Using log returns to determine the p and of GARCH

Yes, you can use the log return of the adjusted close to determine the optimal values of p and q for a GARCH model. In fact, it is a common approach to select the values of p and q based on the autocorrelation and partial autocorrelation functions of the log returns.

You can use the autocorrelation function (ACF) and partial autocorrelation function (PACF) plots to identify the appropriate values of p and q. The ACF plot shows the correlation between the log returns at different lags, while the PACF plot shows the correlation between the log returns at a specific lag after removing the effects of the shorter lags.

If the ACF plot shows a slow decay and the PACF plot shows a sharp drop after a certain lag, it suggests that the log returns have some autoregressive (AR) structure. On the other hand, if the ACF plot drops sharply and the PACF plot shows a slow decay, it suggests that the log returns have some moving average (MA) structure.

You can also use the Bayesian Information Criterion (BIC) or the Akaike Information Criterion (AIC) to choose the optimal values of p and q. These criteria balance the goodness of fit of the model with its complexity, penalizing models with too many parameters.

Once you have determined the optimal values of p and q, you can use the GARCH model to estimate the volatility of the log returns and incorporate it into your trading strategy. For example, you can use the GARCH forecasts to set stop-loss levels or to adjust the position size of your trades based on the expected volatility.

Trial 2

Sure, here is a plan for building a trading strategy using the talib python library:

1. Identify the financial instrument and time frame that you want to trade. For example, you might want to trade the S&P 500 index using daily candlestick data.
2. Load the historical price data for the instrument and time frame into a pandas dataframe. You can use the pandas\_datareader library to fetch the data from a financial data provider such as Yahoo Finance.
3. Use the talib library to calculate technical indicators on the price data. Some popular indicators to consider might include moving averages (such as the simple moving average or exponential moving average), oscillators (such as the relative strength index or stochastic oscillator), or trend indicators (such as the average directional index or parabolic SAR).
4. Define the trading rules based on the technical indicators. For example, you might buy when the price crosses above the 50-day moving average and the relative strength index is below 30, and sell when the price crosses below the 50-day moving average and the relative strength index is above 70.
5. Implement the trading rules in code using the pandas dataframe and the talib indicators. You can use the pandas "apply" method to apply the trading rules to the price data and generate a series of buy and sell signals.
6. Backtest the trading strategy using historical data to evaluate its performance. You can simulate trades based on the buy and sell signals and calculate the profit and loss over time. You can also use performance metrics such as the Sharpe ratio or maximum drawdown to evaluate the strategy.
7. Optimize the trading strategy by adjusting the parameters of the technical indicators or the trading rules based on the backtesting results. You can use tools such as grid search or genetic algorithms to find the optimal parameters.
8. Monitor the performance of the trading strategy in real-time and adjust the parameters as needed. You can use a trading platform such as Interactive Brokers or Alpaca to execute trades based on the signals generated by the strategy.

Overall, building a successful trading strategy requires a combination of domain expertise, statistical analysis, and programming skills. The talib python library can be a powerful tool for analyzing price data and generating trading signals, but it is important to combine it with other techniques and data sources to build a robust and profitable strategy.