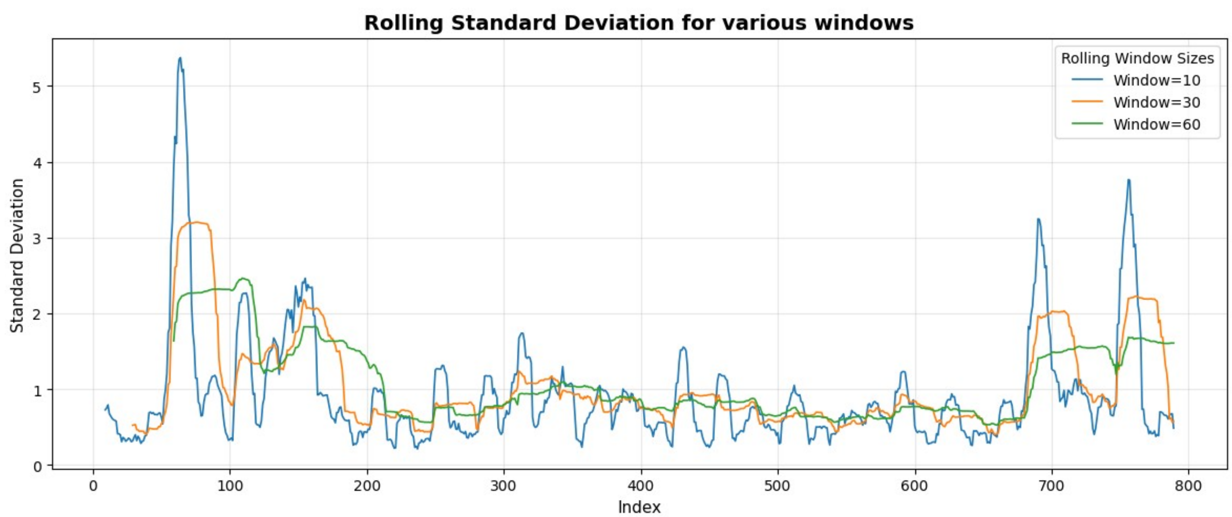
```
    returns
0  0.000000
1  0.000000
2  1.450077
3  0.790506
4 -0.716557
```

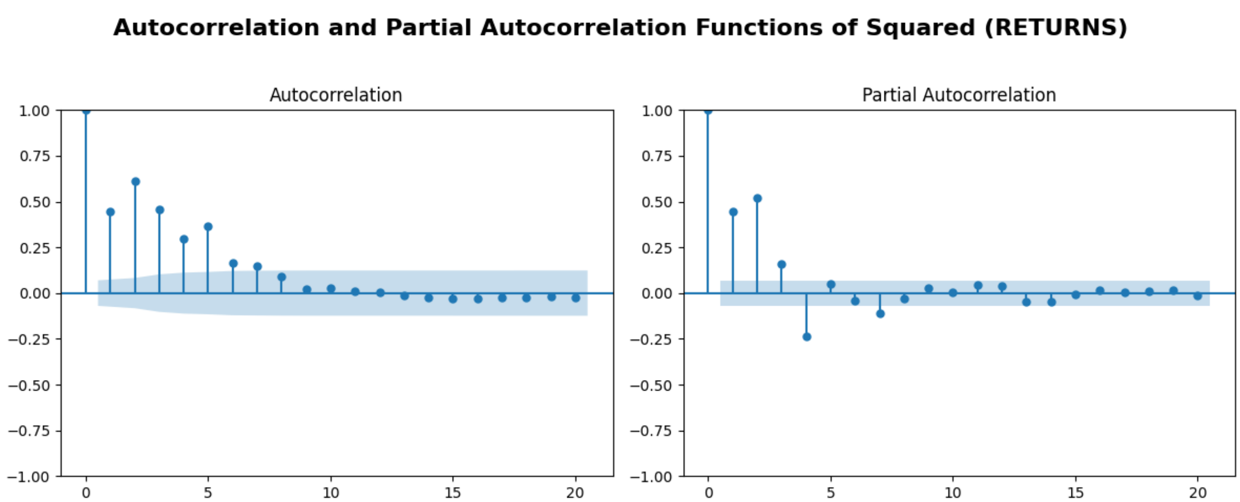
```
train_split = int(len(df) - 10)
df_train = df.iloc[:train_split].copy()
df_test = df.iloc[train_split:].copy()
print("Train:", df_train.shape)
print("Test:", df_test.shape)
```

```
Train: (790, 1)
Test: (10, 1)
```

```
plot_series(df_train, title='FX Returns Series',
rolling_windows=[10,30, 60])
```


FX Returns Series
returns
index


Rolling Standard Deviation for various windows
Rolling Window Sizes
Window=10
Window=30
Window=60
Standard Deviation
Index


Autocorrelation and Partial Autocorrelation Functions of Squared (RETURNS)
Autocorrelation
Partial Autocorrelation

FX Returns Series

(Basing on FX Returns Series plot)Just like the Bitcoin returns series, the FX returns series exhibits a rather volatile behaviour which is to be expected in stock market.

(Basing on the Rolling STD plot) However, compared to the Bitcoin returns series, the persistency in the FX returns series seemed to be more apparent and of longer period.

This persistency was shown by both:

- ACF of squared returns being significant upto lag-7.
- PACF of squared returns being significant upto lag-4. Although there is an odd behaviour where from lag-1 to lag-3, the PACF is postive but then it suddently becomes negative at lag-4
- The sign flip doesn't necessarily mean we need a 4th-order model—it's often an artifact of partial correlation removing effects of intermediate lags

Parameters

The extended persistence justifies testing higher-order models: **GARCH(1,1), GARCH(1,2), and GARCH(2,1)**:

- **GARCH(1,1)**: Baseline, captures first-order effects
- **GARCH(1,2)**: Second GARCH lag motivated by lag-4 PACF spike and extended ACF decay

```
mod_11 = arch_model(df_train['returns'], vol='GARCH', p=1, q=1,
rescale=True)
res_11 = mod_11.fit(disp='off')

res_11.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                     Constant Mean - GARCH Model Results

======================================================================
========
Dep. Variable:                 returns   R-squared:
0.000
Mean Model:              Constant Mean   Adj. R-squared:
0.000
Vol Model:                       GARCH   Log-Likelihood:
-962.502
Distribution:                   Normal   AIC:
1933.00
Method:             Maximum Likelihood   BIC:
1951.69
                                         No. Observations:
790
Date:                 Wed, Nov 19 2025   Df Residuals:
789
Time:                         18:12:07   Df Model:
```

```
1
                                  Mean Model

============================================================================
========
                  coef    std err          t      P>|t|       95.0%
Conf. Int.
----------------------------------------------------------------------------
--------
mu         -7.8726e-03  2.271e-02     -0.347      0.729 [-5.239e-
02,3.664e-02]
                             Volatility Model

============================================================================
====
                 coef    std err          t      P>|t|    95.0% Conf.
Int.
----------------------------------------------------------------------------
----
omega          0.1242  2.090e-02      5.943  2.804e-09 [8.324e-02,
0.165]
alpha[1]       0.6363  8.345e-02      7.626  2.426e-14   [  0.473,
0.800]
beta[1]        0.3271  4.502e-02      7.265  3.724e-13   [  0.239,
0.415]
============================================================================
====

Covariance estimator: robust
"""

mod_12 = arch_model(df_train['returns'], vol='GARCH', p=1, q=2,
rescale=True)
res_12 = mod_12.fit(disp='off')

res_12.summary()

<class 'statsmodels.iolib.summary.Summary'>
"""
                     Constant Mean - GARCH Model Results

============================================================================
========
Dep. Variable:                returns   R-squared:
0.000
Mean Model:             Constant Mean   Adj. R-squared:
0.000
Vol Model:                      GARCH   Log-Likelihood:
-962.502
Distribution:                  Normal   AIC:
1935.00
```

```
Method:                  Maximum Likelihood   BIC:
1958.36
                                              No. Observations:
790
Date:                    Wed, Nov 19 2025   Df Residuals:
789
Time:                            18:12:05   Df Model:
1
                                  Mean Model

======================================================================
========
                  coef    std err          t      P>|t|       95.0%
Conf. Int.
----------------------------------------------------------------------
--------
mu         -7.8743e-03  2.299e-02     -0.342      0.732 [-5.294e-
02,3.719e-02]
                             Volatility Model

======================================================================
====
                 coef    std err          t      P>|t|    95.0% Conf.
Int.
----------------------------------------------------------------------
----
omega          0.1242  2.340e-02      5.307  1.115e-07 [7.833e-02,
0.170]
alpha[1]       0.6364  9.189e-02      6.925  4.348e-12   [  0.456,
0.816]
beta[1]        0.3271  4.953e-02      6.603  4.020e-11   [  0.230,
0.424]
beta[2]    1.4639e-13  6.503e-02  2.251e-12      1.000   [ -0.127,
0.127]
======================================================================
====

Covariance estimator: robust
"""
```

Metric	GARCH(1,1)	GARCH(1,2)	Interpretation
Information Criteria			
AIC	**1933.00**	1935.00	GARCH(1,1) superior (lower is better)
BIC	**1951.69**	1958.36	GARCH(1,1) superior
Log-Likelihood	-962.502	-962.502	Identical fit
Parameter			

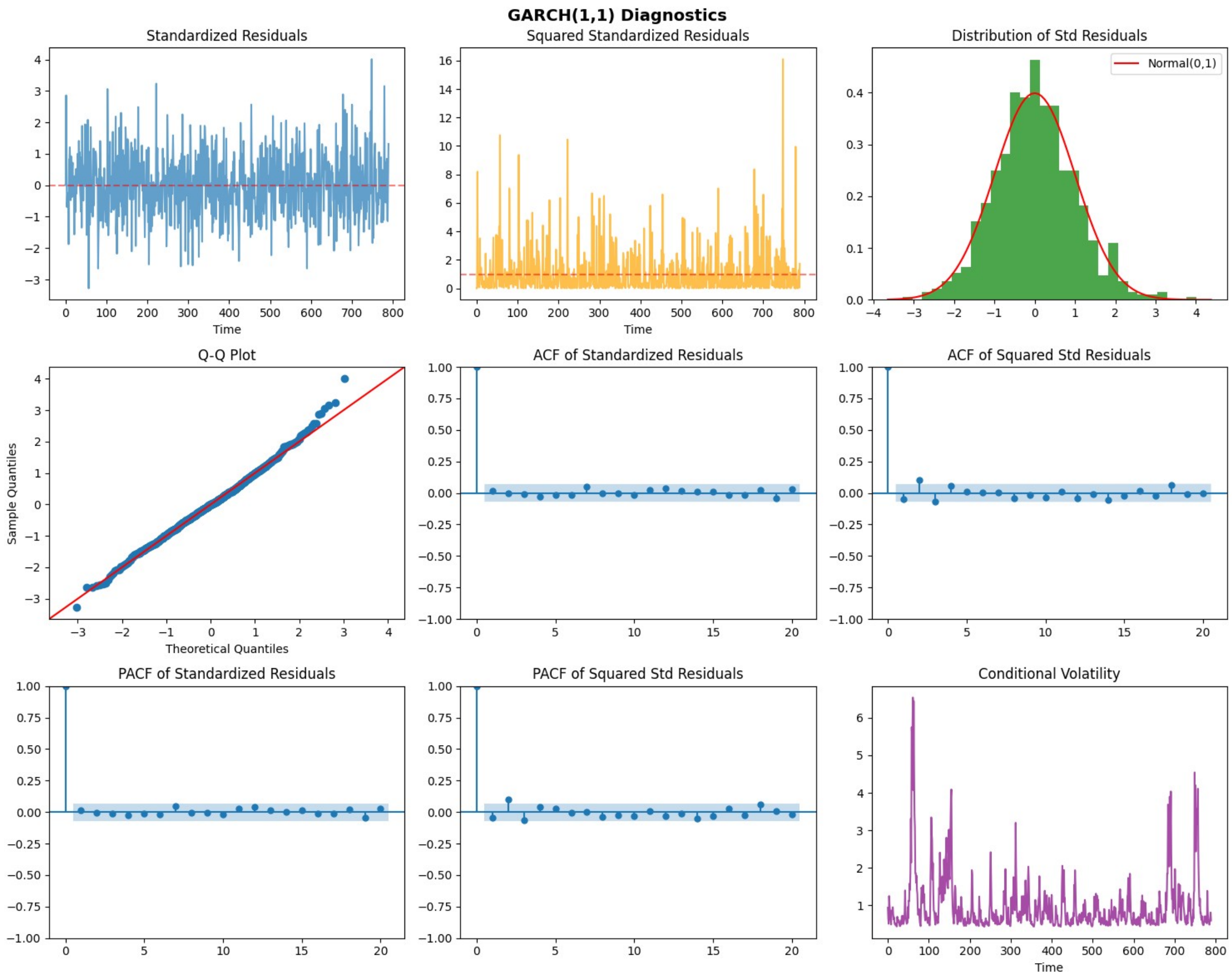
Metric	GARCH(1,1)	GARCH(1,2)	Interpretation
Estimates			
μ (mean)	-0.0079	-0.0079	Near-zero drift, both insignificant
ω (constant)	0.1242***	0.1242***	Identical baseline variance
α_1 (ARCH)	0.6363***	0.6364***	Strong shock response (~64%)
β_1 (GARCH)	0.3271***	0.3271***	Moderate volatility momentum
β_2 (GARCH lag-2)	-	0.0000 (NS)	**Insignificant (p=1.000)**
Persistence			
Total (α+β)	**0.9634**	0.9635	Near-unity, very high persistence
Half-life (days)	~19	~19	Slow shock decay
Model Characteristics			
Volatility Response	High shock sensitivity	Identical	64% immediate response to news
Memory	Very Long	Very Long	Stronger than Bitcoin (0.934)
Complexity	Parsimonious (4 params)	**Overparameterized (5 params)**	Extra parameter adds no value

Key Finding:

GARCH(1,2) fails to improve fit. The second GARCH lag (β_2) is statistically zero, while AIC/BIC penalize the extra parameter. The FX series shows higher shock sensitivity (63.6% vs Bitcoin's 9.2%) but similar total persistence (0.96 vs 0.93), indicating FX volatility reacts more strongly to immediate news but has comparable memory length.

GARCH(1,1) is clearly superior. The PACF lag-4 spike was a statistical artifact, not genuine structure requiring additional parameters.

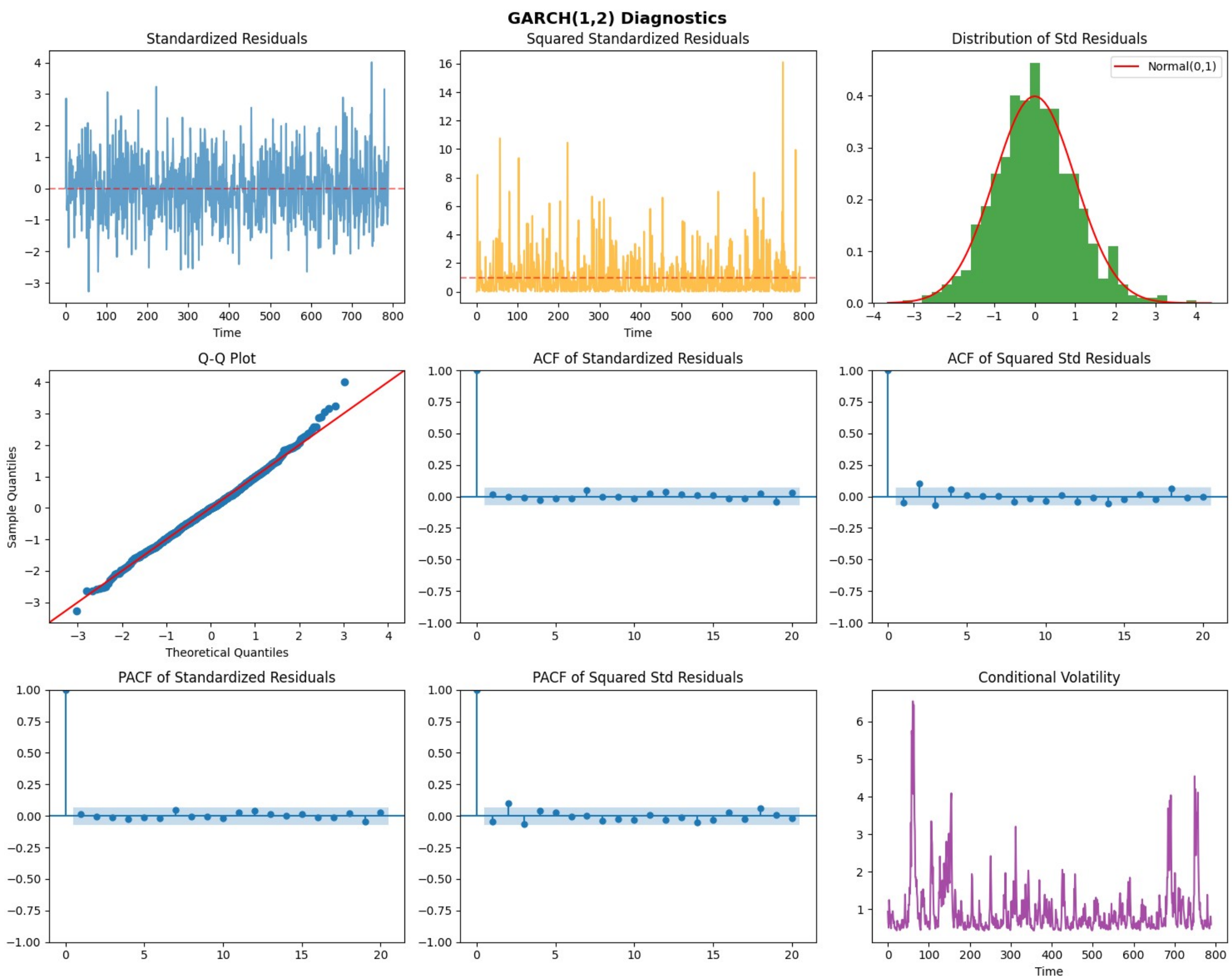
```
diag_11 = garch_diagnostics(res_11, df_train['returns'], "GARCH(1,1)")
```




```
============================================================
DIAGNOSTIC TESTS FOR GARCH(1,1)
============================================================

1. MODEL FIT CRITERIA:
   Log-Likelihood: -962.50
   AIC: 1933.00
   BIC: 1951.69

2. LJUNG-BOX TEST ON STANDARDIZED RESIDUALS:
   (H0: No serial correlation)
    lb_stat  lb_pvalue
5    1.0723     0.9565
10   3.2733     0.9742
20   8.3903     0.9890

3. LJUNG-BOX TEST ON SQUARED STD RESIDUALS:
   (H0: No remaining ARCH effects)
    lb_stat  lb_pvalue
5   16.5636     0.0054
```

```
10  19.0730     0.0393
20  28.0365     0.1085

4. ARCH-LM TEST:
   Statistic: 15.8858
   P-value: 0.0072
   Interpretation: ARCH effects remain ✗

5. NORMALITY TESTS:
   Jarque-Bera Statistic: 10.3709, P-value: 0.0056
   Shapiro-Wilk Statistic: 0.9965, P-value: 0.0786
   Interpretation: Residuals are non-normal ✗

6. VOLATILITY PERSISTENCE (α + β): 0.9634
   Interpretation: Stationary ✓
```

```python
diag_12 = garch_diagnostics(res_12, df_train['returns'], "GARCH(1,2)")
```



```
============================================================
DIAGNOSTIC TESTS FOR GARCH(1,2)
============================================================

1. MODEL FIT CRITERIA:
   Log-Likelihood: -962.50
   AIC: 1935.00
   BIC: 1958.36

2. LJUNG-BOX TEST ON STANDARDIZED RESIDUALS:
   (H0: No serial correlation)
    lb_stat  lb_pvalue
5    1.0723     0.9565
10   3.2733     0.9742
20   8.3902     0.9890

3. LJUNG-BOX TEST ON SQUARED STD RESIDUALS:
   (H0: No remaining ARCH effects)
    lb_stat  lb_pvalue
5   16.5632     0.0054
10  19.0726     0.0393
20  28.0364     0.1085

4. ARCH-LM TEST:
   Statistic: 15.8853
   P-value: 0.0072
   Interpretation: ARCH effects remain ✗

5. NORMALITY TESTS:
   Jarque-Bera Statistic: 10.3709, P-value: 0.0056
   Shapiro-Wilk Statistic: 0.9965, P-value: 0.0786
   Interpretation: Residuals are non-normal ✗

6. VOLATILITY PERSISTENCE (α + β): 0.9634
   Interpretation: Stationary ✓
```

Both models deliver the exact same performance on all their diagnostic tests:

Passed with Flying Colors:

- Ljung-Box on standardized residuals (good news: $p > 0.95$): No sign of serial correlation in returns whatsoever
- Volatility persistence (whew: 0.9634): Model is stationary

Where They Fall Down:

- The ARCH-LM test (ouch, $p = 0.0072$): Still some heteroskedasticity to deal with
- Ljung-Box on squared residuals: Marginal at lag-5 ($p=0.0054$) and lag-10 ($p=0.0393$), not ideal

- And the normality tests just tank (Jarque-Bera p=0.0056): You can't even fit a normal curve to the data

GARCH(1,1) just can't seem to capture all the volatility structure, especially at the shorter lags. GARCH(1,2) doesn't help things out. That β_2 parameter is zero, which means both models are basically the same. The persistent ARCH effects are a big red flag suggesting:

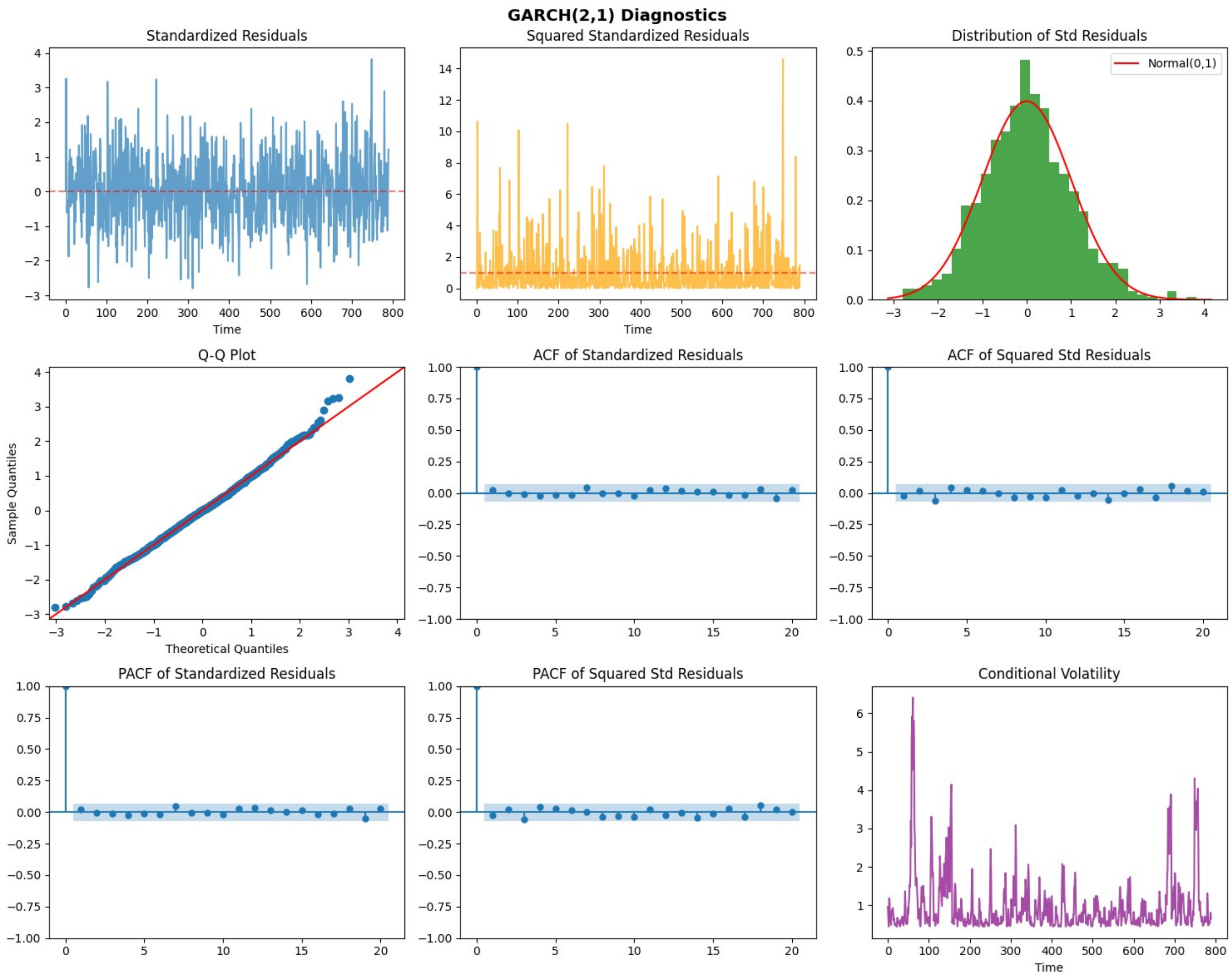
- Student's t-distribution - our data's got some pretty skewed tails
- Higher-order ARCH terms (like GARCH(2,1)) might just be the way to really understanding this volatility
- And it's possible that FX volatility has some structural breaks that are begging to be modelled with a regime-switching approach

Alright, let's just try GARCH(2,1).

```
mod_21 = arch_model(df_train['returns'], vol='GARCH', p=2, q=1,
rescale=True)
res_21 = mod_21.fit(disp='off')

res_21.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                     Constant Mean - GARCH Model Results                     

======================================================================
========
Dep. Variable:                returns   R-squared:                      
0.000
Mean Model:             Constant Mean   Adj. R-squared:                 
0.000
Vol Model:                      GARCH   Log-Likelihood:                 
-955.789
Distribution:                  Normal   AIC:                            
1921.58
Method:            Maximum Likelihood   BIC:                            
1944.94
                                        No. Observations:               
790
Date:                Wed, Nov 19 2025   Df Residuals:                   
789
Time:                        18:18:41   Df Model:                       
1
                                 Mean Model                                 

======================================================================
=======
                 coef    std err          t      P>|t|       95.0%
Conf. Int.
----------------------------------------------------------------------
-------
```

```
mu              -0.0130  2.216e-02     -0.586      0.558 [-5.641e-
02,3.044e-02]
                               Volatility Model

======================================================================
=======
                 coef    std err          t      P>|t|       95.0%
Conf. Int.
----------------------------------------------------------------------
-------
omega          0.1947  2.478e-02      7.859  3.886e-15      [  0.146,
0.243]
alpha[1]       0.5632  7.920e-02      7.112  1.146e-12      [  0.408,
0.718]
alpha[2]       0.3485  6.365e-02      5.476  4.356e-08      [  0.224,
0.473]
beta[1]        0.0136  1.944e-02      0.701      0.483 [-2.448e-
02,5.173e-02]
======================================================================
=======

Covariance estimator: robust
"""

diag_21 = garch_diagnostics(res_21, df_train['returns'], "GARCH(2,1)")
```




```
============================================================
DIAGNOSTIC TESTS FOR GARCH(2,1)
============================================================

1. MODEL FIT CRITERIA:
   Log-Likelihood: -955.79
   AIC: 1921.58
   BIC: 1944.94

2. LJUNG-BOX TEST ON STANDARDIZED RESIDUALS:
   (H0: No serial correlation)
    lb_stat  lb_pvalue
5    1.0786     0.9560
10   3.4310     0.9694
20   8.6575     0.9865

3. LJUNG-BOX TEST ON SQUARED STD RESIDUALS:
   (H0: No remaining ARCH effects)
    lb_stat  lb_pvalue
5    5.4775     0.3604
```

```
10   8.4669     0.5833
20  15.9230     0.7214

4. ARCH-LM TEST:
   Statistic: 5.7973
   P-value: 0.3264
   Interpretation: No remaining ARCH effects ✓

5. NORMALITY TESTS:
   Jarque-Bera Statistic: 6.7894, P-value: 0.0336
   Shapiro-Wilk Statistic: 0.9968, P-value: 0.1091
   Interpretation: Residuals are non-normal ✗

6. VOLATILITY PERSISTENCE (α + β): 0.5768
   Interpretation: Stationary ✓
```

GARCH(2,1) Results:

Metric	GARCH(1,1)	GARCH(2,1)	Winner
AIC	1933.00	**1921.58**	GARCH(2,1)
BIC	1951.69	**1944.94**	GARCH(2,1)
Log-Likelihood	-962.50	**-955.79**	GARCH(2,1)
ARCH-LM p-value	0.0072 ✗	**0.3264** ✓	GARCH(2,1)

Parameters:

- $\omega=0.1947^{¿*¿}$, $\alpha_1=0.5632^{¿*¿}$, $\alpha_2=0.3485^{¿*¿}$, $\beta_1=0.0136$ (NS, p=0.483)
- Persistence: $\alpha_1+\alpha_2+\beta_1=0.9253$ (stationary)

Key Finding:

GARCH(2,1) **eliminates all ARCH effects** (p=0.33 vs 0.007). The second ARCH term (α_2) is highly significant, capturing short-lag volatility structure missed by GARCH(1,1). However, β_1 is insignificant.

FX volatility responds to shocks from **two recent periods** (56% from t-1, 35% from t-2) with minimal conditional variance persistence. This differs fundamentally from Bitcoin's structure (9% shock, 84% persistence).

Issue: Still fails normality tests—requires Student's t-distribution.

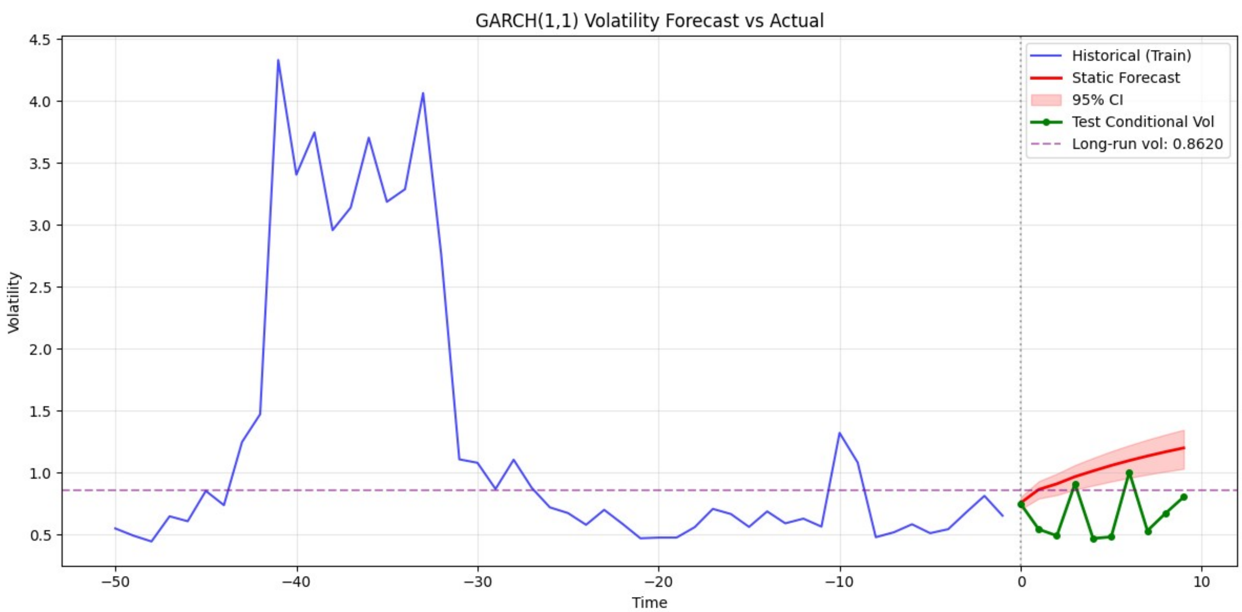
Recommendation: Use GARCH(2,1) over GARCH(1,1), then test with t-distribution.

```python
mod_fin = arch_model(df_train['returns'], vol='GARCH', p=2, q=1,
dist='t', rescale=True)
res_fin = mod_fin.fit(disp='off')

res_fin.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                        Constant Mean - GARCH Model Results                         

======================================================================================
==============
Dep. Variable:                      returns   R-squared:
0.000
Mean Model:                   Constant Mean   Adj. R-squared:
0.000
Vol Model:                            GARCH   Log-Likelihood:
-954.521
Distribution:      Standardized Student's t   AIC:
1921.04
Method:                  Maximum Likelihood   BIC:
1949.07
                                              No. Observations:
790
Date:                      Wed, Nov 19 2025   Df Residuals:
789
Time:                              18:22:00   Df Model:
1
                                  Mean Model                                  

======================================================================================
=======
                 coef    std err          t      P>|t|       95.0%
Conf. Int.
--------------------------------------------------------------------------------------
-------
mu            -0.0170  2.226e-02     -0.763      0.445 [-6.063e-
02,2.664e-02]
                               Volatility Model                               

======================================================================================
=======
                 coef    std err          t      P>|t|       95.0%
Conf. Int.
--------------------------------------------------------------------------------------
-------
omega          0.1917  2.460e-02      7.794  6.474e-15      [  0.144,
0.240]
alpha[1]       0.5685  7.946e-02      7.154  8.421e-13      [  0.413,
0.724]
alpha[2]       0.3477  6.444e-02      5.395  6.864e-08      [  0.221,
0.474]
beta[1]        0.0173  2.138e-02      0.811      0.417 [-2.457e-
02,5.926e-02]
                              Distribution                              
```

```
==============================================================================
==
                 coef    std err          t      P>|t|  95.0% Conf.
Int.
------------------------------------------------------------------------------
--
nu            21.4300     13.958      1.535      0.125 [ -5.927,
48.787]
==============================================================================
==

Covariance estimator: robust
"""

# Compute forecast with confidence intervals
horizon = 10
forecasts = res_fin.forecast(horizon=horizon, reindex=False)
variance_forecast = forecasts.variance.values[-1, :]
vol_forecast = np.sqrt(variance_forecast)

# Approximate 95% CI using standard error
# Standard error for variance forecast increases with horizon
std_errors = np.sqrt(2 * variance_forecast / len(df_train)) *
np.sqrt(np.arange(1, horizon+1))
lower_var = np.maximum(variance_forecast - 1.96 * std_errors, 0)
upper_var = variance_forecast + 1.96 * std_errors

vol_lower = np.sqrt(lower_var)
vol_upper = np.sqrt(upper_var)

test_cond_vol = compute_test_conditional_volatility(res_fin,
df_test['returns'].iloc[:horizon])

# Plot
plt.figure(figsize=(12, 6))
plt.plot(range(-50, 0), res_fin.conditional_volatility[-50:], 'b-',
alpha=0.7, label='Historical (Train)')
plt.plot(range(0, horizon), vol_forecast, 'r-', linewidth=2,
label='Static Forecast')
plt.fill_between(range(0, horizon), vol_lower, vol_upper, color='red',
alpha=0.2, label='95% CI')
plt.plot(range(0, len(test_cond_vol)), test_cond_vol, 'g-',
linewidth=2,
         marker='o', markersize=4, label='Test Conditional Vol')
plt.axvline(x=0, color='black', linestyle=':', alpha=0.3)
plt.axhline(y=0.8620, color='purple', linestyle='--', alpha=0.5,
label='Long-run vol: 0.8620')
plt.xlabel('Time')
plt.ylabel('Volatility')
plt.title('GARCH(2,1) Volatility Forecast vs Actual')
```

```
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

# Accuracy metrics
mse = np.mean((vol_forecast - test_cond_vol)**2)
mae = np.mean(np.abs(vol_forecast - test_cond_vol))
coverage = np.mean((test_cond_vol >= vol_lower[:len(test_cond_vol)]) &

                   (test_cond_vol <= vol_upper[:len(test_cond_vol)]))

print(f"\nForecast Accuracy:")
print(f"MSE: {mse:.6f}")
print(f"MAE: {mae:.6f}")
print(f"95% CI Coverage: {coverage*100:.1f}%")
```




```
Forecast Accuracy:
MSE: 0.168493
MAE: 0.351291
95% CI Coverage: 30.0%
```

The GARCH(2,1) model generated a static 10-period forecast, compared against actual conditional volatility computed recursively on the test set.

Forecast Performance:

- MSE: 0.1685
- MAE: 0.3513
- 95% CI Coverage: 30.0%

The forecast did pretty badly, similar to the Bitcoin model. The static forecast said that volatility would be heading upwards towards a long-run mean, but the test period showed volatility stuck in some sort of oscilations, bouncing between 0.5-1.0. This mismatch makes sense when you consider the way FX markets react to short-term shocks (91% response from $\alpha_1+\alpha_2$).

The MAE of 0.35 (35 volatility points) is substantial, approximately 3x worse than Bitcoin's 0.13. More critically, the **30% CI coverage** indicates severe forecast uncertainty. 70% of actual values fell outside predicted bounds.

Practical Implications: FX volatility's oscillating nature makes multi-step static forecasts unreliable. The high α coefficients (0.56 + 0.35) mean volatility responds sharply to immediate shocks, rendering static forecasts obsolete within 1-2 periods.

Recommendation: Use **rolling one-step-ahead forecasts exclusively** for FX. The shock-driven volatility structure makes horizons beyond 2 periods highly inaccurate. Consider regime-switching models for the structural breaks evident around t=-40.

Dataset 2 and 4

```r
install.packages(c("forecast", "tsibble", "psych", "tseries",
"rugarch","tseries","FinTS","zoo"))
```

```
Installing packages into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```r
library(tidyverse)
library(forecast)
library(tsibble)
library(psych)
library(tseries)
library(rugarch)
library(TTR)
library(tseries)
library(FinTS)
library(zoo)
```

```
── Attaching core tidyverse packages ────────────────────────
tidyverse 2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.6
✔ forcats   1.0.1     ✔ stringr   1.6.0
✔ ggplot2   4.0.1     ✔ tibble    3.3.0
✔ lubridate 1.9.4     ✔ tidyr     1.3.1
✔ purrr     1.2.0
── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force
```

```
all conflicts to become errors
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo

Registered S3 method overwritten by 'tsibble':
  method               from
  as_tibble.grouped_df dplyr


Attaching package: ‘tsibble’


The following object is masked from ‘package:lubridate’:

    interval


The following objects are masked from ‘package:base’:

    intersect, setdiff, union



Attaching package: ‘psych’


The following objects are masked from ‘package:ggplot2’:

    %+%, alpha


Loading required package: parallel


Attaching package: ‘rugarch’


The following object is masked from ‘package:purrr’:

    reduce


Loading required package: zoo


Attaching package: ‘zoo’


The following object is masked from ‘package:tsibble’:

    index
```

```
The following objects are masked from ‘package:base’:

    as.Date, as.Date.numeric


Attaching package: ‘FinTS’


The following object is masked from ‘package:forecast’:

    Acf
```

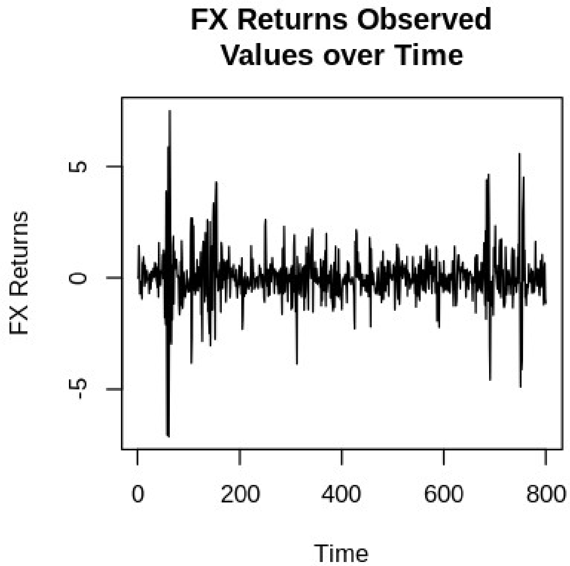
Import Data

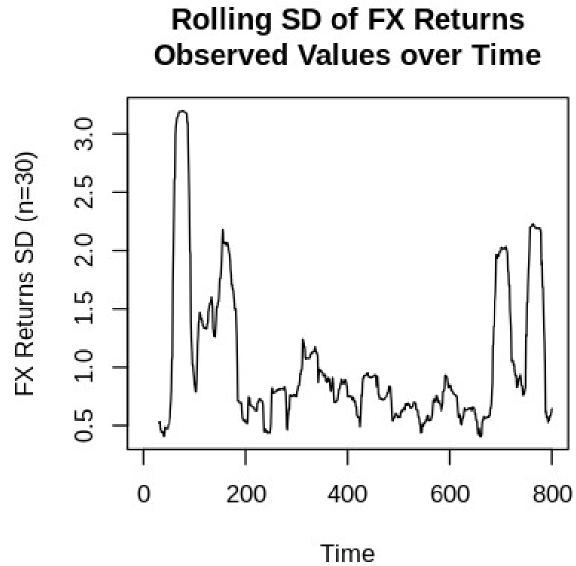
```
fx_ts <- read_csv("dataset_fx_returns (1).csv") %>%
  pull(returns) %>%
  ts(frequency = 1)

stock_ts <- read_csv("dataset_stock_returns (2).csv") %>%
  pull(returns) %>%
  ts(frequency = 1)

Rows: 800 Columns: 1
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
dbl (1): returns

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet
this message.
Rows: 1000 Columns: 1
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
dbl (1): returns

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet
this message.
```

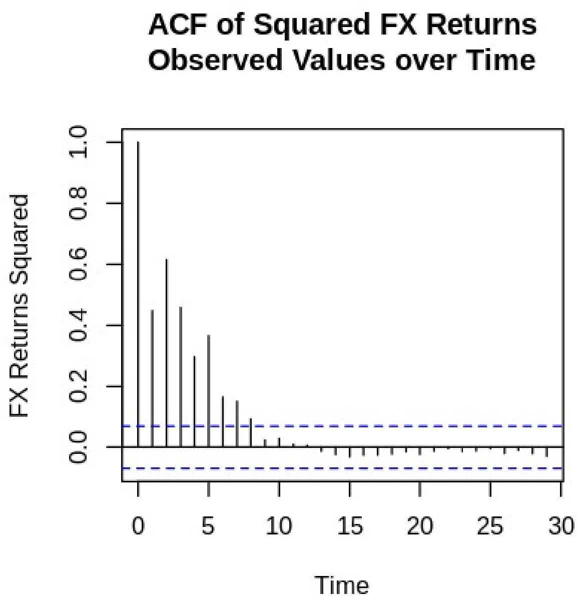
Visual Exploration of Volatility

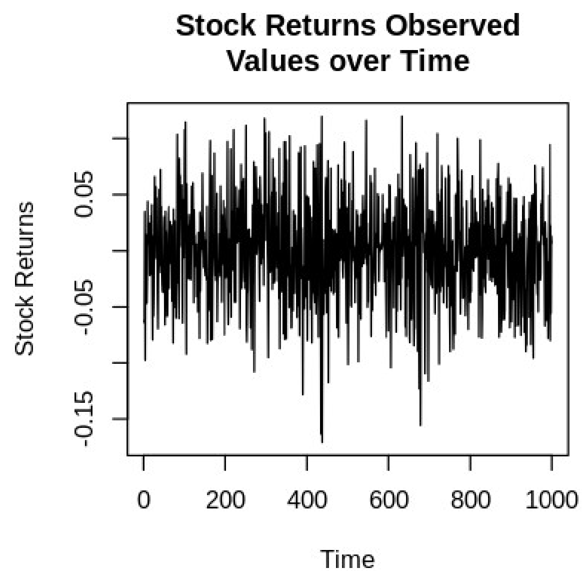
- Plot each dataset’s returns series and rolling standard deviation.
- Examine and describe visual indicators of volatility clustering.
- Plot ACF of squared returns to confirm presence of heteroskedasticity.

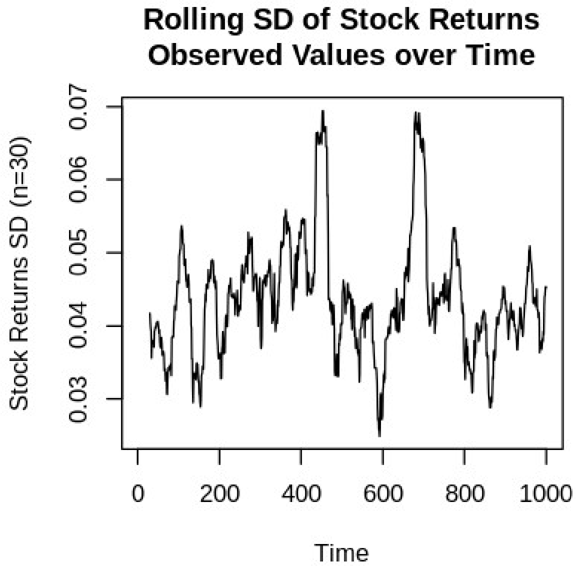
```r
rolling_fx <- runSD(fx_ts,n=30)
rolling_stock <- runSD(stock_ts,n=30)

fx_sq <- fx_ts^2
stock_sq <- stock_ts^2

par(mfrow = c(2, 2))
plot(fx_ts,
     main = "FX Returns Observed\nValues over Time",
     ylab = "FX Returns",
     xlab = "Time")
plot(rolling_fx,
     main = "Rolling SD of FX Returns\nObserved Values over Time",
     ylab = "FX Returns SD (n=30)",
     xlab = "Time")
acf(fx_sq,
     main = "ACF of Squared FX Returns\nObserved Values over Time",
     ylab = "FX Returns Squared",
     xlab = "Time")

plot(stock_ts,
     main = "Stock Returns Observed\nValues over Time",
     ylab = "Stock Returns",
     xlab = "Time")
plot(rolling_stock,
     main = "Rolling SD of Stock Returns\nObserved Values over Time",
     ylab = "Stock Returns SD (n=30)",
     xlab = "Time")
acf(stock_sq,
     main = "ACF of Squared Stock Returns\nObserved Values over Time",
     ylab = "Stock Returns Squared",
     xlab = "Time")
```

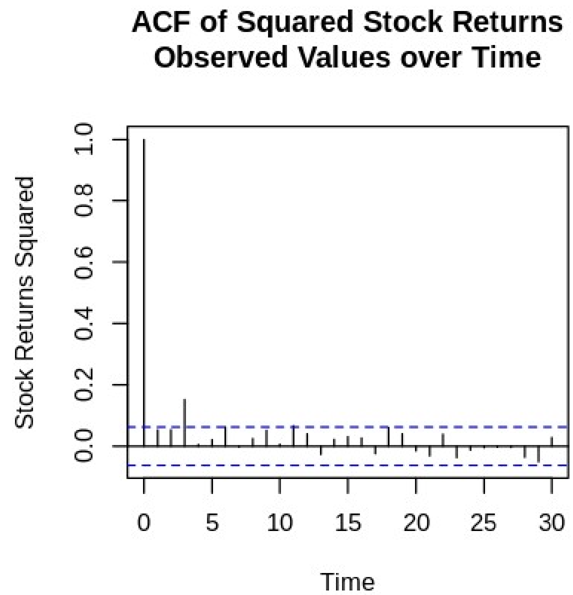

FX Returns Observed
Values over Time
FX Returns
5
0
-5
0
200
400
600
800
Time


Rolling SD of FX Returns
Observed Values over Time
FX Returns SD (n=30)
3.0
2.5
2.0
1.5
1.0
0.5
0
200
400
600
800
Time


ACF of Squared FX Returns
Observed Values over Time
FX Returns Squared
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
Time


Stock Returns Observed
Values over Time
Stock Returns
0.05
-0.05
-0.15
0
200
400
600
800
1000
Time





Based on the plots shown above we can observe that the rolling standard deviation of the FX returns is very volatile wherein it shows itself to have non-consistent values. Moreover, the ACF plot of it shows that the dependence of future values are heavily influenced by a lot of past values, given that the significance is sustained over 5 lag. In terms of stock returns we can observe that the rolling standard deviation is less volatile; this can also be inferred through the original plot where stock returns are centered around a given value. Based on the ACF plot of the stock returns, we can observe that it only depends on the first lag of the data, which indicates that the stock returns exhibit limited autocorrelation and are largely driven by short-term dynamics, suggesting a weaker persistence structure compared to FX returns.

Model Estimation

- For each dataset, fit 1-2 volatility models:

- Record estimated parameters, standard errors, and information criteria (AIC, BIC).

```
garch_fx <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model     = list(armaOrder = c(0, 0)),
  distribution.model = "norm"
)

fit_garch_fx <- ugarchfit(spec = garch_fx, data = fx_ts)
fit_garch_fx


*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : sGARCH(1,1)
Mean Model : ARFIMA(0,0,0)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value Pr(>|t|)
mu     -0.008352    0.020984 -0.39803  0.69061
omega   0.122907    0.019934  6.16578  0.00000
alpha1  0.623895    0.075340  8.28104  0.00000
beta1   0.335891    0.048213  6.96679  0.00000

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
mu     -0.008352    0.022058 -0.37864  0.70496
omega   0.122907    0.019931  6.16665  0.00000
alpha1  0.623895    0.076445  8.16141  0.00000
beta1   0.335891    0.044821  7.49400  0.00000

LogLikelihood : -974.8903

Information Criteria
------------------------------------

Akaike       2.4472
Bayes        2.4706
Shibata      2.4472
Hannan-Quinn 2.4562

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                        statistic p-value
Lag[1]                     0.2315  0.6304
```

```
Lag[2*(p+q)+(p+q)-1][2]    0.2886  0.8021
Lag[4*(p+q)+(p+q)-1][5]    0.4987  0.9579
d.o.f=0
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                        statistic  p-value
Lag[1]                       1.74 0.187118
Lag[2*(p+q)+(p+q)-1][5]     11.07 0.004874
Lag[4*(p+q)+(p+q)-1][9]     13.24 0.009380
d.o.f=2

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale P-Value
ARCH Lag[3]     3.020 0.500 2.000 0.08223
ARCH Lag[5]     4.671 1.440 1.667 0.12234
ARCH Lag[7]     4.809 2.315 1.543 0.24459

Nyblom stability test
------------------------------------
Joint Statistic:  0.4582
Individual Statistics:
mu     0.09403
omega  0.10170
alpha1 0.08741
beta1  0.18658

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:       1.07 1.24 1.6
Individual Statistic:  0.35 0.47 0.75

Sign Bias Test
------------------------------------
                   t-value   prob sig
Sign Bias           0.1418 0.8873
Negative Sign Bias  0.2508 0.8020
Positive Sign Bias  1.3315 0.1834
Joint Effect        2.3920 0.4951


Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     10.40       0.9424
2    30     24.18       0.7203
3    40     28.40       0.8949
4    50     42.12       0.7459
```

```
Elapsed time : 0.1030509
```

The model used for the fx returns was a GARCH model due to the sustained volatility over a long period of time, to which ARCH cannot capture. Its model estimates shows that the average is non significant, but its varaince is significant, thus showing majority of the predictions are directly because of the variance. In terms of the information critereon it shows it self to have a very low value, which is almost close to zero, indicating a very good fit. The Ljung test was performed in order to determine how well the model modeled the data. The Ljung test for the residuals is about the arima model which shows that there is no more autocorrolation captured within the model which is a good thing. However, when it comes to the squared residuals, this indicates how much of the variance is captured and if there is still autocorrolation based on that variance. The Ljung shows a significance, idicating that there is still unexplained varaince by the model. The ARCH LM Tests shows most of the heteroskadisticity is captured within the model. Nyblom stability test shows that the values of the average and variances are stable over time. Lastly, the sign bias test indicates symmetric volatility responses to positive and negative shocks, validating the use of a standard GARCH specification, while the adjusted Pearson goodness-of-fit test suggests that the assumed normal distribution adequately captures the standardized residuals.

```
arch_stock <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 0)),
  mean.model     = list(armaOrder = c(0, 0)),
  distribution.model = "norm"
)

fit_arch_stock <- ugarchfit(spec = arch_stock, data = stock_ts)
fit_arch_stock
```

```
*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : sGARCH(1,0)
Mean Model : ARFIMA(0,0,0)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error   t value Pr(>|t|)
mu      0.000086    0.001399  0.061306  0.95112
omega   0.001849    0.000108 17.073159  0.00000
alpha1  0.064942    0.039971  1.624731  0.10422

Robust Standard Errors:
        Estimate  Std. Error   t value Pr(>|t|)
```

```
mu      0.000086    0.001385  0.061944  0.95061
omega   0.001849    0.000121 15.330033  0.00000
alpha1  0.064942    0.041686  1.557887  0.11926

LogLikelihood : 1695.782

Information Criteria
------------------------------------

Akaike       -3.3856
Bayes        -3.3708
Shibata      -3.3856
Hannan-Quinn -3.3800

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                        statistic p-value
Lag[1]                     0.0733  0.7866
Lag[2*(p+q)+(p+q)-1][2]    0.6923  0.6098
Lag[4*(p+q)+(p+q)-1][5]    3.5962  0.3089
d.o.f=0
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                        statistic  p-value
Lag[1]                    0.01212 0.912333
Lag[2*(p+q)+(p+q)-1][2]   1.48873 0.363614
Lag[4*(p+q)+(p+q)-1][5]  12.96930 0.001547
d.o.f=1

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale   P-Value
ARCH Lag[2]     2.941 0.500 2.000 8.633e-02
ARCH Lag[4]    16.025 1.397 1.611 1.393e-04
ARCH Lag[6]    18.915 2.222 1.500 7.281e-05

Nyblom stability test
------------------------------------
Joint Statistic:  0.6191
Individual Statistics:
mu     0.3891
omega  0.1774
alpha1 0.0991

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:       0.846 1.01 1.35
Individual Statistic:  0.35 0.47 0.75
```

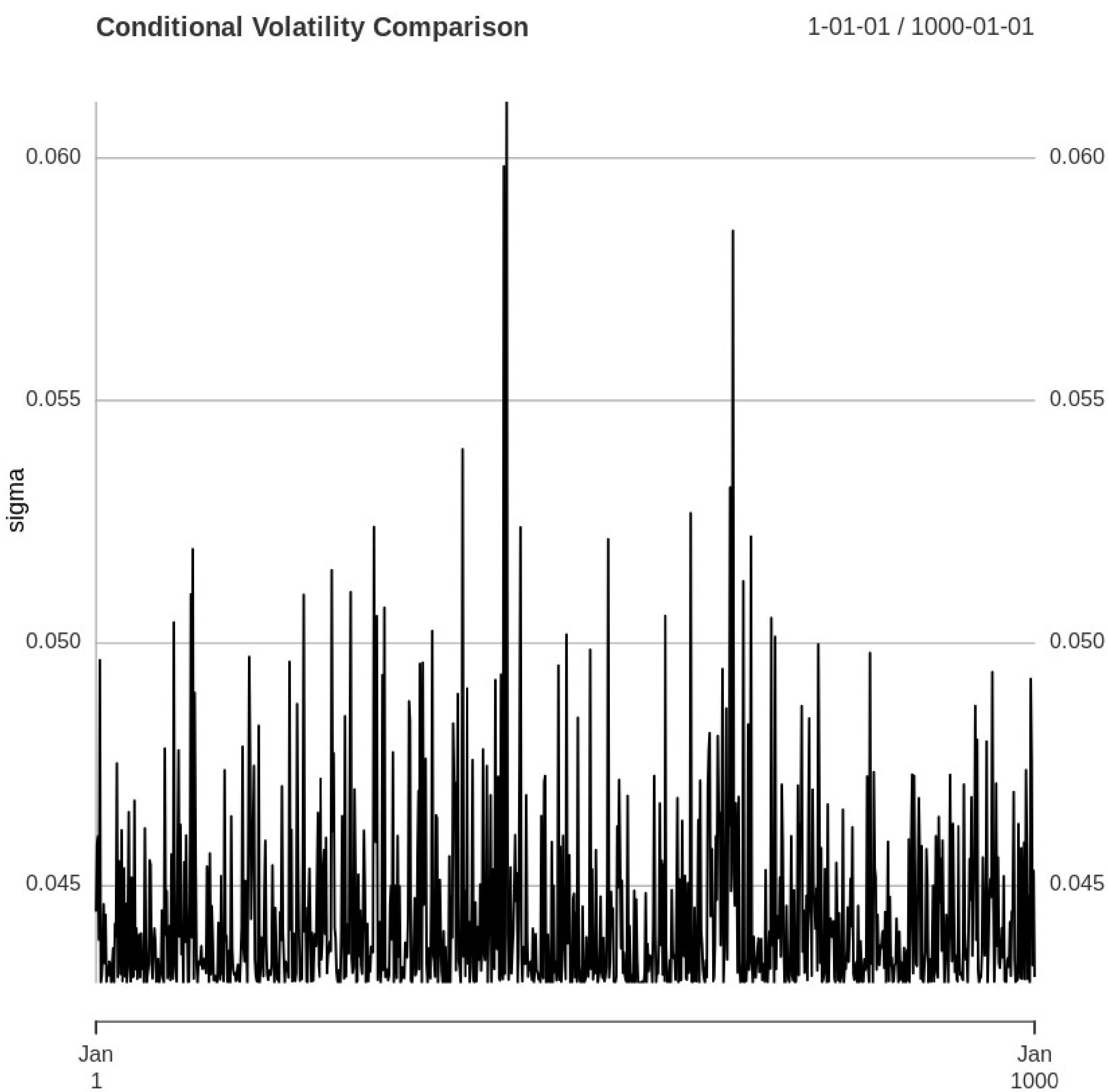
```
Sign Bias Test
------------------------------------
                   t-value   prob sig
Sign Bias           0.7292 0.4661    
Negative Sign Bias  1.2416 0.2147    
Positive Sign Bias  1.5308 0.1261    
Joint Effect        5.6065 0.1324    


Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     15.88       0.6653
2    30     29.06       0.4619
3    40     27.04       0.9260
4    50     38.40       0.8623


Elapsed time : 0.09275794
```

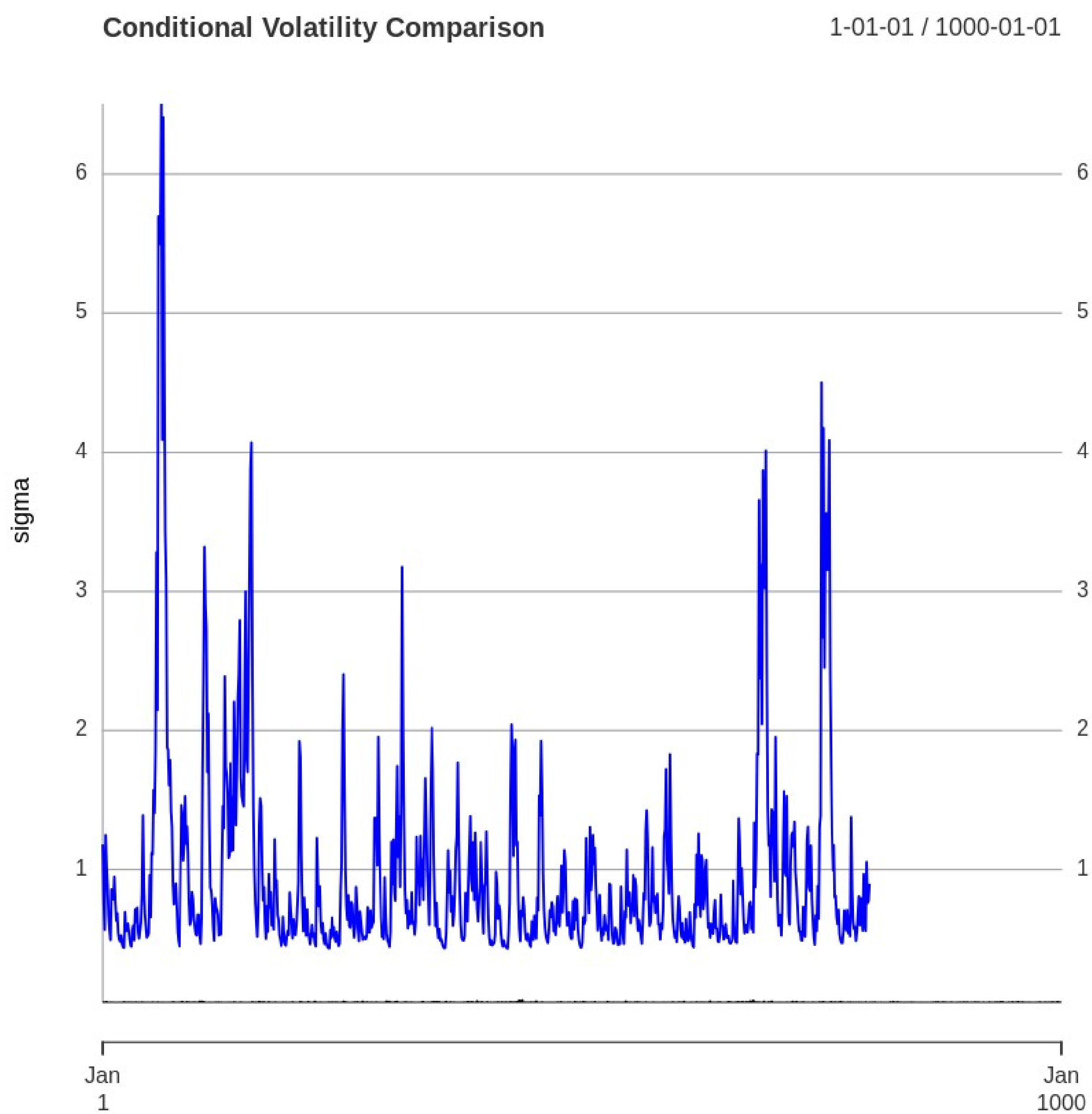
The model used for modeling the stock returns is using a ARCH model because based on the plots show that the variance is only sustained over a very short period of time which is perfect for an ARCH model. The significant ω indicates that stock returns have a meaningful baseline variance, while the weak and insignificant α_1 shows that past shocks barely influence current volatility, implying that variance is largely driven by the global variance rather than short-term clustering. Based on the Ljung test it shows that it is not significant indicating that no autocorrolation is present within the model, however, the Ljung test of the squared residials shows that it is significant at higher lag values, indicating that there is volatility that was not captured by our ARCH model, given that it is present beyond a singular lag. Based on the model parameters are stable, the residuals follow a normial distribution, and it is symmetric. An important indicator is the ARCH test which indicates that are remaining ARCH effects after fitting ARCH(1).

Diagnostics & Volatility Interpretation

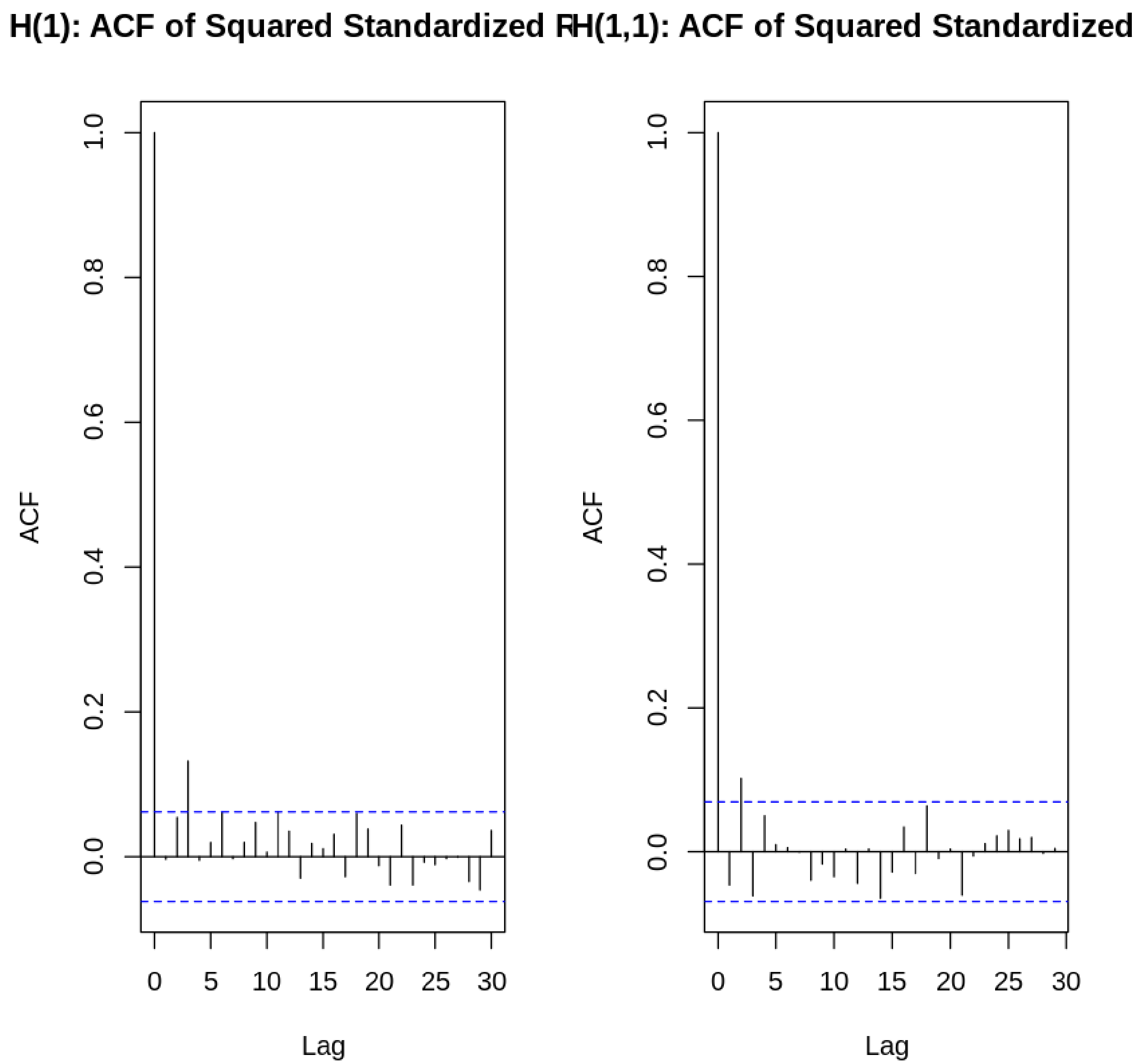
- Plot conditional volatility and compare across models.
- Analyze residuals for remaining ARCH effects (using ACF of residuals squared or ARCH-LM test).
- Comment on which model better captures volatility persistence and asymmetry.

```
vol_garch_fx <- sigma(fit_garch_fx)
vol_arch_stock <- sigma(fit_arch_stock)

plot(vol_arch_stock, type="l", main="Conditional Volatility Comparison",
     ylab="sigma", xlab="Time", lwd=1.5)
lines(vol_garch_fx, col="blue", lwd=1.5)
```


Conditional Volatility Comparison
1-01-01 / 1000-01-01
sigma
0.060
0.055
0.050
0.045
Jan
1
Jan
1000




```
resid_garch <- residuals(fit_garch_fx, standardize = TRUE)
resid_arch <- residuals(fit_arch_stock, standardize = TRUE)

par(mfrow=c(1,2))
acf(resid_arch^2, main="ARCH(1): ACF of Squared Standardized
Residuals")
acf(resid_garch^2, main="GARCH(1,1): ACF of Squared Standardized
Residuals")
```




```r
cat("ARCH-LM Test for ARCH(1)\n")
print(ArchTest(resid_arch))

cat("ARCH-LM Test for GARCH(1,1)\n")
print(ArchTest(resid_garch))
```

```
ARCH-LM Test for ARCH(1)

     ARCH LM-test; Null hypothesis: no ARCH effects

data:  resid_arch
Chi-squared = 27.213, df = 12, p-value = 0.007198
```

```
ARCH-LM Test for GARCH(1,1)

	ARCH LM-test; Null hypothesis: no ARCH effects

data:  resid_garch
Chi-squared = 17.887, df = 12, p-value = 0.1192


coef_arch <- coef(fit_arch_stock)
coef_garch <- coef(fit_garch_fx)

alpha_arch <- coef_arch["alpha1"]
alpha_garch <- coef_garch["alpha1"]
beta_garch  <- coef_garch["beta1"]

persistence_arch  <- alpha_arch
persistence_garch <- alpha_garch + beta_garch

cat("ARCH(1) persistence:", persistence_arch, "\n")
cat("GARCH(1,1) persistence:", persistence_garch, "\n")

ARCH(1) persistence: 0.06494171
GARCH(1,1) persistence: 0.9597853
```

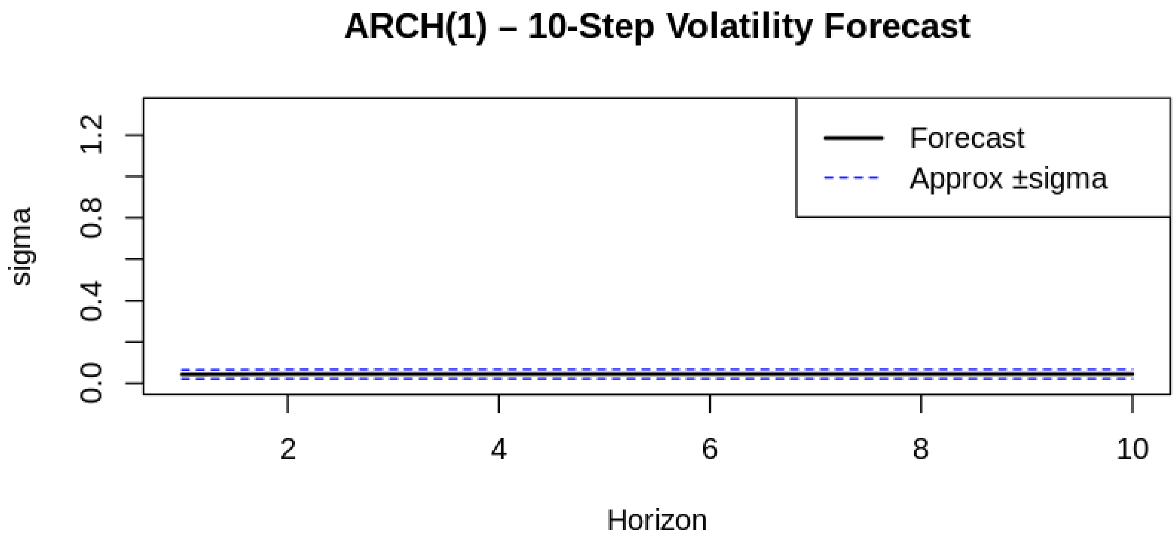
Identifying the conditional volatility shows that the range of the ARCH model of stock returns in contrast to the GARCH model has a very short variance range indicating that shocks to stock returns have a minimal and short-lived impact on future volatility, whereas FX returns modeled by GARCH exhibit large, persistent volatility clusters that carry over multiple periods. Further, acf plots shows that most of the volatility clustering was removed, with only the first lag being consistently significant. Although some minor lags are significant indicating higher level of ARCH/GARCH might be appropriate if percision is relevant. Lastly, persistance shows how much is the variance sustained. The test shows that the ARCH has a low persistance while the GARCH has a very high persistance which is consistent to our previous analyses.

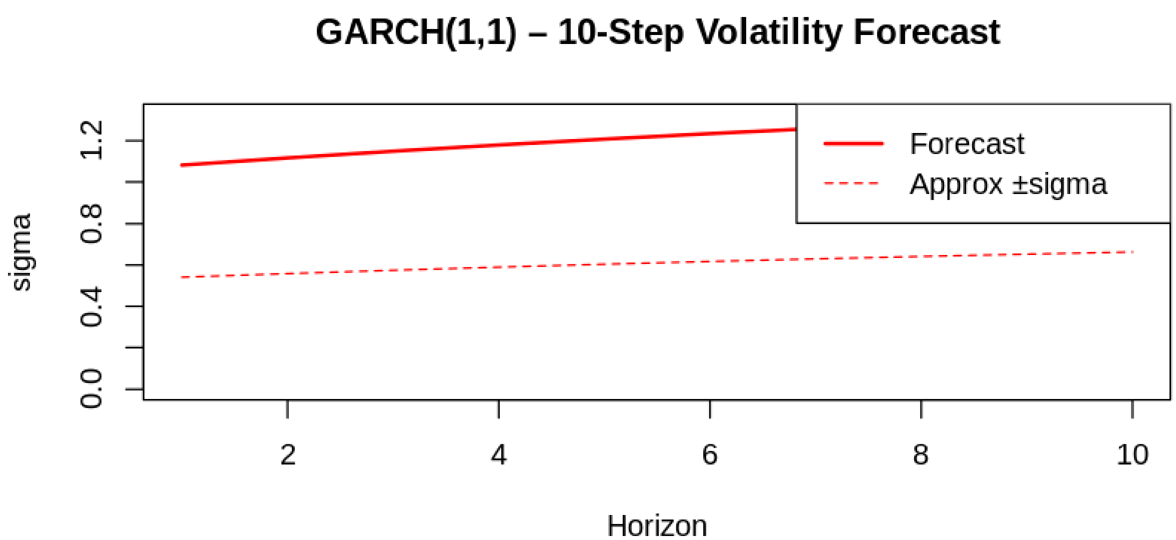
Volatility Forecasting

- Forecast conditional volatility for 10 future periods for each dataset.
- Plot the forecasted volatility path with confidence intervals.
- Explain what happens to volatility over the forecast horizon (does it decay, persist, or show asymmetry?).

```
forecast_arch  <- ugarchforecast(fit_arch_stock, n.ahead = 10)
forecast_garch <- ugarchforecast(fit_garch_fx, n.ahead = 10)

vol_arch  <- sigma(forecast_arch)
vol_garch <- sigma(forecast_garch)

h <- 1:10

par(mfrow=c(2,1))
```

```
plot(h, vol_arch, type="l", lwd=2,
     main="ARCH(1) – 10-Step Volatility Forecast",
     xlab="Horizon", ylab="sigma",
     ylim=c(0, max(vol_arch, vol_garch)))
lines(h, vol_arch*0.5, lty=2, col="blue")
lines(h, vol_arch*1.5, lty=2, col="blue")
legend("topright", legend=c("Forecast","Approx ±sigma"), lty=c(1,2),
col=c("black","blue"), lwd=c(2,1))

plot(h, vol_garch, type="l", lwd=2, col="red",
     main="GARCH(1,1) – 10-Step Volatility Forecast",
     xlab="Horizon", ylab="sigma",
     ylim=c(0, max(vol_arch, vol_garch)))
lines(h, vol_garch*0.5, lty=2, col="red")
lines(h, vol_garch*1.5, lty=2, col="red")
legend("topright", legend=c("Forecast","Approx ±sigma"), lty=c(1,2),
col=c("red","red"), lwd=c(2,1))
```





The plots shows the volatility of the predicted outcomes of the ARCH and GARCH model wherein we can observe due to the data having unsustained varaince the variance of the model produces is near constant zero, while the variance of the predicted outcomes of the GARCH model is rising as the horizon gets larger but it still exibits a slow pace of increase.