# 거래량 0 데이터의 처리 방법 심층 분석

금융 시계열 데이터에서 **거래량이 0인 기간(즉 해당 간격에 거래가 전혀 없는 캔들)**이나 데이터 누락 구간을 어떻게 처리할지는 백테스팅과 실거래 시스템 모두에서 매우 중요합니다. 아래에서는 Pandas 기반 기술 지표 라이브러리들의 내부 동작부터 업계 표준 관행, 그리고 세 가지 대표적 처리 방법의 비교와 각 방법이 기술 지표 계산에 미치는 영향, 마지막으로 백테스팅 측면에서 최적의 방안을 순서대로 다룹니다.

## 1. pandas-ta와 TA-Lib 등의 내부 처리 방식

**pandas-ta**(Pandas Technical Analysis)와 **TA-Lib**(Technical Analysis Library)은 입력된 데이터를 기반으로 기술 지표를 계산하는 라이브러리입니다. **이러한 라이브러리들은 거래량(Volume)이 0인 데이터에 대해 특별한 예외 처리를 하지 않고, 주어진 값을 그대로 사용**하는 것이 일반적입니다. 즉, 라이브러리 자체에서는 “거래량이 0이므로 계산에서 제외한다”는 식의 내부 로직은 없으며, **사용자가 제공한 OHLCV 데이터를 신뢰하여 계산을 수행**합니다.

* **가격 기반 지표** (이동평균, RSI 등): 거래량이 직접 사용되지 않는 지표의 경우, 거래량이 0이어도 지표 계산에는 영향이 없습니다. 다만 **만약 해당 캔들의 시가/고가/저가/종가 값이 모두 동일(가격 변동 없음)**하거나 이상치라면, 그 가격 데이터 자체가 지표 계산에 들어갑니다. 예를 들어 **RSI**의 경우 가격 변화분으로 계산되는데, 거래량이 0인 구간에서 가격이 변하지 않았다면 해당 구간의 **상승분(up)과 하락분(down)이 모두 0**으로 처리됩니다. 이때 RSI 계산식에서는 **이전까지의 평균 상승폭과 평균 하락폭에 0값을 추가하여 계속 계산**할 뿐, 별도의 조정 없이 RSI 값이 산출됩니다. (일반적인 RSI 구현은 Wilder의 방식으로서 가격 변화가 없으면 이전 RSI 수준을 유지하거나 점차 50에 수렴하는 경향을 보입니다.)
* **거래량 기반 지표** (OBV, MFI 등): **거래량이 지표에 직접 사용되는 경우에도, 0은 그냥 0으로 취급**됩니다. 예를 들어 **OBV(On-Balance Volume)**는 종가의 상승/하락에 따라 거래량을 더하거나 빼는 누적 지표인데, TA-Lib이나 pandas-ta의 OBV 구현은 다음과 같은 공식을 따릅니다:

*종가가 이전 종가보다 높으면*: OBV = 이전 OBV + 현재 거래량  
*종가가 이전 종가보다 낮으면*: OBV = 이전 OBV – 현재 거래량  
*종가가 이전 종가와 같으면*: OBV = 이전 OBV 유지[[1]](https://www.investopedia.com/terms/o/onbalancevolume.asp#:~:text=1,Previous%20OBV%20%2B%20today%27s%20volume)

이 공식에서 **거래량이 0일 경우**, 종가 상승 시 OBV += 0, 하락 시 OBV –= 0이므로 **종가가 오르든 내리든 OBV 값은 변하지 않습니다**. 즉 **OBV는 거래량 0인 캔들에서는 그대로 유지**됩니다[[1]](https://www.investopedia.com/terms/o/onbalancevolume.asp#:~:text=1,Previous%20OBV%20%2B%20today%27s%20volume). 이는 특별 처리라기보다 OBV 공식에 따라 자연스럽게 나온 결과입니다. **MFI(Money Flow Index)** 등의 지표도 거래량이 0이면 그 기간의 자금흐름이 0으로 계산되므로 지표에 변화를 주지 않습니다.

* **NaN 처리**: Pandas-ta나 TA-Lib 함수들은 **입력 값이 NaN(결측값)**인 경우를 대비한 옵션을 일부 제공합니다. 예컨대 다른 파이썬 기술분석 라이브러리인 ta (bukosabino의 *Technical Analysis Library in Python*)에는 fillna 파라미터가 있어서 **계산 결과 중 NaN인 부분을 일정 값으로 채우는** 기능이 있습니다[[2]](https://technical-analysis-library-in-python.readthedocs.io/en/latest/ta.html#:~:text=match%20at%20L300%20,if%20True%2C%20fill%20nan%20values)[[3]](https://technical-analysis-library-in-python.readthedocs.io/en/latest/ta.html#:~:text=match%20at%20L499%20,fill%20nan%20values%20with%2050). 그러나 **거래량이 0인 것은 NaN이 아니므로 이런 fillna 옵션과는 무관**합니다. Pandas-ta와 TA-Lib 모두 **0은 정상적인 수치로 간주**하며, 별도로 건너뛰거나 처리하지 않습니다.

요약하면, **pandas-ta와 TA-Lib 등은 거래량 0인 데이터에 대해 특별한 내부 예외 처리를 하지 않고**, 가격 데이터와 거래량 데이터를 **있는 그대로 사용하여 지표를 계산**합니다. 따라서 **데이터 전처리 단계에서 거래량 0 캔들을 어떻게 다룰지는 사용자 몫**이며, 잘못 처리된 0값 (예: 가격이 0으로 표시된 캔들)을 넣으면 지표 계산 결과가 왜곡될 수 있습니다. **이러한 라이브러리는 “쓰레기 입력 -> 쓰레기 출력 (Garbage In, Garbage Out)” 원칙을 따르므로**, 데이터 정제에 각별히 신경 써야 합니다.

## 2. 알고리즘 트레이딩 업계의 표준 데이터 전처리 방법론

알고리즘 트레이딩 및 백테스팅에서 **“거래 없는 캔들”**을 다루는 방법은 통계적 일관성과 실전 가정에 큰 영향을 미칩니다. **업계 표준**으로 널리 언급되는 방법론은 다음 두 가지 관점으로 나뉩니다.

* **(A) 데이터 공급자 측 관점**: 많은 시계열 데이터 공급자들은 **거래가 전혀 없었던 기간의 바(bar)를 아예 생성하지 않고 생략**합니다. 예를 들어 **Polygon.io**와 같은 데이터 제공사의 경우 “해당 기간에 **유효한 거래(체결)가 없으면 그 구간의 OHLCV 바를 생성하지 않는다”**고 명시합니다[[4]](https://polygon.io/knowledge-base/article/why-are-there-missing-aggregates-in-polygons-data#:~:text=We%20do%20not%20populate%20an,occurred%20during%20that%20aggregate%20period). 실제 Polygon.io의 1분봉 데이터에는 거래 없는 분(minute)은 빈 데이터(누락)로 나타나며, 연속적 분봉 시계열에 결측이 생깁니다. 이처럼 **데이터 소스에서 애초에 거래 없는 기간을 제공하지 않는다면, 사용자는 이를 별도로 감안하여 처리**해야 합니다. (예: Pandas에서 시간 인덱스로 reindex하여 빈 기간을 채워넣거나, 혹은 그대로 두고 분석 로직에서 결측으로 인식.)
* **(B) 트레이더/분석가 측 관점**: **일부 차트/charting 소프트웨어나 트레이딩 플랫폼은 시간의 연속성을 유지하기 위해 거래 없는 기간도 “빈 캔들”로 포함**시키기도 합니다. 이러한 경우 **해당 캔들의 시가/종가를 이전 가격으로 설정하고, 고가/저가는 그 이전 가격과 동일하게 처리**하는 것이 관례입니다[[5]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=%E2%80%A2%20%204y%20ago)[[6]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=High%2Flow%20are%20the%20weird%20ones,them%20equal%20to%20the%20Open). 실제로 업계에서는 **“거래량 0인 캔들은 이전 종가에서 가격이 변하지 않은 것으로 본다”**는 암묵적 규칙이 있습니다[[7]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Just%20assume%20the%20price%20is,So%20if%20the%20data%20is)[[8]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Yep%20this%20is%20it,3%20days%20ago). 예를 들어 한 트레이더 커뮤니티에서는 다음과 같은 변환을 정석으로 소개합니다:
* 원본 데이터 (가운데 캔들이 거래량 0인 경우):  
  Open: 123, Close: 124  
  Open: 0, Close: 0 (← 거래 없음)  
  Open: 122, Close: 122
* 처리된 데이터 (거래 없음 구간을 이전 가격으로 유지):  
  Open: 123, Close: 124  
  Open: 124, Close: 124 (← 이전 종가로 채움)  
  Open: 122, Close: 122

위 사례처럼, **거래 없던 캔들의 Open/Close를 이전 캔들의 Close 가격으로 대체**하고, **High/Low도 해당 가격으로 설정**하는 방식입니다[[7]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Just%20assume%20the%20price%20is,So%20if%20the%20data%20is)[[8]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Yep%20this%20is%20it,3%20days%20ago). 이렇게 하면 차트에서 **해당 기간 가격이 횡보(수평 유지)한 것으로 표시**됩니다. 많은 전통적 차트 툴과 데이터베이스가 이 원칙을 사용해 왔습니다. (예: 증권사 차트에서 거래량 0인 날은 봉이 이어지며 가격이 변하지 않는 것으로 나타나는 경우가 있음.)

* **(참고)**: 시장마다 처리 방식의 차이는 있습니다. **주식 시장**의 일별 데이터는 주말/공휴일이 데이터에 없지만 굳이 채우지 않는 게 일반적입니다 (그 기간은 **시장 자체가 휴장**이므로 *분석 대상이 아님*). 반면 **암호화폐 시장**은 24/7 거래되므로 **원칙적으로 시간 연속적인 데이터**이며, 극단적으로 유동성이 낮은 코인에 한해 특정 기간 거래가 없을 수 있습니다. 따라서 **주식의 “휴장일”**과 **실시간 시장의 “거래 없는 간격”**은 구분해야 합니다. **백테스트용 데이터셋을 구축할 때**도, 주식은 거래일 기준으로만 구성하고 휴장일은 제외하는 것이 일반적이고, 암호화폐는 연속 시간으로 취급하는 편입니다.

요약하면, **업계 표준 방법론은 크게 두 갈래**입니다: **데이터를 연속 시간으로 보고 비어있는 구간은 이전 가격으로 메꿔 지속시키는 방법**, 혹은 **아예 해당 바를 생략하고 불연속으로 처리하는 방법**입니다. **전문 백테스팅 시스템**에서는 대개 **데이터 제공 형식에 따라** 이 중 하나를 택하여 일관성 있게 처리합니다. 중요한 것은 **일관된 기준**이며, 어떤 방식을 택하든 **전략 로직에서 이로 인한 영향 (예: 지표 계산, 트레이딩 신호, 체결 가능성 등)을 충분히 고려**하는 것이 업계의 공통된 인식입니다.

## 3. 세 가지 주요 데이터 처리 방법 비교

거래량 0인 구간을 처리하는 대표적인 방법 세 가지를 **(방법 1) 유지**, **(방법 2) 제거**, **(방법 3) 채우기**로 나누어 상세히 살펴보겠습니다. 각 방법마다 **(a) 개념과 설명**, **(b) Pandas를 활용한 예시 코드**, **(c) 장점과 단점**, **(d) 권장 사용 시나리오**를 다룹니다.

### 방법 1: 데이터 유지 (Provide As-Is)

**(a) 개념 및 설명:**  
**방법 1은 거래량 0인 캔들을 데이터에서 제거하지 않고 유지하되, 가격 정보를 보정하는 방식**입니다. 구체적으로는 **해당 캔들의 시가, 고가, 저가, 종가를 모두 직전 유효 캔들의 종가로 설정**합니다. 거래량은 0으로 둔 채, 가격만 이전 값으로 “들고 오는” 것입니다. 이렇게 하면 **가격 시계열은 시간적으로 연속**되며, 거래가 없던 동안 **가격이 그대로 정지해 있었던 것으로 간주**됩니다[[7]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Just%20assume%20the%20price%20is,So%20if%20the%20data%20is). 관례적으로 **고가와 저가도 이전 종가(=현재 시가)와 동일하게 설정**하여, 그 기간에 **가격 변동 폭이 없었음을 명확히 표시**합니다[[8]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Yep%20this%20is%20it,3%20days%20ago). 이 방법은 **전통적인 차트 관행과 호환**되며, 많은 차트 툴이 내부적으로 이 방식을 사용합니다.

예를 들어 5분 봉 차트에서 09:55에 마지막 거래가 100원에 체결되고, 10:00~10:10 사이 거래가 없다가 10:15에 102원에 거래가 발생했다면:  
- **처리 전**: 09:55 캔들(종가 100, 거래량 X), *10:00 캔들 없음*, *10:05 캔들 없음*, 10:10 캔들 없음, 10:15 캔들(종가 102, 거래량 Y)  
- **방법 1 처리 후**: 09:55 캔들(종가 100, 거래량 X), 10:00 캔들(시가=고가=저가=종가 100, 거래량 0), 10:05 캔들(모두 100, 0), 10:10 캔들(모두 100, 0), 10:15 캔들(시가=100, 종가=102, 고가=102, 저가=100, 거래량 Y)

위에서 10:00~10:10의 세 개 캔들은 **모두 가격 100으로 유지**되고 거래량만 0인 상태로 존재합니다. 이렇게 하면 **시계열 데이터프레임에 연속적인 인덱스(시간)가 보존**되고, **가격이 변동 없이 정체되었다가 10:15에 상승한 모양**을 갖춥니다.

**(b) Pandas 예제 코드:**

import pandas as pd  
  
# 예시 데이터프레임 생성: 시간, 가격, 거래량  
df = pd.DataFrame({  
 'Time': pd.date\_range('2023-01-01 09:55', periods=4, freq='5T'),  
 'Open': [100, 0, 0, 102],  
 'High': [101, 0, 0, 105],  
 'Low': [99, 0, 0, 100],  
 'Close': [100, 0, 0, 102],  
 'Volume':[500, 0, 0, 300]  
})  
df.set\_index('Time', inplace=True)  
  
# 거래량 0인 행들에 대해 가격 열을 이전 값으로 채우기  
mask = df['Volume'] == 0  
df.loc[mask, ['Open','High','Low','Close']] = df.loc[mask].shift(1)[['Close','Close','Close','Close']].values  
  
print(df)

위 코드에서는 **거래량이 0인 행들을 찾아(mask)**, 그 행들의 O/H/L/C를 **이전 행의 종가로 채우고** 있습니다. 실행 결과:

Open High Low Close Volume   
Time   
2023-01-01 09:55:00 100.0 101.0 99.0 100.0 500   
2023-01-01 10:00:00 100.0 100.0 100.0 100.0 0   
2023-01-01 10:05:00 100.0 100.0 100.0 100.0 0   
2023-01-01 10:10:00 100.0 100.0 100.0 100.0 0   
2023-01-01 10:15:00 100.0 105.0 100.0 102.0 300

10:00, 10:05, 10:10의 가격이 모두 **100으로 유지**되고 Volume이 0인 것을 확인할 수 있습니다. (참고: 실제 사용 시에는 shift(1) 사용에 주의해야 하는데, 위 예시는 연속 데이터라 정상 동작합니다. 첫 행이 거래량 0인 경우 등을 별도 처리할 필요가 있습니다.)

**(c) 장점과 단점:**  
- **장점:** 시간 인덱스의 연속성이 보존되므로, **모든 연속형 지표 계산에 일관성**이 생깁니다. 결측 시간이 없으므로 **이동평균, RSI 등의 계산에서 누락으로 인한 문제 발생이 적습니다**. 또한 시각적으로 차트를 그릴 때 **가격이 “틈 없이” 이어져 보이므로 자연스러운 흐름**을 얻습니다[[9]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=place,Imagine). 특히 단일 자산을 분석할 때 시간 공백을 채워 넣음으로써 **백테스트 시뮬레이션에서 매 시각 캔들을 처리**할 수 있어 구현이 단순해집니다.

* **단점:** **현실 왜곡의 위험**이 있습니다. 거래가 없었다는 것은 **시장 유동성이 없었던 구간**인데, 가격을 단순히 이전 가격으로 유지한다고 해서 **실제 시장이 그 가격에 거래 가능함을 보장하진 않습니다**[[10]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Uh%2C%20if%20you%20have%20a,a%20more%20consistent%20time%20frame)[[9]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=place,Imagine). 예컨데 **매수/매도 호가가 벌어져있거나(price gap)** 비정상적으로 넓은 스프레드 상태일 수 있지만, 방법 1로 처리된 데이터는 이를 드러내지 못하고 가격이 그대로인 것으로만 나타냅니다[[11]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=inherently%20based%20on%20time%20so,become%20flat%20relevant%20to%20the). 그 결과 **지표가 지나치게 안정적으로 계산**될 위험이 있습니다. 또한 **고가/저가를 모두 동일하게 이전 가격으로 설정**하므로, **해당 기간의 변동성이 0으로 간주**되어 **ATR** 등의 변동성 지표에 영향을 줍니다 (후述).
* **추가 단점:** 여러 자산을 동시에 볼 때, 어떤 자산은 거래 없던 구간이 가격 유지로 처리되고 다른 자산은 거래가 활발했다면, **포트폴리오 관점에서 지표나 전략 성능 비교가 왜곡**될 수 있습니다. 그리고 만약 사용자가 거래량 0 캔들의 가격을 이전 값으로 채우지 않고 그대로 두면 (일부 데이터에는 0으로 들어오는 경우가 있음), 그 **0 가격이 지표 계산에 들어가 심각한 오류**를 초래할 수 있습니다. 반드시 **0 가격을 이전 가격으로 대체**해야 하는 관리 비용이 있습니다.
* **매매 전략 시 고려사항:** 이 방법을 쓰면 **거래량 0인 기간에도 캔들이 존재하기 때문에**, 전략 백테스트 시 해당 기간에 **매매 신호가 발생할 수는 있지만 실제 체결은 일어나지 못하는 상황**이 생길 수 있습니다. 따라서 **백테스터에서 거래량 0인 바에서는 매수/매도를 시도하지 않도록 조건을 넣는 것**이 바람직합니다 (예: “거래량 0인 캔들에서는 진입X” 필터링)[[12]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Don%27t%20trade%20on%200%20volume,would%20work%20would%20be%20unrealistic).
* **요약:** 방법 1은 **데이터의 시간적 연속성을 보존하여 분석 편의성과 지표 일관성을 높여주지만**, **유동성 부족 상황을 과소평가하거나 실제 시장 위험을 숨길 수 있다는 단점**이 있습니다.

**(d) 적합한 시나리오:**  
- **단일 자산 분석**: 한 종목(혹은 코인)에 대해 백테스트 또는 인디케이터 분석을 할 때 적합합니다. 데이터가 한 자산에 국한되면, 시간 공백을 채워 continuity를 유지하는 편이 지표 계산이 수월합니다. 예를 들어 **소외주 같은 낮은 유동성 주식 종목을 개별적으로 백테스트**할 때 이 방식을 써서 RSI나 이동평균이 정상 계산되도록 할 수 있습니다. 단, 앞서 언급한 **“신호 발생 vs 실제 체결”** 문제를 알고리즘에 반영해야 합니다.

* **소규모 포트폴리오 (유사한 거래 시간대 자산)**: 자산들 모두 동일한 세션(예: 미국 주식들)으로 거래되고, 동일한 결측 패턴(예: 주말 등)이라면, 각 자산별로 방법1을 적용해도 비교적 무리가 없습니다. 모두 동일하게 휴장일은 제외하거나 채우는 식으로 통일하면 됩니다.
* **실시간 데이터 시각화**: GUI 기반으로 **실시간 차트**를 그리는 경우, 새 데이터가 들어오지 않으면 가격이 이전 값에서 멈춰있는 형태로 차트에 유지됩니다. 이 때 새로운 캔들을 0거래량으로 추가하여 가격을 그대로 표시하면, 사용자에게 “현재까지 가격 변동 없음”을 명확히 보여줄 수 있습니다. 이는 실시간 모니터링 측면에서도 자연스럽습니다.
* **권장하지 않는 경우:** **다중 자산 포트폴리오 백테스팅**에서 각 자산의 거래 공백이 제각각 다르면, 방법1을 전부에 적용하면 어떤 자산은 수백개의 0거래 캔들이 끼어들고 다른 자산은 없는 등, 시간축은 맞출 수 있으나 **전략의 복잡도가 크게 증가**합니다. 이보다는 **방법 3으로 전체 타임라인을 통합 재색인**하는 편이 더 적합합니다. 또한 **유동성 함정(liquidity trap)** 우려가 있는 전략(스프레드 민감 등)에는 방법1 결과만 맹신하면 위험하므로, 보조적으로 **호가 스프레드 데이터를 구하거나** 최소한 **거래량 0 구간에 페널티를 주는 로직**을 고려해야 합니다[[10]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Uh%2C%20if%20you%20have%20a,a%20more%20consistent%20time%20frame).

### 방법 2: 데이터 제거 (Remove)

**(a) 개념 및 설명:**  
방법 2는 **거래량 0인 행(row) 자체를 완전히 제거하여 시계열 데이터에서 제외**하는 접근입니다. 즉 **거래가 없었던 시간 간격은 통째로 건너뛰는** 것입니다. 이 방법을 적용하면 데이터프레임에서 해당 타임스탬프 행이 사라지므로, 시간 인덱스가 **불연속적**이 됩니다. 앞의 예를 다시 보면, 10:00~10:10의 3개 캔들을 아예 제거하여 09:55 다음이 바로 10:15 데이터로 연결됩니다.

이를 통해 **데이터셋에는 “실제로 거래가 있었던 시각”만 남게 되어, 허구의 데이터 포인트가 제거되는 효과**가 있습니다. 실제 많은 데이터 공급자들이 이 방식을 취합니다 (예: 앞서 언급한 Polygon.io, FirstRateData 등은 **zero-volume 바를 제공하지 않음**[[4]](https://polygon.io/knowledge-base/article/why-are-there-missing-aggregates-in-polygons-data#:~:text=We%20do%20not%20populate%20an,occurred%20during%20that%20aggregate%20period)[[13]](https://firstratedata.com/free-intraday-data#:~:text=Free%20Intraday%20Data%20For%20backtesting,gaps%20in%20the%20sequence)). 또한 트레이딩 시스템에서 **거래량이 0인 바는 실행 불가능한 시그널만 만들기에 의미가 없다고 판단**하여 분석시 삭제하기도 합니다[[12]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Don%27t%20trade%20on%200%20volume,would%20work%20would%20be%20unrealistic).

**(b) Pandas 예제 코드:**

# 방법 1의 df를 이어서 사용한다고 가정  
df\_removed = df[df['Volume'] != 0].copy()  
  
print(df\_removed)

결과:

Open High Low Close Volume   
Time   
2023-01-01 09:55:00 100.0 101.0 99.0 100.0 500   
2023-01-01 10:15:00 100.0 105.0 100.0 102.0 300

10:00~10:10의 행이 모두 제거되어, **09:55 다음에 10:15가 바로 연결**된 것을 볼 수 있습니다.

실제 코드에서도 df[df['Volume'] != 0] 같은 **조건 필터링**을 통해 간단히 제거할 수 있습니다. 또는 DateTime 인덱스에서 **공휴일이나 장외시간 등**을 제외하는 것도 이 범주에 해당합니다 (예: 주식은 주말 timestamp 제거).

**(c) 장점과 단점:**  
- **장점:** 이 방법의 가장 큰 장점은 **데이터의 “사실성”을 유지**한다는 점입니다. **거래가 없었던 시각을 제거함으로써, 남아있는 데이터 포인트는 모두 실제 체결이 있었던 시점**입니다. 따라서 **백테스트에서 시그널이 존재하는 시각 = 실제 거래 가능 시각**이므로, 거래 가능성 측면에서 일종의 **엄밀성**을 확보합니다[[10]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Uh%2C%20if%20you%20have%20a,a%20more%20consistent%20time%20frame)[[12]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Don%27t%20trade%20on%200%20volume,would%20work%20would%20be%20unrealistic). 예를 들어 거래량 0인 바를 없애면 그 바에서는 애초에 매매신호 자체가 생기지 않으므로, **유동성 부족으로 인한 백테스트/실전 괴리가 줄어듭니다**. 또한 **거래가 발생한 event들만 연결**하기 때문에, **데이터 양이 줄어 경량화**됩니다 (대용량 고빈도 데이터의 경우 거래 없는 수많은 틱을 저장하지 않음으로써 효율 증가).

* **단점:** **시간상의 불연속성으로 인해 기술 지표 계산이 꼬일 수 있습니다.** 대부분의 기술적 지표는 **고정된 기간길이(window)** 또는 **연속적인 시계열 가정** 하에 정의됩니다. 예를 들어 **20일 이동평균**은 보통 20개 *연속된 거래일*의 평균으로 계산하는데, 만약 거래일이 중간중간 빠진다면 (예: 휴장일 제외) 사실상 “20일”이 20개의 데이터포인트일 뿐 실질 시간은 더 깁니다. 마찬가지로 **분봉에서 10개 캔들 이동평균**을 계산하려는데 그 사이 몇 분이 빠졌다면, **실제로는 더 긴 시간 범위의 데이터를 섞어 평균**하게 됩니다. 이는 **시간 왜곡**을 야기할 수 있습니다. 예를 들어 5분봉 12:00와 12:30 사이에 거래 없는 3개의 캔들이 빠졌다면, **12:30의 데이터까지 6개 바를 평균**하면 공백 포함 30분 구간의 움직임을 담는 게 아니라 **실제 거래 6개 바 (공백 제외) = 12:00 이전 데이터까지 들어간 평균**이 될 수 있습니다. 결과적으로 **지표 반응이 빨라지거나 이상치가 나올 수 있습니다**. 한 사용자는 **“거래 없는 분봉을 채우지 않으면 이동평균이 제대로 작동하지 않는다”**고 언급하기도 했습니다[[14]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=%E2%80%A2%20%204y%20ago).
* 또한 **다중 자산 포트폴리오**에서 이 방법은 **데이터 정합성 문제**를 일으킬 수 있습니다. 예를 들어 자산 A는 10:00에 거래가 있었고 자산 B는 없었다면, 제거 방식 적용 시 A 데이터에는 10:00이 남고 B는 10:00이 사라집니다. 이 상태에서 **두 자산을 동시에 처리하는 백테스트**를 하려면 시간축을 맞추기 어렵습니다. 결국 다시 시간 병합 (reindex)을 해야 할 수 있습니다. 그러면 방법3과 유사해지므로, **여러 종목 동시 백테스트에는 부적합**합니다.
* **지표 및 전략 영향:** 방법2에서는 **데이터의 시간 공백을 아예 제거**하므로 **“갭(gap)”으로 인한 영향이 한꺼번에 차기 캔들에 반영**됩니다. 예를 들어 가격이 100에서 거래 없던 몇 분을 지나 105로 뛰었다면, 중간 캔들들이 없으므로 **이전 캔들(100)과 다음 캔들(105)이 연속하게 붙어 5단위 갭 상승**으로 처리됩니다. **ATR** 등의 지표는 이 갭을 **한 번에 큰 True Range로 인식**할 것이고, RSI도 중간 변화 없이 **한 번에 큰 상승으로 계산**될 것입니다. 이는 방법1이나 3에 비해 **지표의 급변 가능성을 높입니다** (왜곡 여부는 후술).
* **추가 단점:** 데이터 제거는 **실제 차트를 그릴 때 시각적 공백**을 야기합니다. 차트상 **시간 축에 빈 공간**이 생기거나, 캔들이 뚝뚝 끊어져 이어지게 됩니다. 사람이 볼 때 “아 그땐 거래가 없었구나”라고 인지할 수 있지만, **연속성을 가정한 알고리즘(예: 푸리에 변환 기반 분석)**에는 문제가 될 수 있습니다.
* **요약:** 방법2는 **데이터의 현실 매매 가능 시간만 남긴다는 점에서 엄밀**하지만, **시계열의 연속성이 깨져 많은 기술적 처리에 부담**을 줍니다. 특히 **지표 계산이나 멀티-시리즈 동기화에 추가 로직**이 필요합니다.

**(d) 적합한 시나리오:**  
- **엄밀 백테스트 (체결 가능성 중시)**: 전략이 **유동성 문제에 민감**하거나, **빈약한 거래가 발생한 시그널을 완전히 배제하고 싶을 때** 방법2가 유용합니다. 예를 들어 **고빈도 트레이딩 전략**의 백테스트에서 거래 공백 구간은 의미 없으니 다 빼버리고, tick이 발생한 순간들로만 시뮬레이션을 돌릴 수 있습니다. 또한 **“거래량 0인 날에는 진입하지 않는다”**는 룰이 있는 경우, 그런 날 자체를 데이터에서 빼고 보는 것도 방법입니다.

* **일부 옵션/선물 데이터**: 유동성이 매우 낮은 옵션 등은 장중에도 거래가 띄엄띄엄 있을 수 있는데, 이를 백테스트할 때 **체결 이벤트 기준으로만 시뮬레이션**하는 경우가 있습니다[[15]](https://quant.stackexchange.com/questions/69458/backtesting-option-strategies-with-iv-data-only#:~:text=Backtesting%20Option%20Strategies%20with%20IV,zero%20volume%20or%20even). 이때 시간 공백을 제거하고 체결 시점들만 시계열로 삼아도 됩니다.
* **단일 자산 + 지표 직접계산이 아닌 이벤트 기반 전략**: 전략 로직이 지표보다는 **가격 이벤트에 기반**하는 경우 (예: 특정 가격 도달 시점, 점프 크기 등), 공백이든 아니든 지표를 굳이 쓰지 않을 수 있습니다. 이럴 때는 차라리 데이터에서 공백을 빼서 간소화하는 편이 나을 수 있습니다.
* **적합하지 않은 경우:** **전통적인 연속 지표(모멘텀, 추세선 등)를 사용하는 경우**나 **다중 자산 동시 운영 전략**에는 부적합합니다. 이때는 차라리 방법1이나 3으로 **연속성을 유지하면서** 분석하는 게 일반적입니다. 또한 **차트 시각화** 측면에서도, 사용자에게 보이는 차트에서 시간이 뜯어진 것처럼 보이는 건 혼란을 줄 수 있어 권장되지 않습니다 (차트 툴들은 일반적으로 방법1로 표시함).

### 방법 3: 데이터 채우기 (Forward Fill)

**(a) 개념 및 설명:**  
방법 3은 **미리 전체 시간 축을 정의한 뒤, 누락된 타임스탬프를 생성하고 과거 데이터로 채우는 방식**입니다. 즉, **시간 축 자체를 기준으로 데이터프레임을 재색인(reindex)**하여 **빈칸을 NaN으로 만들고**, 이후에 **직전 시점의 유효한 값으로 전진 채움(ffill)**을 수행합니다[[16]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=%E2%80%A2%20%204y%20ago). 결과적으로 방법1과 비슷하게 **가격은 이전 값으로 유지되고 거래량은 0**이 됩니다. 다만 구현 절차상 **데이터 소스에 누락된 timestamp가 있었는지 여부를 떠나, 알고리즘적으로 시간상의 연속성을 강제한다는 점**에서 차이가 있습니다.

이 방법은 특히 **Polygon.io처럼 빈 바가 아예 제공되지 않는 경우**나, **다중 자산 데이터를 하나의 시간 그리드로 맞춰야 할 경우**에 유용합니다. Forward fill(전진 채우기)은 Pandas에서 제공하는 기능으로, **NaN을 바로 앞의 값으로 채워주는 메서드**입니다[[17]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=If%20you%20re%20using%20pandas%2C,ffill%20should%20be%20handy). 채우기 전에 NaN을 만들기 위해 **전체 기간에 대한 인덱스**가 필요하며, 보통 pd.date\_range를 사용해 시작~끝 사이 원하는 빈도(freq)의 인덱스를 생성합니다.

**(b) Pandas 예제 코드:**

# 원본 데이터: 09:55, 10:15 시 두 개 시점만 있다고 가정 (사실상 방법2 결과와 유사함)  
df\_sparse = pd.DataFrame({  
 'Time': [pd.Timestamp('2023-01-01 09:55:00'), pd.Timestamp('2023-01-01 10:15:00')],  
 'Open': [100.0, 100.0],  
 'High': [101.0, 105.0],  
 'Low': [99.0, 100.0],  
 'Close':[100.0, 102.0],  
 'Volume':[500, 300]  
}).set\_index('Time')  
  
# 5분 간격 전체 인덱스 생성 (09:55 ~ 10:15)  
full\_idx = pd.date\_range(start='2023-01-01 09:55:00', end='2023-01-01 10:15:00', freq='5T')  
df\_full = df\_sparse.reindex(full\_idx)  
  
# Forward fill로 앞 값 채우기  
df\_full[['Open','High','Low','Close']] = df\_full[['Open','High','Low','Close']].ffill()  
# Volume은 거래 없던 곳은 0으로 채움  
df\_full['Volume'] = df\_full['Volume'].fillna(0)  
  
print(df\_full)

결과:

Open High Low Close Volume   
2023-01-01 09:55:00 100.0 101.0 99.0 100.0 500.0   
2023-01-01 10:00:00 100.0 101.0 99.0 100.0 0.0   
2023-01-01 10:05:00 100.0 101.0 99.0 100.0 0.0   
2023-01-01 10:10:00 100.0 101.0 99.0 100.0 0.0   
2023-01-01 10:15:00 100.0 105.0 100.0 102.0 300.0

위에서 10:00, 10:05, 10:10 시각이 새로 생겼고, **Open/High/Low/Close가 모두 09:55의 값으로 채워진 것**을 볼 수 있습니다. **High/Low도 101, 99로 그대로 채운 점**에 유의해야 합니다. 이는 방법1의 설명과 약간 다른데, forward fill을 단순 적용하면 이런 현상이 생깁니다. **High/Low는 엄밀히 말해 이전 캔들의 값이 아니라 “그 기간에 형성된 최고/최저”여야 하므로**, 거래 없던 캔들의 high/low는 open(=이전 close)으로 하는 것이 관례에 맞습니다[[6]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=High%2Flow%20are%20the%20weird%20ones,them%20equal%20to%20the%20Open). 따라서 forward fill 후에 **High와 Low는 Open과 동일하게 수정**하는 후처리가 필요합니다. 위 예시에선 우연히 이전 캔들의 Open=100, High=101, Low=99라서 101,99로 채워졌지만, 이는 사실 10:00 캔들의 고저가로는 부적절합니다. 올바르게는 df\_full.loc[df\_full['Volume']==0, ['High','Low']] = df\_full['Close'].shift(1) 등을 추가로 적용해야 합니다.

**(c) 장점과 단점:**  
- **장점:** 방법3은 **방법1과 동일하게 시간 연속성 확보** 장점이 있습니다. 차이점은 **원본 데이터에 없는 시간까지도 아예 생성**한다는 것입니다. 즉, 데이터 소스가 제공하지 않은 구간까지 포함시켜 **전체 타임라인을 완전하게 구축**합니다. 이 때문에 **여러 자산의 데이터를 동일한 인덱스 기준으로 병합하기 수월**합니다. 예컨대 주식 A와 B의 데이터를 둘 다 1분 빈도로 full reindex하여 forward fill하면, **두 종목 모두 동일한 행 개수와 인덱스를 갖게 되어** 쉽게 합쳐서 분석할 수 있습니다. 이는 **포트폴리오 백테스팅 시 동기화에 큰 이점**입니다.

또한 Pandas의 고급 기능을 활용하여 **각 컬럼마다 다른 채움 전략**을 적용할 수 있습니다 (이미 예제에서 가격과 거래량 다르게 처리). 필요에 따라 **선형 보간 (interpolation)** 등을 할 수도 있는데, 일반적으로 거래 없던 구간은 선형보간보다는 forward fill이 논리에 맞습니다[[18]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=%E2%80%A2%20%204y%20ago).

방법3은 **프로세스가 체계적**이라서, 방법1처럼 일일이 마스크 처리하기보다, **재색인 -> ffill** 순서로 코딩하면 **결측 처리와 채움이 일관성 있게 적용**됩니다. 특히 **오랜 기간의 시계열에 걸쳐 누락된 날짜/시간**들을 자동으로 채워주므로, **공휴일이나 주말 처리도 응용**할 수 있습니다 (주말을 휴장으로 보려면 forward fill 대신 그냥 NaN으로 두거나 등 조절).

* **단점:** Forward fill 기법상 **바로 이전의 “유효한” 값이 없으면 채울 수 없습니다**. 예를 들어 **시계열의 시작 부분에 누락이 있거나, 첫 데이터 자체가 거래량 0**인 경우는 ffill로도 못 채우므로 별도 처리해야 합니다. 그리고 **거래량 컬럼처럼 0이 자연스러운 값**에는 ffill을 쓰면 안 됩니다. 볼륨을 ffill하면 마지막 거래량이 계속 흘러가 버려 **사실과 어긋나므로**, 반드시 **0으로 채우는** 식의 분리 처리를 해야 합니다. 이는 방법1에서도 같은 고민이지만, method3 구현시 실수로 df\_full.ffill() 전체 적용하면 거래량까지 이전 값 복사되는 실수를 할 수 있습니다. 그래서 예제처럼 **가격과 거래량을 분리**해 채워야 합니다.
* **고가/저가 처리:** 앞서 지적했듯이, **ffill은 각 열 개별적으로 처리**하므로 **High/Low 값이 부적절하게 유지될 수 있습니다**. 예를 들어 이전 캔들에서 고가가 종가보다 훨씬 높았다면, forward fill한 거래 없는 구간에 그 높았던 가격이 고가로 남게 됩니다. 그러나 실제로 거래가 없었으므로 **그 구간의 고가는 사실상 이전 종가(또는 open) 수준이어야 옳습니다**[[6]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=High%2Flow%20are%20the%20weird%20ones,them%20equal%20to%20the%20Open). 따라서 **forward fill 적용 후 추가 보정**이 필요하며, 이러한 세부 작업을 간과하면 데이터 무결성이 깨집니다.
* **계산 비용:** 방법3는 **필요한 경우 매우 많은 빈 행을 생성**할 수 있습니다. 특히 고빈도 데이터(틱/초단위)를 장기간에 대해 full index하면 **실제 데이터 양보다 훨씬 큰 sparse 데이터프레임**이 만들어질 수 있습니다. 그리고 ffill 연산 비용도 고려해야 합니다. 다만 Pandas는 C단에서 최적화되어 비교적 빠르게 fill을 하지만, 그래도 방법1이나 2에 비해 추가 메모리/연산이 듭니다.
* **왜곡 위험:** 방법1에서 논한 현실 왜곡 문제는 방법3도 유사하게 갖고 있습니다. **실제 존재하지 않는 캔들을 만들어내는 것이므로, 어디까지나 가정된 가격 유지일 뿐**입니다. 따라서 **가격이 움직였을 가능성(호가 변화 등)을 반영하지 못하며**, **지표가 움직이지 않고 정체되는 방향으로 영향을 줍니다** (이에 대해서는 다음 질문에서 상세히).
* **요약:** 방법3은 **다수 자산이나 데이터 누락이 많은 상황에서 유연하고 강력한 채우기 방법**입니다. **데이터 사이언스적인 접근**이기도 해서 Pandas 기능을 최대 활용할 수 있지만, **주의하지 않으면 엉뚱한 값이 채워질 수 있는 위험**도 있습니다. 전반적으로 **데이터 일관성 확보와 다차원 데이터 동기화에 가장 효과적**이지만, **현실과 동떨어진 값이 생기지 않도록 후처리 검증이 중요**합니다.

**(d) 적합한 시나리오:**  
- **포트폴리오 백테스팅**: 여러 자산을 동시에 백테스트하거나, 벤치마크 지수와 개별 종목을 함께 분석하는 경우 등 **다중 시계열 동기화**가 필요한 경우 강력히 추천됩니다. 모든 시계열을 동일한 시간축으로 강제하고, **비어있는 구간을 앞선 값으로 채워 넣으면** 데이터 병합과 지표 계산이 수월해집니다. 예를 들어 **한 종목의 거래가 정지된 동안 다른 종목은 움직이는 상황**에서, forward fill로 정지된 종목의 가격을 그대로 놔두고 비교하면 포트폴리오 수준의 지표 계산이 가능합니다.

* **데이터 누락 보정**: API 오류나 수집 누락으로 **중간에 일부 데이터가 빠져있는 경우**, 방법3으로 메꿀 수 있습니다. 이때도 이전 값으로 채우는 것이 합리적인지 검토해야 하지만, 다른 보간법보다 보수적 가정이라 많이 쓰입니다.
* **실시간 신호 처리**: 실시간 시스템에서, **정해진 주기마다 데이터가 들어와야 하는데 간혹 빈 주기가 있을 때** 이전 값으로 때우고 넘어가는 경우가 있습니다. 예를 들어 1분마다 지표를 계산하는데 어떤 1분 데이터가 안 들어오면 이전 값으로 유지한 채 진행하는 식입니다. Forward fill은 이런 실시간 스트림에도 적용 가능합니다 (누락 감지 -> 채우기).
* **적합하지 않은 경우:** **극단적으로 낮은 빈도로 드문 거래가 있는 경우** forward fill로 채우면 오히려 장기간 가격이 안 변한 걸로 나오는데, 차라리 그런 자산은 제거하거나 다른 접근이 필요합니다. 또한 **주식의 휴장 기간**처럼 **애초에 시장이 열리지 않은 시간대**를 채우는 것은 의미 없습니다 (그 시간은 거래 불가능이 명백하므로 넣지 않는 편이 낫습니다). 예컨대 일봉 데이터에 주말을 ffill로 넣을 필요는 없습니다. 이는 **데이터 도메인 지식에 기반해 결정**할 사항입니다.

요약하면, **방법1과 3은 유사한 “앞값 유지” 접근**이지만, **방법3은 데이터에 존재하지 않던 시간까지 만들어 채운다는 점에서** 특히 **포트폴리오나 멀티-에셋(backtesting) 환경에 유용**합니다. 반면 **방법2는 데이터 정합성과 현실성 면에서 깨끗하지만** 시계열 분석에는 불편함이 따른다는 차이가 있습니다.

## 4. 처리 방식에 따른 기술 지표 영향 비교

세 가지 처리 방식(유지, 제거, 채우기)은 **시계열 데이터의 모양을 바꾸므로**, 여러 **기술 지표의 계산 결과에 서로 다른 영향을 미칩니다.** 주요 지표별로 어떤 영향을 받는지 구체적으로 살펴보겠습니다 (이동평균, RSI, ATR, OBV 등을 중심으로).

### 이동평균 (MA) 지표에 미치는 영향

* **방법 1 & 3 (연속 유지)**: 거래 없는 구간의 가격을 **변화 없이 유지**하므로, **이동평균선이 해당 기간 동안 수평으로 유지**되는 효과가 나타납니다. 예를 들어 **10기간 단순이동평균(SMA)**을 계산할 때, 만약 중간 3개 기간이 모두 이전 종가와 동일하다면, 평균 계산에 **동일한 값이 3번 더 포함**되어 **최근 평균이 그 가격 쪽으로 안정**됩니다. 결과적으로 **가격이 갑자기 뛰거나 떨어져도 이동평균이 완만하게 반응**합니다. 이는 **지표의 변동성이 낮아지는 쪽으로 영향**을 줍니다. 특히 **장기 이동평균선**일수록, 중간의 “평탄한” 구간 데이터가 추가되면 **노이즈를 줄여주는 효과**가 있습니다. 다만 이는 실제로 가격이 *정말로* 그 동안 안 움직였다는 가정 하에서만 타당합니다. 방법1/3은 그 가정을 취한 것이므로, **MA 계산도 가격이 정체되었다고 보고 값을 유지하거나 서서히 변화**시킵니다.

반면 **방법 2 (제거)**에서는 거래 없는 구간의 데이터가 아예 빠지므로, 이동평균 계산에서 **그 구간은 제외**됩니다. 이 경우 이동평균은 **남은 최근 N개의 실제 거래가 발생한 바**로 계산됩니다. **시간적으로 보면 더 옛날 데이터까지 포함된 셈**이 될 수도 있습니다. 예를 들어 10개 캔들 SMA인데 그 사이 3개가 제거됐다면, 실제로는 과거 13개 시간 간격에 걸친 데이터를 10개로 간주하여 평균낸 것입니다. **이동평균선이 큰 갭 직후에 확 꺾이거나 뛰는** 모습을 보일 수 있습니다. 왜냐하면 가격이 한꺼번에 변한 경우 (갭) 그 변화가 **이동평균에 바로 반영**되기 때문입니다. 방법1/3이라면 갭 구간을 채워넣어 **MA가 점진적으로 변했겠지만**, 방법2에서는 **이전 지점과 다음 지점을 바로 연결해 평균**을 내므로 **MA 값이 단번에 변화**합니다. 예를 들어 가격이 5일간 100으로 정체되다가 하루 건너뛰고 105가 된 상황에서, 5일 SMA를 비교하면:  
- 채운 경우: 100이 5번 들어가 평균 100이던 것이, 100이 4번+105 1번으로 평균 101로 **서서히 상승**.  
- 제거한 경우: 100 4번+105 1번 (총5일 중 마지막이 105, 이게 동일하지만) 사실상 기간에 공백이 있으나 계산은 동일... 좀 더 큰 차이는 만약 105가 3일 후 나왔다면, 채웠으면 MA에 100들이 들어갔을텐데, 제거하면 그냥 105와 직전 멀지 않은 과거 100들로 평균. 대체로 **방법2가 MA의 민감도를 높인다**고 볼 수 있습니다.

추가로, **지수이동평균(EMA)**의 경우 연속된 시계열을 가정하기에, 방법2처럼 불연속 데이터에 적용하면 **결과가 왜곡**될 수 있습니다. EMA는 이전 EMA값에 일정 가중치로 현재값을 합성하는데, **공백이 있으면 가중 시간이 비일관**합니다. EMA 구현은 보통 라이브러리에서 해주므로 사용자 레벨에서 구체 조정은 어렵지만, **데이터가 불연속이면 EMA도 사실상 방법2는 건너뛴 부분을 무시**하고 계산됩니다. 이는 **EMA가 실제보다 더 빠르게 최신 가격을 따라가는 효과**를 내며, 방법1/3보다 **방법2에서 EMA 반응이 빠를 것**으로 예상됩니다.

### RSI (상대강도지수) 지표에 미치는 영향

* **방법 1 & 3 (연속 유지)**: RSI는 **가격의 연속된 상승/하락 폭 평균을 이용**하는 모멘텀 지표입니다. 거래 없는 구간을 가격변화 0으로 채워 넣으면, **그 기간들의 상승폭도 0, 하락폭도 0**으로 들어가게 됩니다. RSI 계산의 핵심인 **평균 상승폭 (Avg Gain)**과 **평균 하락폭 (Avg Loss)**에 여러 개의 0변화값이 포함되면, **이동평균 계산 상 이전 값들을 희석하거나 그대로 유지하는 효과**가 있습니다. Wilder의 RSI 공식을 따른다면, **가격 변동이 없을 때는 이전 RSI 값이 크게 변하지 않고 유지**되는 경향이 있습니다. 예를 들어 **RSI가 60이었던 상태에서 몇 기간 연속 가격이 그대로라면, RSI가 계속 60 부근을 유지**할 확률이 높습니다 (Wilder 방법에서는 변화가 0이면 이전 평균이 지수적으로 감소하지만 상승과 하락 평균 모두 비슷하게 감소하여 비율은 일정하게 유지됩니다). 또는 장기간 변화 없으면 **서서히 50으로 수렴**한다는 설명도 있는데, 실제 구현상 RSI=50은 **상승도 하락도 없는 중립 상태**를 나타내므로, 가격 정체가 오래 지속되면 **RSI가 50 근처에서 안정되는 경우**도 볼 수 있습니다.

결과적으로, 방법1/3은 **RSI를 안정화/평탄화**시킵니다. **급격한 RSI 변화**(예: 과매수 70 돌파 등)가 **지연되거나 완화**될 수 있습니다. 예를 들어 가격이 100에서 105로 뛰었을 때, 중간에 여러 캔들에 걸쳐 천천히 올랐다고 가정(=forward fill된 값 + 마지막에 상승)하면 RSI도 단계적으로 상승할 것입니다. 반면 방법2처럼 바로 100에서 105로 상승한 것으로 보면, RSI는 **한 번에 큰 상승폭을 반영**하여 이전보다 훨씬 큰 값으로 튈 수 있습니다.

* **방법 2 (제거)**: 거래 없는 기간을 제외하면, **RSI 계산에서 그 기간들은 고려되지 않습니다.** 이것이 구체적으로 어떤 차이를 초래할지 이해하려면, RSI의 **기간 설정**을 떠올려야 합니다. 일반적으로 RSI 14라 하면 **14개의 데이터포인트 연속으로** 필요합니다. 방법2에서는 결측 구간이 빠지므로, **이전 14개의 실제 거래 바**를 사용하게 됩니다. 만약 중간에 3개의 바가 제거됐다면, **시간상으로는 17개의 기간**을 건너뛰어 14개 데이터포인트를 모은 셈입니다. 이 기간 동안 가격 변화가 없었다면, 방법1에서는 0변화 3번이 들어가 RSI가 꽤 유지됐겠지만, 방법2에서는 그 0변화들이 애초에 없던 일이 되므로 **RSI가 더 민감하게 반응**할 수 있습니다. 특히 **큰 가격 갭**이 발생한 경우, 방법2에서는 **그 갭이 하나의 큰 변화로 RSI에 반영**됩니다. 예를 들어 RSI14 계산 중 13번째와 14번째 데이터 사이에 가격이 5% 뛰었지만 그 사이 바가 없었다면, **RSI의 평균 상승폭에 5% 한 번만 들어가고 나머진 0이 아니라 “없음”**이므로, **평균 상승폭이 상대적으로 크게 나오게 됩니다**. 이는 **RSI 값이 방법1보다 높게** 나올 것입니다.

또한 **RSI의 초기화** 측면에서도 차이가 있습니다. RSI는 보통 첫 계산 시 이전 값들 평균으로 시작하는데, 방법2에선 데이터포인트 수가 적어지면 **초기 RSI 계산 시점이 달라질 수 있습니다**. 하지만 이는 경미한 이슈고, 전반적으로 **방법2는 RSI 곡선을 좀 더 요동치게** 만듭니다. **과매수/과매도 신호가 빨리 포착**될 수도 있지만, 그것이 실제론 거래 없는 정체 후 튀었기 때문에 생긴 것일 수도 있습니다.

### ATR (평균진폭) 지표에 미치는 영향

* **방법 1 & 3 (연속 유지)**: **ATR(Average True Range)**은 **진폭(변동폭)을 측정**하는 지표입니다. True Range(TR)은 **현재 고가-저가, 현재 고가-이전 종가 차이, 현재 저가-이전 종가 차이 중 최대값**으로 정의됩니다[[19]](https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/atr#:~:text=ATR%20%3D%20%28Previous%20ATR%20,%2F%20n). 거래 없는 구간을 채우면, **그 구간의 고가와 저가가 모두 이전 종가 수준**이므로 **TR이 대체로 0에 가깝게 나옵니다**. 예를 들어 가격이 100에서 몇 기간 유지되면, 유지된 각 기간마다 고가-저가=0, 고가-이전종가=|100-100|=0 등으로 **TR=0**이 계산됩니다. 결국 **ATR 계산 창에 0이 여러 개 포함**되면, **ATR 값이 하락**하게 됩니다. ATR은 **변동성의 이동평균**인데, 중간에 변동성이 0인 구간이 길면 **전체 평균을 끌어내리는 효과**가 있습니다. 따라서 **방법1/3은 ATR 지표를 낮추거나 안정화시키는 방향**으로 영향을 줍니다. 특히 **장기간 거래 정체 후 다시 움직이는 자산**이라면, 그 정체기간 동안 ATR이 계속 낮아졌다가 이후 가격 움직임이 시작되어도 ATR이 한동안 낮게 유지될 수 있습니다 (과거 14기간 중 많은 수가 0 변동이었으므로). 이는 **변동성 돌파 전략** 등의 경우 신호 지연이나 잘못된 기대(변동성 낮다고 안심)로 이어질 수 있습니다.
* **방법 2 (제거)**: 거래 없는 캔들을 제거하면, **ATR 계산에서 그 0 변동 구간들이 제외**됩니다. 따라서 **ATR이 더 높게 유지**되는 효과가 있습니다. 예를 들어 앞과 동일한 상황에서, 방법2에서는 0 TR 값들을 빼버리니 ATR이 상대적으로 덜 감소합니다. 더 중요하게는, **가격이 급변하는 갭이 발생했을 때 ATR이 그 갭을 즉시 크게 반영**합니다[[20]](https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/atr#:~:text=Average%20True%20Range%20,use%2020%20to%2050%20periods). ATR은 정의상 이전 종가와 현재 가격의 차이도 TR에 넣기에, **공백을 제거한 데이터에서는 이전 실제 거래 가격과 다음 실제 가격 차이가 TR으로 산정**됩니다. **방법1/3에서는 이전 종가와 현재 가격 사이의 차이를 한 번에 인식하지 못할 수도 있습니다.** 예를 들어 3일간 거래 없던 주식이 금요일 100 -> 월요일 110이 됐다면:
* 방법2: 금요일(100) 다음 월요일(110) 연결 -> TR=max(110-110, |110-100|, |110-100|)=10, ATR에 10 반영.
* 방법1/3: 토/일 데이터를 100으로 넣고 월요일 110 -> 월요일 TR=max(110-110, |110-100|, |110-100|)=10 (사실 동일하게 10이 반영됨).  
  갭의 경우는 결국 비슷하지만, **중간에 서서히 변했더라면** 차이가 났을 것이라는 점이 중요합니다. 예를 들어 5분봉에서 09:55 100, 10:15 105 (중간 없음):
* 제거: 09:55->10:15 바로 비교 -> TR=max(105-105, |105-100|, ...)=5 (큰 변동)
* 유지: 10:00,10:05,10:10을 100으로 채움, 10:15 105 -> 10:15 TR=max(105-105, |105-100|, ...)=5 (같음).  
  이 예선 동일하네요, 좀 다른 예: 만약 09:55 100, 10:00 없음, 10:05 102 실제 (즉 10:00 공백 후 10:05 체결):
* 제거: 09:55->10:05 -> TR=max(102-102, |102-100|, |102-100|)=2.
* 채움: 10:00 100 채워짐, 10:05 102 -> 10:05 TR=max(102-102, |102-100|, |102-100|)=2 (같음...).  
  사실 ATR 공식이 이전 종가를 항상 참조하기 때문에, **방법1/3과 방법2가 ATR 결과에서 차이를 보이는 경우는 “이전 종가”의 정의가 달라질 때**입니다. 거래 없는 기간이 길면, 방법2에선 이전 종가가 훨씬 과거 것이 될 수 있고, 방법1에선 바로 직전(채워넣은) 것이 됩니다. 다만 그 이전 종가 값은 결국 같으니 TR 동일... 헷갈릴 수 있는데, **핵심은 방법1/3에서는 여러 기간에 걸쳐 분산될 변동성이, 방법2에서는 응축된다**는 것입니다. ATR은 어차피 그 갭을 잡아내지만, **방법1/3에서는 갭 이후에도 한동안 0변동 구간이 있었다면 ATR이 낮은 값들을 가져오면서 상승폭이 완만**할 수 있고, 방법2에서는 아예 갭 직전까지 변동 없던 기간이 없으므로 **ATR이 상대적으로 높게 유지**될 수 있습니다.

결론적으로, **방법2는 ATR 값을 다소 보수적으로(높게) 유지**시켜 **실제 변동성을 과소평가하지 않도록** 해줍니다. 반면 방법1/3은 **장기간 정체 시 ATR을 매우 낮춰버려 나중 변동성 재개 시 제대로 대응하지 못할 위험**이 있습니다.

### OBV (온-밸런스 볼륨) 지표에 미치는 영향

* **방법 1 & 3 (연속 유지)**: OBV는 앞서 설명했듯이 **종가 상승 시 누적+, 하락 시 누적-, 동일 시 변화없음**으로 계산됩니다[[1]](https://www.investopedia.com/terms/o/onbalancevolume.asp#:~:text=1,Previous%20OBV%20%2B%20today%27s%20volume). 거래 없는 기간을 가격 동일하게 유지하면 **종가가 이전과 같으므로 OBV에 변화가 없습니다**. 또한 **그 기간 거래량은 0이므로, 설령 가격이 소폭 움직이는 상황이어도 OBV 가중치는 0**이어서 OBV 값은 그대로입니다. 즉 **OBV 곡선이 수평으로 유지**됩니다. 여러 기간 연속 거래 없으면 OBV는 그 구간 내내 일자형으로 누적치 변화없이 이어집니다. **방법2 (제거)**의 경우, 어차피 거래 없는 바를 제외하므로 **OBV도 동일하게 변화없음**입니다 (없으니 고려할 변화도 없음). **OBV의 최종 누적값은 방법1/3이나 방법2 모두 동일한 결과**를 냅니다 **(실제 거래가 일어난 시점에서만 OBV 변동)**하기 때문입니다. 예를 들어 5일간 거래 없고 6일차에 거래량 1000과 함께 가격 상승 -> 방법1/3: 5일간 OBV 유지 후 6일차 +1000, 방법2: 바로 6일차 +1000, 같음.

**차이가 있다면**, 방법1/3에서는 **거래 없는 기간에도 OBV 값이 데이터로 존재**한다는 점입니다. 차트에서 보면 OBV 선이 그 기간 동안 평평하게 그려지겠지만, 방법2에서는 그 구간이 아예 없으니 OBV 선은 5일 동안 “빈 화면”으로 있다가 6일차에 점프하는 식입니다. **지표의 연속적인 가시성 측면에서 방법1/3이 부드럽게 연결**됩니다.

* **추가 고려: 기타 거래량 지표** – OBV 외에 **거래량 이동평균**, **거래량 비율(VR)** 등 **순수 거래량 기반 지표**들은 forward fill을 그대로 적용하면 안 됩니다. 보통 **거래량이 0인 기간은 0으로 넣는 게 맞으며**, 방법1/3에서도 그렇게 했다고 가정합니다. 이때 **거래량 0 값들이 지표에 들어가는 효과**가 있습니다. 예컨대 **20일 평균 거래량** 계산 시, 5일간 거래 없던 기간이 0으로 포함되면 **평균 거래량이 크게 떨어집니다**. 방법2에서는 그 5일을 제외하니 평균 계산에 들어가지 않아 **평균 거래량이 상대적으로 높게 유지**됩니다. 따라서 **방법1/3은 거래량 기반 지표에 “쉬는 기간은 거래량=0”이라고 가정하여 낮은 값으로 영향을 주고**, 방법2는 **해당 지표를 실제 거래된 날들만으로 계산**하게 됩니다. 어느 쪽이 더 바람직한지는 맥락에 따라 다른데, **예컨대 전략이 “평균 대비 거래량 급증”을 포착한다면** 방법1/3에서는 쉬던 기간 때문에 평균이 낮아져 **급증 신호가 더 쉽게 뜰 수 있고**, 방법2에서는 평균이 높아서 **신호 민감도가 낮아질** 수 있습니다.

### 요약 및 예시적인 비교

* **방법1 (As-Is 유지)**: **지표들을 안정적이고 보수적으로 보이게 만드는 경향**이 있습니다. 가격이 한동안 움직이지 않았다고 가정하므로 **추세 지표는 횡보, 모멘텀 지표는 중립에 머물고, 변동성 지표는 하락**하는 식입니다. 이후 실거래가 발생하여 가격이 움직이면 지표도 움직이지만, **그 변화가 완만하거나 지연**될 수 있습니다. 예를 들어 **이동평균선 두 개로 데드크로스 골든크로스 판단**할 때, 방법1 처리 데이터에서는 **아무 일 없던 기간 동안 단기선이 장기선과 만나지 않고 그대로 평행하게 갈 확률이 높고**, 그러다 **나중에야 교차**할 수 있습니다.
* **방법2 (제거)**: **지표들을 민감하고 즉각적으로 반응하게 하는 경향**이 있습니다. 가격이 움직일 때 **바로바로 지표에 반영**되므로, **신호 발생이 빠를 수 있지만 노이즈도 커집니다**. 특히 큰 갭 변동 시 **지표가 튀거나 꺾이는 폭이 클 수 있습니다**. 예를 들어 **RSI 전략**이라면, 방법2에서는 짧은 시간에 RSI가 30→70으로 급등락하는 일이 방법1보다 자주 나타날 수 있습니다 (왜냐하면 중간 완충이 없으므로). **이동평균 간 교차**도 방법2에서는 **조금 더 일찍** 일어나거나 (혹은 아예 빈도 높게 발생) 할 수 있습니다.
* **방법3 (Forward fill)**: **방법1과 거의 동일한 지표 영향**을 가집니다. 사실상 차이는 구현만 다르고, **데이터 결과는 “거래 없던 기간 = 가격 이전 수준 유지”로 같기 때문**입니다. 다만 주의할 점은 **forward fill 후 추가적인 보정이 필요**한 부분 (High/Low, Volume 등)을 제대로 처리하지 않으면 **지표 계산에 오류**가 생길 수 있다는 것입니다. 예를 들어 High를 ffill한 걸 그대로 두고 ATR을 계산하면, 실제로는 변동 없던 기간에 이전 멀리 과거의 high값과 비교해 **가짜 변동성**을 계산할 수도 있습니다. 이러한 실수를 방지한다면, **방법3의 지표 영향은 방법1과 본질적으로 같습니다**.
* **공통적으로**: 세 방법 모두 **지표 계산 로직 자체는 동일**합니다. 차이는 **입력 데이터의 시계열이 다르다는 것**입니다. **“쓰레기 in -> 쓰레기 out”** 관점에서, 잘못 채우거나 잘못 제거한 데이터는 지표를 잘못된 방향으로 왜곡합니다. **예시**로 “거래량 0인 캔들의 가격을 0으로 남겨둔 경우”, 이동평균은 그 0 때문에 한동안 매우 낮은 값이 되고, RSI는 엄청난 하락으로 간주해 0에 수렴할 것입니다. 이는 명백한 오류입니다. 따라서 **어떤 방법을 쓰더라도, 전처리한 데이터가 지표 계산에 논리적으로 일관되도록 하는 것**이 중요합니다.

## 5. 백테스팅 관점에서 가장 신뢰성 높은 방법과 이유

백테스팅에서는 **데이터 처리 방법에 따라 전략 성능과 신뢰도에 차이가 발생**할 수 있습니다. **특히 미래 데이터를 참조하는 편향(Lookahead bias)**이나 비현실적 가정을 최소화하는 것이 핵심입니다. 이를 종합적으로 고려할 때, **가장 권장되는 방법은 방법 1 (데이터 유지, 가격 이전값 대체)** 혹은 **방법 3 (시간축 채우기 후 forward fill)**의 접근입니다. 두 방법은 실질적으로 동일한 가정을 활용하며, 백테스팅 시 **일관성과 보수적 추정을 제공**한다는 점에서 우수합니다. 그 이유를 구체적으로 살펴보겠습니다.

**1) 미래 데이터 참조(룩어헤드) 편향 최소화:**  
방법1/3은 **이전 시점의 정보만을 사용하여 누락 구간을 채우므로** **미래 정보를 전혀 도입하지 않습니다**. Forward fill은 말 그대로 과거값을 앞으로 가져오는 것이므로, **백테스트 시 특정 시점의 캔들 데이터는 그 시점 이전까지의 실제 정보만 반영**합니다[[16]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=%E2%80%A2%20%204y%20ago). 이는 **Lookahead Bias**가 없음을 보장합니다. 반면, 만약 잘못하여 **백필(backfill)**을 하거나 미래값으로 채운다면 엄청난 편향이 생기겠지만, 우리가 논의한 방법들에는 그런 요소가 없습니다. **방법2** 역시 미래 참조는 없으나, 이는 “데이터를 삭제”하는 것이어서 편향 면에서는 안전하지만 다른 문제들이 있습니다.

**2) 거래 실행 가능성에 대한 보수적 접근:**  
백테스트에서 **거래량 0인 기간에 발생한 지표 신호나 매매 시도는 실제로는 실행 불가능**합니다. 방법1/3을 사용하면, **그 기간에 가격은 그대로이지만 거래량=0으로 남아있기 때문에**, 전략 구현 단계에서 “거래량=0이면 매매하지 않는다”는 룰을 넣어 **해당 신호를 무시**할 수 있습니다. 이때 **지표 계산은 정상적으로 이루어지되, 실행 단계에서 필터링**이 됩니다. 이는 **현실적인 제약을 반영**하는 것이며, 결과적으로 **백테스트 성능을 보다 보수적으로, 그러나 정확하게** 만들어줍니다. 실제로 알고리즘 트레이더들은 **“거래량 0인 바에서는 거래하지 말라”**고 조언합니다[[12]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Don%27t%20trade%20on%200%20volume,would%20work%20would%20be%20unrealistic). 방법1/3에서는 데이터에 그 바가 존재하므로 **전략 코드에서 이를 감지하여 skip**할 수 있습니다. 반면 **방법2**에서는 아예 그 바가 없어 신호 자체가 없으므로 얼핏 문제없어 보이지만, **지표가 왜곡**되어 다른 시점에 과도한 신호를 낼 수 있습니다. 그리고 포트폴리오 맥락에서 보면, A자산은 신호 있는데 B자산은 그 시간 바 자체가 없어 비교나 동기화가 어려울 수 있습니다.

**3) 데이터 연속성으로 인한 지표 안정성:**  
연속 데이터를 사용하는 방법1/3은 **지표 계산의 안정성을 높여주고, 백테스트에서 급작스런 지표 이상치를 줄여줍니다**. 이는 **백테스트 결과의 신뢰성**을 높이는 요소입니다. 예를 들어 방법2를 사용한 백테스트에서 RSI가 들쭉날쭉하면, 전략 로직이 그것에 민감하게 반응하여 **불필요한 매매를 하거나 변동성이 큰 성과**를 낼 수 있습니다. 반면 방법1/3으로 RSI를 계산하면 **불연속으로 인한 급변이 없으므로 전략이 보다 안정적으로 작동**합니다. **특히 포트폴리오 전략**에서는 각 자산의 지표가 동일한 기간 길이로 계산되어야 공정한데, 방법2는 자산별 데이터 포인트 수가 달라질 수 있어 위험합니다. 방법3으로 모든 자산을 같은 타임라인에 맞춰 놓고 지표를 계산하면 **동일한 기준으로 전략 판단을 하게 되어 일관성**이 있습니다.

**4) 실전 매매시의 보수적 일치:**  
방법1/3은 **실제 매매 상황을 보수적으로 흉내낸 것**입니다. 현실에서 거래가 없었다면 가격은 마지막 체결가 부근에 머물러 있었을 가능성이 높습니다. 그리고 **그 기간에는 매매가 체결될 수 없었을 것**입니다. 방법1/3의 백테스트는 **가격은 그대로 두고 체결도 안 일어나도록(거래량0)** 하여, **백테스트 중 해당 기간을 자동으로 “관망”하게 만듭니다**. 이는 실전에서도 거래가 없으면 관망할 수밖에 없는 상황을 그대로 반영한 것입니다. 반면 **방법2**는 그 관망 기간을 완전히 삭제함으로써, 백테스트 시간을 압축시켜버립니다. 이는 **전략 입장에서 보면 쉬는 시간이 사라지고 바로 다음 이벤트로 넘어가는 것**이라, **전략의 타이밍이나 포지션 홀딩 기간 등의 해석을 혼란**스럽게 만들 수 있습니다. 예를 들어 원래 1시간 동안 가격이 정체돼있다가 움직였는데, 방법2 데이터에서는 그 1시간이 증발해버리니 전략은 “바로 움직였다”고 생각합니다. **이는 백테스트 성능을 과대평가하거나 리스크 산정을 그르치게 할 수 있습니다.**

**5) Lookahead Bias와 관련된 추가 고려:**  
사실 세 방법 모두 **미래 데이터를 섞지는 않으므로** 전통적 의미의 lookahead bias는 없습니다. 하지만 **방법1/3의 경우 부주의하게 사용할 시 “앞으로 데이터가 없으면 그냥 이전 값으로 쭉 채우는” 행위가 일종의 편향을 유발할 수 있다는 지적**도 있습니다[[9]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=place,Imagine). 예컨대, 시장이 닫혀있는 밤새 가격을 그대로 유지한다고 가정하면 다음 날 개장 시 갭 리스크를 과소평가한다는 식입니다. 그러나 이는 미래를 미리 아는 것이 아니라 **미래 변동성을 0으로 가정하는 보수적 처리**이므로, 편향보다는 **보수적 시나리오**라 보는 게 맞습니다. 오히려 **방법2는 그 공백 기간 자체를 무시**해서, 전략이 “그동안 아무런 움직임도 없었음에도 기다린 시간”을 고려 못하게 합니다. 이것이 어떤 의미에선 실제보다 나은 결과(기회비용 무시)를 줄 수도 있어 일종의 편향이라 볼 수도 있습니다. 예를 들어 **하루 종일 거래 없다가 다음날 급등하면, 실제론 하루 동안 자본이 묶여있었지만 방법2 백테스트는 바로 수익 난 것으로 계산**할 수 있습니다. **방법1은 그 하루를 가격은 그대로, 포지션 홀딩 기간으로 인지**하므로 **성과 계산이 더 보수적**입니다.

종합하면, **방법1/3 (특히 forward fill 방법3)**이 **백테스팅에 가장 적합하고 신뢰성 높은 결과**를 제공합니다. **데이터의 시간 연속성을 보존하면서도 미래 정보를 사용하지 않고, 전략 구현 시 유동성 제약을 반영하기 쉽기 때문**입니다. 실제 현업에서도 **백테스트 전에 시계열 데이터의 결측을 forward fill하거나 0거래 캔들을 유지**하는 것이 일반적이며, 그런 후 **전략 로직에서 거래 가능 여부를 조건으로 확인**하는 패턴을 많이 사용합니다[[14]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=%E2%80%A2%20%204y%20ago).

마지막으로, **주의할 점은 어떤 방법을 선택해도 전략 성능을 해석할 때 반드시 그 방법의 한계를 고려해야 한다는 것**입니다. Forward fill 방식 백테스트가 결과가 좋다고 해도, **실전에서는 0거래 구간 동안 발생한 잠재적인 시장 변화(호가 변화 등)로 인해 슬리피지나 미체결 위험**이 있다는 사실을 알아야 합니다[[21]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Lots%20of%20good%20suggestions%20here,it%27s%20going%20to%20hugely%20underperform)[[9]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=place,Imagine). 그렇지만 이러한 위험은 **거래량 0 데이터를 제거한다고 해서 사라지지 않습니다** – 오히려 보이지 않을 뿐입니다. 차라리 **데이터에 남겨두고 보수적으로 접근하는 것이 위험 관리에 유리**합니다. **암호화폐와 주식 모두**에 대해, **데이터 공백/0거래 처리는 백테스트 신뢰도에 큰 영향을 미치므로 forward fill+보수적 필터링 전략**이 가장 무난한 선택임을 많은 퀀트 연구자들이 강조합니다[[10]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Uh%2C%20if%20you%20have%20a,a%20more%20consistent%20time%20frame).

**첨언:** 질문 외의 맥락까지 포함해 보면, **pandas-ta, TA-Lib 외의 다른 지표 패키지 (예: ta 라이브러리)**들도 기본적으로는 위와 같은 데이터 처리 문제를 사용자가 처리해주길 기대합니다. 어떤 패키지도 “거래량 0이면 데이터를 어떻게 해석하라”는 내장 규칙을 두고 있진 않습니다. 대신, ta 라이브러리의 경우 fillna 옵션 등을 통해 **결과 값의 NaN을 채워주기도 하지만** 이는 **계산 후 처리일 뿐 데이터 전처리가 아닙니다**. 따라서 **GUI 기반 백테스팅/실거래 프로그램**을 개발하시는 경우, **데이터 수집-전처리 모듈에서부터 거래 없는 구간 처리에 대한 옵션을 제공**하는 것이 좋습니다. 예컨대 체크박스로 “거래 없는 구간 데이터 유지(권장)” vs “삭제(고급자용)” 등을 선택하게 하거나, **암호화폐/주식 각각의 특성에 맞게 기본값**을 다르게 둘 수 있습니다. **암호화폐**는 24시간 거래되므로 보통 forward fill할 구간이 적지만, **주식**은 장 종료 후 오랜 공백이 일상적이므로 일봉에서는 제거, 분봉 이하에서는 유지 등으로 구분할 수도 있습니다. 이러한 세부 정책까지 일관성 있게 정하면, **백테스트 결과의 신뢰도가 높아지고 실거래와의 괴리가 줄어들 것**입니다.

**참고 자료:** 실제 알고리즘 트레이더들의 토론에서도 *“거래량 0 캔들은 forward fill로 채워라”*, *“이동평균 등이 시간을 기반으로 하므로 가격이 안 움직인 기간도 반영돼야 한다”* 등의 조언이 나와 있습니다[[22]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=forward%20fill%2C%20basically%20replace%20the,most%20recent%20volume%20before%20it)[[9]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=place,Imagine). 이는 저희 결론과 맥락을 같이 합니다. **결론적으로**, **백테스팅에서는 최대한 보수적으로, 실제 시장의 제약을 시뮬레이션에 반영하는 방향**이 좋으며, **방법1/3이 그러한 철학에 부합하는 데이터 처리 방법**이라고 할 수 있습니다.

[[1]](https://www.investopedia.com/terms/o/onbalancevolume.asp#:~:text=1,Previous%20OBV%20%2B%20today%27s%20volume) On-Balance Volume (OBV): How It Works and How to Use It

<https://www.investopedia.com/terms/o/onbalancevolume.asp>

[[2]](https://technical-analysis-library-in-python.readthedocs.io/en/latest/ta.html#:~:text=match%20at%20L300%20,if%20True%2C%20fill%20nan%20values) [[3]](https://technical-analysis-library-in-python.readthedocs.io/en/latest/ta.html#:~:text=match%20at%20L499%20,fill%20nan%20values%20with%2050) Documentation — Technical Analysis Library in Python 0.1.4 documentation

<https://technical-analysis-library-in-python.readthedocs.io/en/latest/ta.html>

[[4]](https://polygon.io/knowledge-base/article/why-are-there-missing-aggregates-in-polygons-data#:~:text=We%20do%20not%20populate%20an,occurred%20during%20that%20aggregate%20period) Why are there missing aggregates in Polygon’s data? | Polygon.io

<https://polygon.io/knowledge-base/article/why-are-there-missing-aggregates-in-polygons-data>

[[5]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=%E2%80%A2%20%204y%20ago) [[6]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=High%2Flow%20are%20the%20weird%20ones,them%20equal%20to%20the%20Open) [[7]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Just%20assume%20the%20price%20is,So%20if%20the%20data%20is) [[8]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Yep%20this%20is%20it,3%20days%20ago) [[9]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=place,Imagine) [[10]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Uh%2C%20if%20you%20have%20a,a%20more%20consistent%20time%20frame) [[11]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=inherently%20based%20on%20time%20so,become%20flat%20relevant%20to%20the) [[12]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Don%27t%20trade%20on%200%20volume,would%20work%20would%20be%20unrealistic) [[14]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=%E2%80%A2%20%204y%20ago) [[16]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=%E2%80%A2%20%204y%20ago) [[17]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=If%20you%20re%20using%20pandas%2C,ffill%20should%20be%20handy) [[18]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=%E2%80%A2%20%204y%20ago) [[21]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=Lots%20of%20good%20suggestions%20here,it%27s%20going%20to%20hugely%20underperform) [[22]](https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/#:~:text=forward%20fill%2C%20basically%20replace%20the,most%20recent%20volume%20before%20it) How to deal with missing zero volume candles : r/algotrading

<https://www.reddit.com/r/algotrading/comments/tg06nu/how_to_deal_with_missing_zero_volume_candles/>

[[13]](https://firstratedata.com/free-intraday-data#:~:text=Free%20Intraday%20Data%20For%20backtesting,gaps%20in%20the%20sequence) Free Intraday Data

<https://firstratedata.com/free-intraday-data>

[[15]](https://quant.stackexchange.com/questions/69458/backtesting-option-strategies-with-iv-data-only#:~:text=Backtesting%20Option%20Strategies%20with%20IV,zero%20volume%20or%20even) Backtesting Option Strategies with IV Data Only

<https://quant.stackexchange.com/questions/69458/backtesting-option-strategies-with-iv-data-only>

[[19]](https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/atr#:~:text=ATR%20%3D%20%28Previous%20ATR%20,%2F%20n) [[20]](https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/atr#:~:text=Average%20True%20Range%20,use%2020%20to%2050%20periods) What Is Average True Range? - Fidelity

<https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/atr>