

CrocSwap / blog-posts Public

[Code](#) [Issues 1](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#)

[main](#) ...

[blog-posts / drafts / markout.md](#)



a0910841082130913312 add new content

[History](#)

1 contributor

199 lines (109 sloc) | 21.5 KB

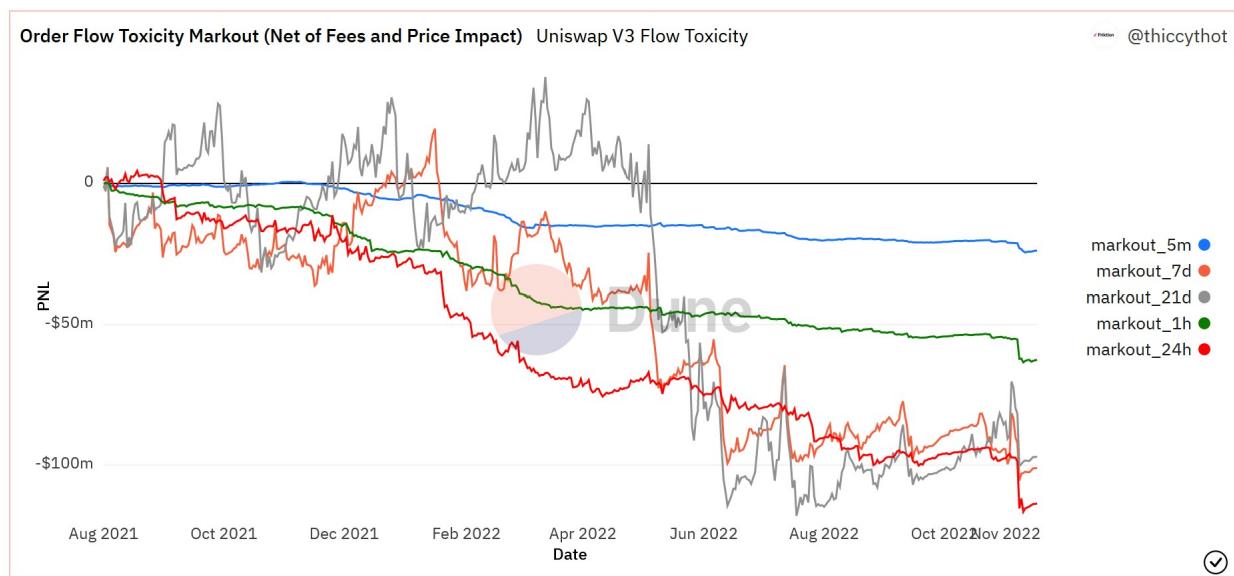
...

Usage of Markout to Calculate LP Profitability in Uniswap V3

This post is a new installment in an [ongoing series](#) by [0xfbifemboy](#) on Uniswap liquidity pools, concentrated liquidity, and fee dynamics.

Introduction

Recently, there has been a great deal of discussion around the profitability of ETH/USDC LPs on Uniswap V3. These debates centered around the validity of a [Dune dashboard](#) produced by Twitter user [@thiccyyhot_](#), which uses a markout-based calculation to demonstrate that the susceptibility of Uniswap V3 LPs to toxic order flow cost them potentially as much as ~\$100 million over the last year:



We will discuss the concept of markout in more detail later; for now, it suffices to know that it is a common metric used in high-frequency trading to analyze a strategy's profitability. (In this setting, we are imagining the Uniswap liquidity pool as implementing a trading strategy that takes the other side of every valid trade submitted to the pool.)

After being [popularized](#) by [@0xShitTrader](#), these findings prompted a [response](#) from the Uniswap Labs team, who generated [their own Dune query](#) and claimed to demonstrate that, after properly accounting for fees, Uniswap LPs were actually quite profitable in aggregate:

However, a [detailed examination](#) of the Uniswap Labs analysis by [@0xShitTrader](#) revealed that there was a major flaw with their Dune query, causing *half of all trades made* to be ignored in the analysis!



defi guy 🔥🦅
@0xShitTrader

...

That's right, one direction of trades is completely ignored.

The direction flag is set to 0 or 1 depending on the trade direction, where it should actually be set to -1 or 1.

ツイートを翻訳

```

18    select *,
19        case when cast(amount0 as real) > 0 then 1
20        | else 0
21        end as direction,
22        case when cast(amount1 as real) > 0 then amount0 -- we want to calculate markout for token0
23            else amount0 * (1 - fee / 1e6)           -- however amount0 includes the fee when its the tokenIn
24        end as adj_amt0,
25        case when cast(amount0 as real) > 0 then abs(amount1 / (amount0 * (1 - fee / 1e6)))
26            else abs(amount1 * (1 - fee / 1e6) / amount0)
27        end as effective_ex,
28        case when cast(amount1 as real) > 0 then amount1 * fee / 1e6
29            else amount0 * fee / 1e6
30        end as fee_amt,
31        last_value(px) over (
32            order by
33                evt_block_time asc range between current row
34                    and interval 5 minutes following) as 5m,
35        last_value(px) over (
36            order by
37                evt_block_time asc range between current row
38                    and interval 1 hour following) as 1h,
39        last_value(px) over (
40            order by
41                evt_block_time asc range between current row
42                    and interval 24 hour following) as 24h
43    from factory_joined
44    ), markout as (
45        select *,
46            date_trunc('minute', evt_block_time) as as_of,
47            direction * (abs(adj_amt0) / pow(10, dec0)) * ((24h - effective_ex) / pow(10, dec1) * pow(10, dec0)) *  token1_usd as markout_24hr,
48            direction * (abs(adj_amt0) / pow(10, dec0)) * ((1h - effective_ex) / pow(10, dec1) * pow(10, dec0)) *  token1_usd as markout_1hr,
49            direction * (abs(adj_amt0) / pow(10, dec0)) * ((5m - effective_ex) / pow(10, dec1) * pow(10, dec0)) *  token1_usd as markout_5m,
50            case when cast(amount1 as real) > 0 then fee_amt * token1_usd / POW(10, dec1)
51            | else fee_amt * token0_usd / POW(10, dec0)
52            end as fee_usd
53    from calculated_variables v

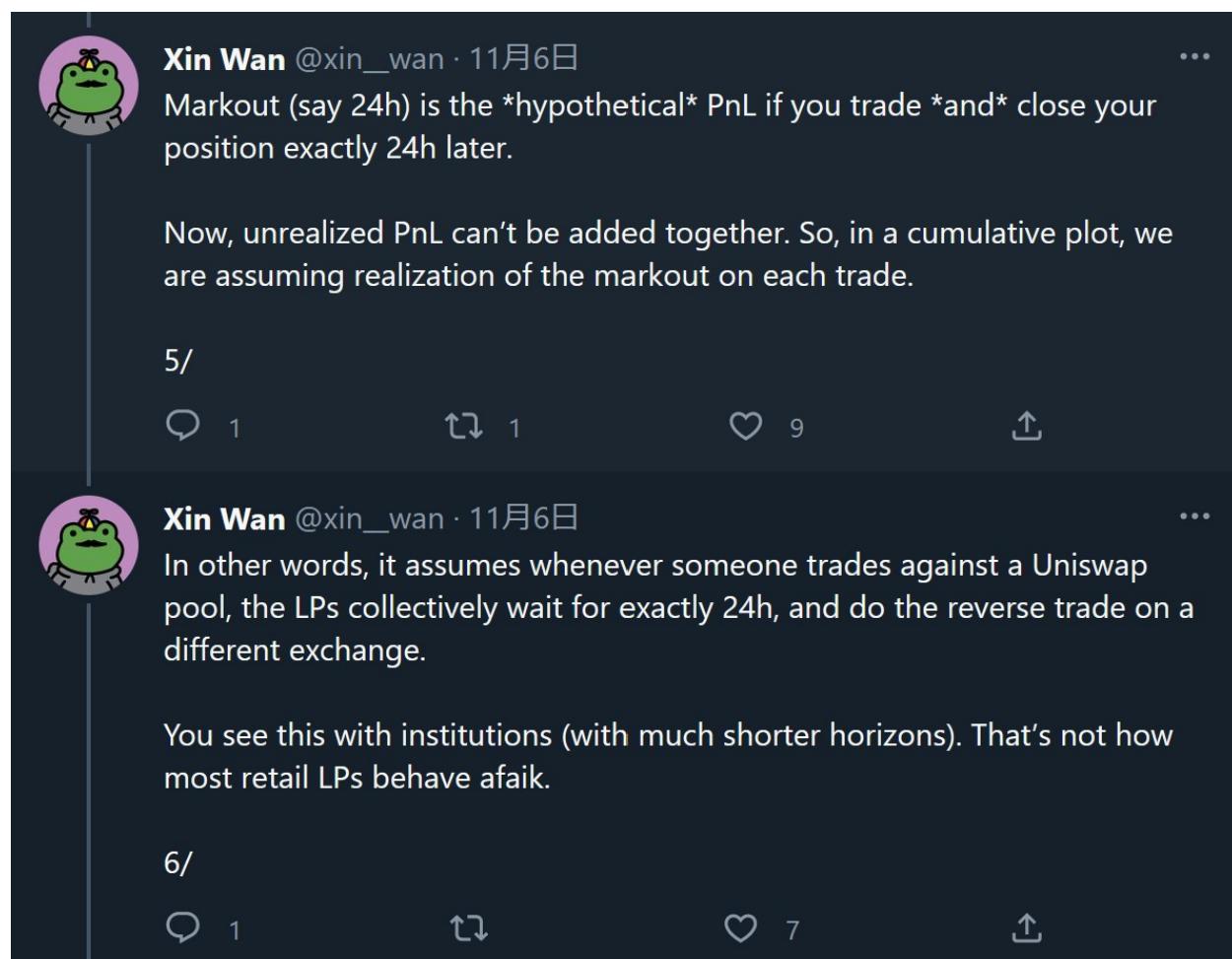
```

午前9:31 · 2022年11月2日 · Twitter Web App

Additionally, [@0xShitTrader](#) noted that fees were already accounted for in the original analysis, in contrary to the claims made by Uniswap Labs.

Correcting the single error identified above resulted in PnL plots quite similar to those originally generated by [@thiccythot_](#), suggesting that ETH/USDC LPs on Uniswap V3 are truly losing staggering amounts of money:

The error was [graciously acknowledged](#) by [Xin Wan](#) at Uniswap. That being said, they cast doubt upon the validity of markout as a useful metric in the context of AMM liquidity providers:



Xin Wan @xin_wan · 11月6日

Markout (say 24h) is the *hypothetical* PnL if you trade *and* close your position exactly 24h later.

Now, unrealized PnL can't be added together. So, in a cumulative plot, we are assuming realization of the markout on each trade.

5/

1 reply 1 retweet 9 likes

Xin Wan @xin_wan · 11月6日

In other words, it assumes whenever someone trades against a Uniswap pool, the LPs collectively wait for exactly 24h, and do the reverse trade on a different exchange.

You see this with institutions (with much shorter horizons). That's not how most retail LPs behave afaik.

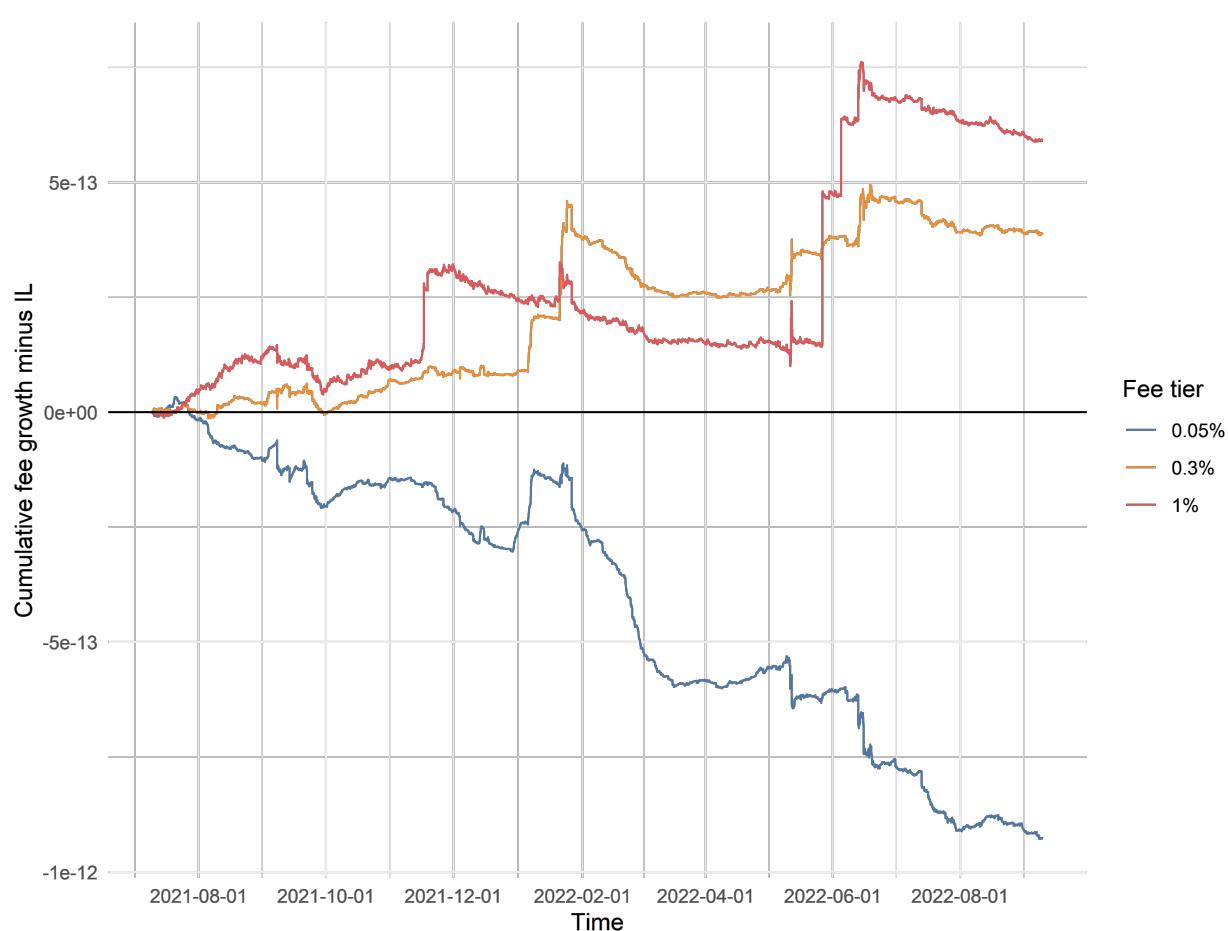
6/

1 reply 1 retweet 7 likes

The validity of markout was previously [discussed](#) at length, although consensus was not reached.

Ultimately, we are left with the overall impression that although there is disagreement about whether or not markout is a useful metric for AMM LPs, if we accept that it is, then ETH/USDC LPs have been immensely unprofitable as a result of toxic swap flow!

Curiously, though, the astute reader may recall that in one of our previous analyses, [Designing a Dynamic Fee Policy that Outperforms All Uniswap ETH/USDC Pools](#), it appeared as though a unit of ETH/USDC ambient liquidity staying in a single fee tier would actually have been net profitable in the 0.3% and 1% fee liquidity pools (considering the balance of impermanent/divergence losses versus fee accrual):



These results appear to contradict those of [@thiccyhot_](#), prompting us to dig a little deeper and identify the source of the discrepancies.

Original calculation of markout PnL

When we reviewed the Dune SQL query used by [@thiccyhot_](#) in his calculation of markout PnL, we identified two major points of disagreement, all related to the calculation of markout PnL.

Consider the following block of code which defines the 5-minute markout PnL:

```
sum(
  protocol_buySell * (markout5m - swapPrice) * eth_swapped
) as pnl_5m
```

This matches exactly the formula for calculating markout PnL given on [@thiccyhot_](#)'s Dune dashboard:

Order flow toxicity is typically comparing the execution price to a future price or "mark".

$m = \text{markout}$
 $d = \text{buy (1) or sell (-1)}$
 $v = \text{volume}$
 $f = \text{future price at time t}$
 $p = \text{execution price}$

$$m = d * v * (f - p)$$

Toxic flow is when the price marked to the future is worse than the execution price after accounting for fees and price impact. Toxicity result of adverse selection of passive market makers on Uniswap by market takers.

We will discuss various aspects in turn.

Swap fees are incorporated into the effective swap price

Let us first consider the question of fees. Is it true, as [claimed](#) by Uniswap Labs, that this analysis does not factor in LP fees? Superficially, it might appear so, because `pn1_5m` does not explicitly add on the LP fee corresponding to the swap. However, the crucial observation here is that the fee is already "baked into" `swapPrice`.

Here is how `swapPrice` is defined:

```
CASE
  WHEN t.token_b_symbol = 'USDC' then t.token_b_amount / t.token_a_amount
  WHEN t.token_b_symbol = 'WETH' then t.token_a_amount / t.token_b_amount
END as swapPrice
```

The important observation here is as [@0xShitTrader pointed out](#)—this swap price is computed from the amount of tokens transferred to and from the swapper, as reported in the Uniswap event logs. What happens when someone makes a swap is the following: they send in x tokens of one type, a fee of $(1 - \text{fee}) * x$ is taken out (for example, $\text{fee} = 0.01$ for a 1% LP fee), and y of the paired token are sent back corresponding to a sale of $(1 - \text{fee}) * x$ tokens of the first type. That is to say, when we calculate the effective execution price using the token in/out values, a higher LP fee will make it appear as though the swapper received a worse-than-market price, and conversely, that the LP filled the order at a better-than-market price.

Let's work through an explicit example. Suppose that ETH is trading at \$1,000 on the 0.3% fee pool and that we have infinite liquidity (so price impact is zero). Someone wants to buy ETH, so they send in 1,000 USDC and receive 0.997 ETH. The effective swap price here is $1000 / 0.997 \approx 1003$. Now, if we assume that the price of ETH remains at \$1,000, the markout PnL calculation immediately reflects the accrual of the LP fee: `protocol_buySell` is equal to -1, and `markout5m - swapPrice` is equal to -3, so we have a positive number multiplied by `eth_swapped`, hence positive PnL.

As such, it is *not* true that the original markout analysis does not include fees! It is in fact very convenient from an analysis standpoint that the fee is automatically incorporated into the effective swap execution price.

However, that sets the stage for a discussion of the three potential irregularities we identified.

Disagreement #1: The markout price should be the pool price

Let's take a look at how `markout5m` is defined:

```
LAST_VALUE(
CASE
WHEN t.token_b_symbol = 'USDC' then t.token_b_amount / t.token_a_amount
WHEN t.token_b_symbol = 'WETH' then t.token_a_amount / t.token_b_amount
END
) OVER (
ORDER BY
block_time asc range between current row
and interval '5 minutes' following
) as markout5m
```

Notice that this is very similar to the calculation of `swapPrice`! In plain language, it is asking: "For any given swap, take the last swap performed in the next 5 minutes, including that swap itself if no other swaps occur. Then, calculate the average execution price of that last swap, either `t.token_b_amount / t.token_a_amount` or `t.token_a_amount / t.token_b_amount` depending on the direction of the swap. That is the 5-minute markout price."

However, this is potentially quite different from the price of the pool after the last swap was made! To illustrate, consider the following example. Assume that ETH is trading at \$1,000.

- You make one swap, buying 0.00001 ETH, with zero price impact. The swap execution price and the pool price are both at 1 ETH : 1,000 USDC.
- In the next 5 minutes, there is only one additional swap. This swap buys 1,000,000 ETH, with a huge price impact. After this swap, the pool price is at 1 ETH : 1,100 USDC, \$100 higher than it was earlier. The swap's average execution price is 1 ETH : 1,050 USDC, which is a full \$50 below the pool price.

This is the same criticism that [Matteo Leibowitz pointed out](#):

**Matteo Leibowitz**

@teo_leibowitz

...

4/

Another issue with this query is that it doesn't factor price impact into the execution price—instead, it only takes the mid-price at the time the trade was executed.

Price impact is good for LPs because it means they are selling fewer assets at less favorable prices.

[ツイートを翻訳](#)

午後0:29 · 2022年11月1日 · Twitter Web App

For very long markout periods, this has a very small impact. For example, if you use a markout period of 1 year, the difference between the markout price and the swap price will be completely dominated by the overall fluctuation of ETH prices over that time period relative to the price impact of the last swap in that 1 year period. However, for small markout periods, like 5 minutes, this can cause quite a substantial change in the overall results.

Disagreement #2: There should be no exit slippage

Stepping back a little bit: In general, when you calculate markout PnL, you should not assume the existence of any exit "slippage" or "fee." Although markout as a metric looks at the profitability of individual trades, it is almost always applied in the context of a continuous trading *strategy*. In this setting, you are continuously making trades at a very rapid pace compared to your markout interval; for example, you might buy some inventory according to a buy signal alpha, then sell some inventory according to a sell signal alpha, and so on and so forth. You only exit at the end of a long trading period, and if you calculate markout PnL assuming an exit fee on every single trade, you will be vastly underestimating your strategy's profitability.

This is even more true for AMMs, where liquidity providers almost always provide liquidity over a continuous span of time (setting aside just-in-time liquidity, which accounts for a minority of trades, and is known to be highly profitable). However, recalling our previous discussion, we realize that:

- The calculation of `swapPrice` and `markout5m` both use the average (effective) swap price
- The calculation of `swapPrice` implicitly includes the swap fee

Therefore, *separate* from the problem that `markout5m` does not reflect the actual (infinitesimal) pool price, this means that estimates of markout PnL are downwardly biased by the inclusion of a swap fee in both the entry and exit transaction! If a given swap goes in the same direction as the last swap in its markout period (including itself), this means that the LP fee is exactly canceled out.

This is easily seen with an example. Suppose that there is no price impact, so that the average swap price and final pool price are always the same. If a given swap is the only swap in its markout period, we have `swapPrice = markout5m` for an overall PnL of exactly 0, which is certainly wrong as the swap fee is still captured by the LP. Of course, if the last swap goes in the opposite direction, then you capture twice the fee; however, the former case should be more common, because directionality of swaps is typically correlated, and because there are likely many intervals in which the markout period contains no additional swaps. As such, we suspect this leads to a systematic underestimation of markout PnL.

Revised calculation of markout PnL

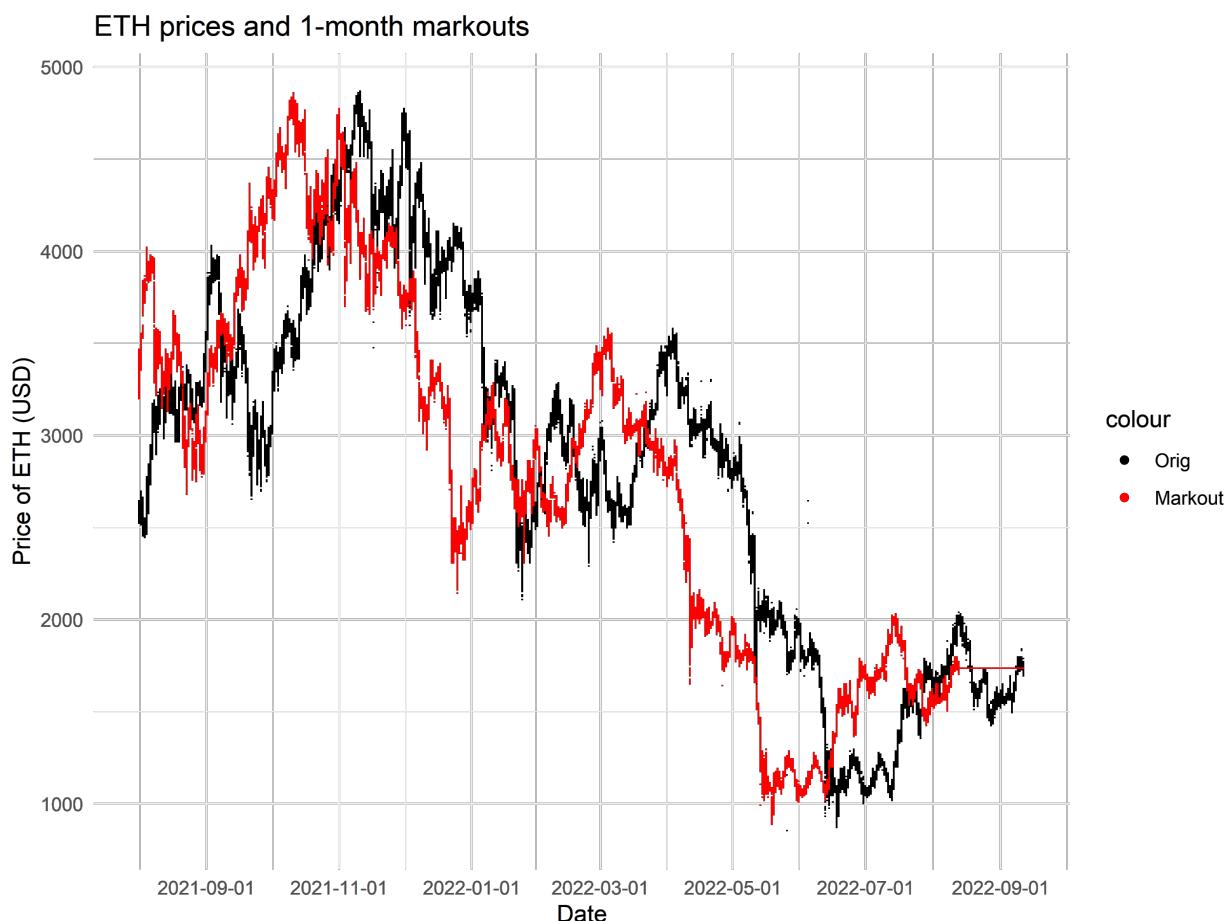
We wondered: Could the two disagreements identified above account for some of the discrepancies between our previous analysis and the results obtained by [@thiccyyhot_](#)? In order to resolve this, we re-analyzed the Uniswap V3 swap data.

In particular, we looked at swap data directly retrieved from Uniswap V3 event logs for the 0.05%, 0.3%, and 1% ETH/USDC fee tiers. We matched the analytic methodology used by [@thiccyyhot_](#) as follows:

- Restricted to swaps happening on or after August 1st, 2021 (Unix timestamp 1627776000) (Dune query: `t.block_time >= '2021-08-01'`)
- Restricted to swaps with nonzero amounts of USDC sent in or out
- Restricted to swaps with at least `1e-8` ETH sent in or out (Dune query: `eth_swapped > 1e-8`)
- Restricted to swaps with an average execution price between \$500 and \$5,000 per ETH (Dune query: `[swapPrice expression] BETWEEN 500 AND 5000`)

In order to determine the liquidity pool's final price after any given swap, we used the `tick` variable emitted by the `Swap` event. The [Uniswap V3 documentation](#) defines this as "the log base 1.0001 of price of the pool after the swap," which is exactly what we want. Note that when multiple swaps occur in the same Ethereum block, they all have the same timestamp; for ease of data analysis, we dealt with this by associating each timestamp with the mean ETH price calculated from each swap's emitted `tick` value, weighted by the amount of USDC transferred in or out. We expect this to be a relatively small, non-biasing factor in the analysis.

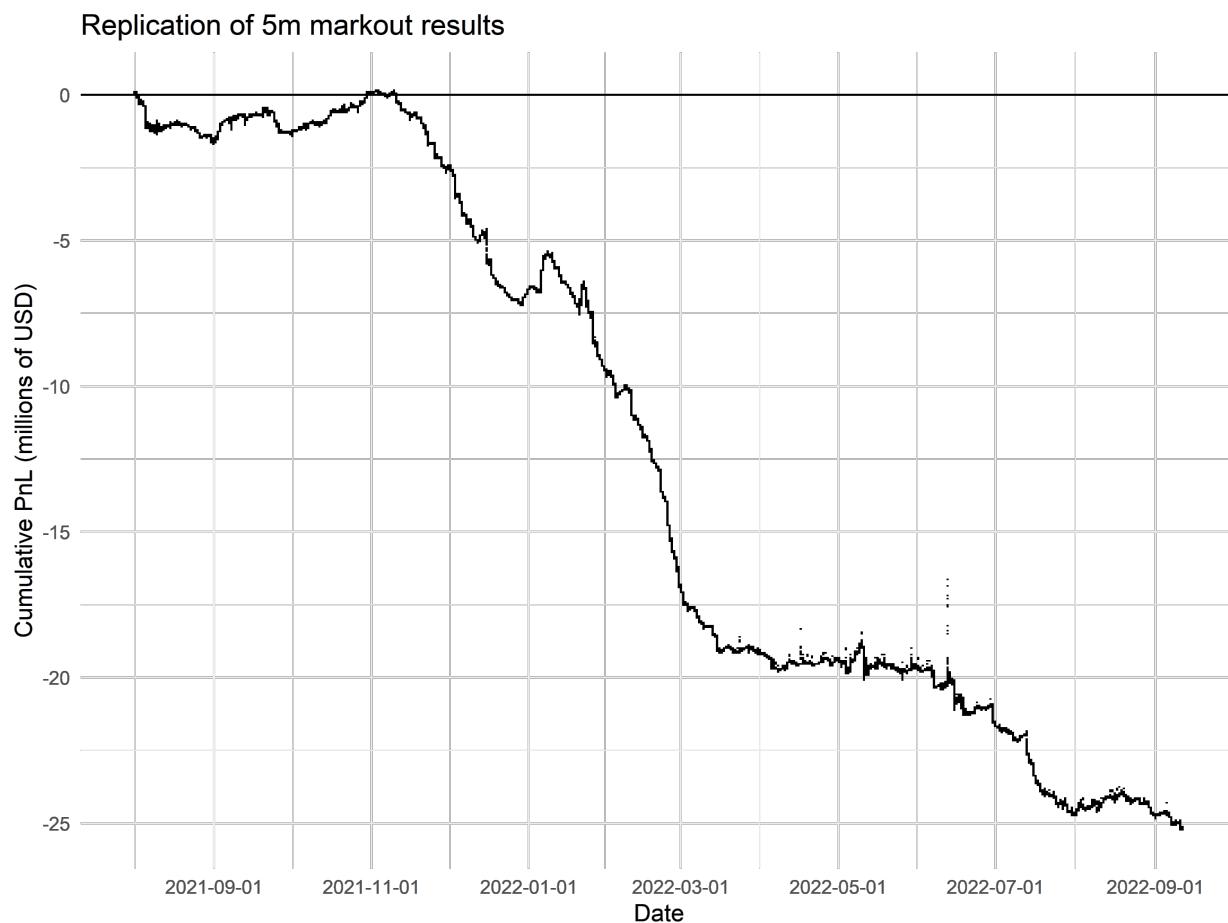
First, we checked for basic data consistency by plotting ETH prices as well as a 1-month markout price after every trade:



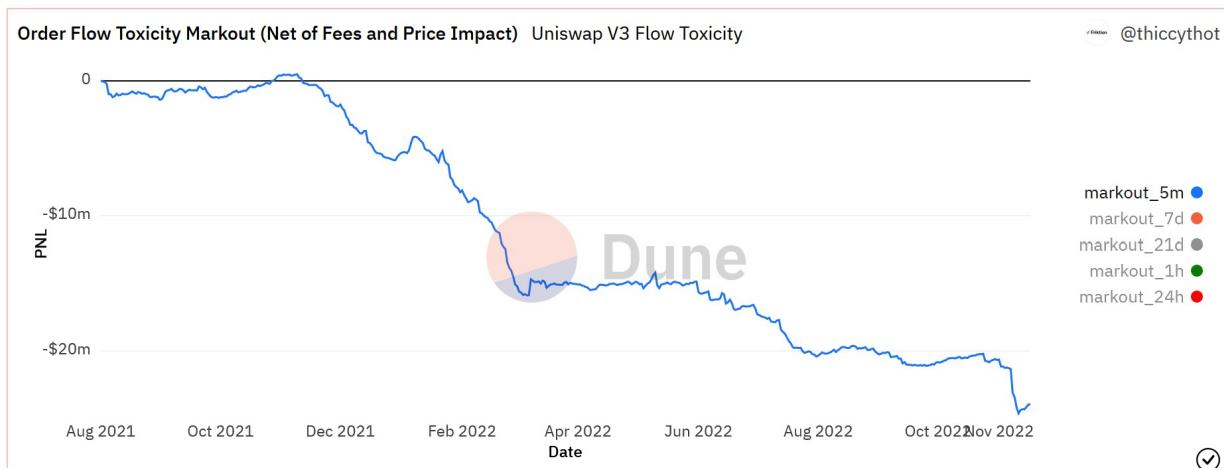
It is straightforward to verify that (1) the black line accurately tracks the price of ETH over time and (2) the red line, representing the 1-month markout price, is indeed equal to the black line shifted left by a 1-month period.

Next, we verified that we can reproduce [@thiccyyhot](#)'s results using the exact same methodology, *i.e.*, calculating markout using the average execution price (includng fees) of the last swap in the next N minutes.

Doing so with a markout period of 5 minutes, we calculate a cumulative PnL of approximately ~\$25 million over the period under consideration:

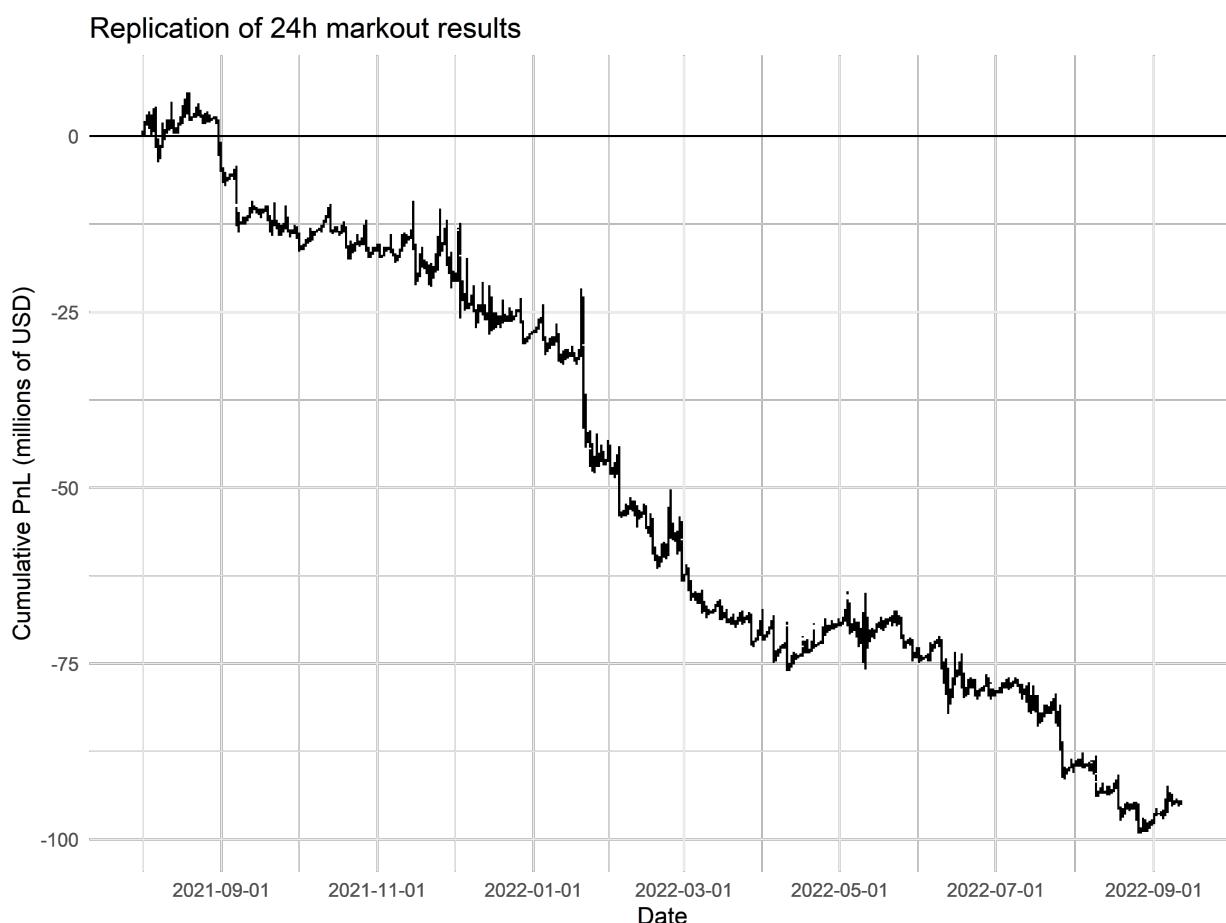


This is qualitatively very similar to the graph of cumulative 5-minute markout PnL given in the [original Dune query](#)

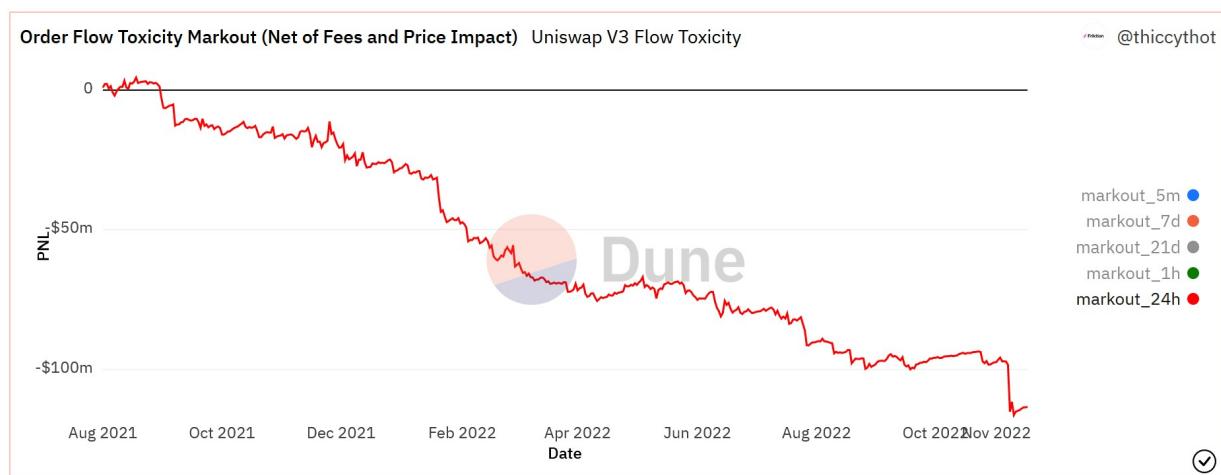


They are not *exactly* identical, and the Dune query includes approximately one more month of swap data, but they match closely enough that it is reasonable, in our opinion, to claim that we have faithfully reproduced the original analysis.

We also replicated the calculation of cumulative PnL using a 24-hour markout interval. In our analysis, we find that losses approach ~\$100 million:

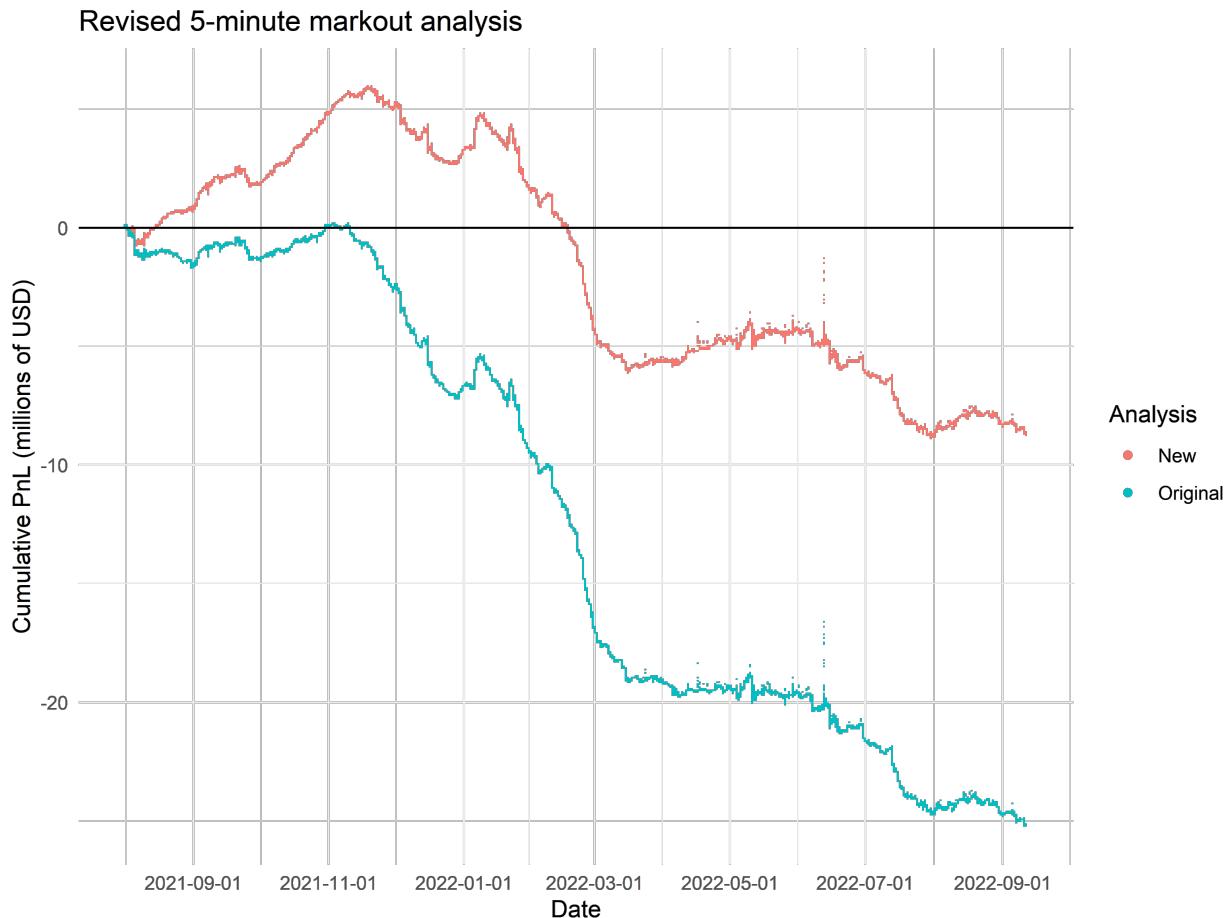


This is again very similar to the results of the original Dune query, again with the exception that their dataset includes approximately 1 more month's worth of data than ours:



We believe that these comparisons indicate that (1) we are using essentially the same data and (2) we are capable of analyzing the data in essentially the same way, even though the date source and analysis methodologies are not exactly the same (e.g., we are performing our analysis in R and not using the Dune `dex.trades` table as the source of swap data).

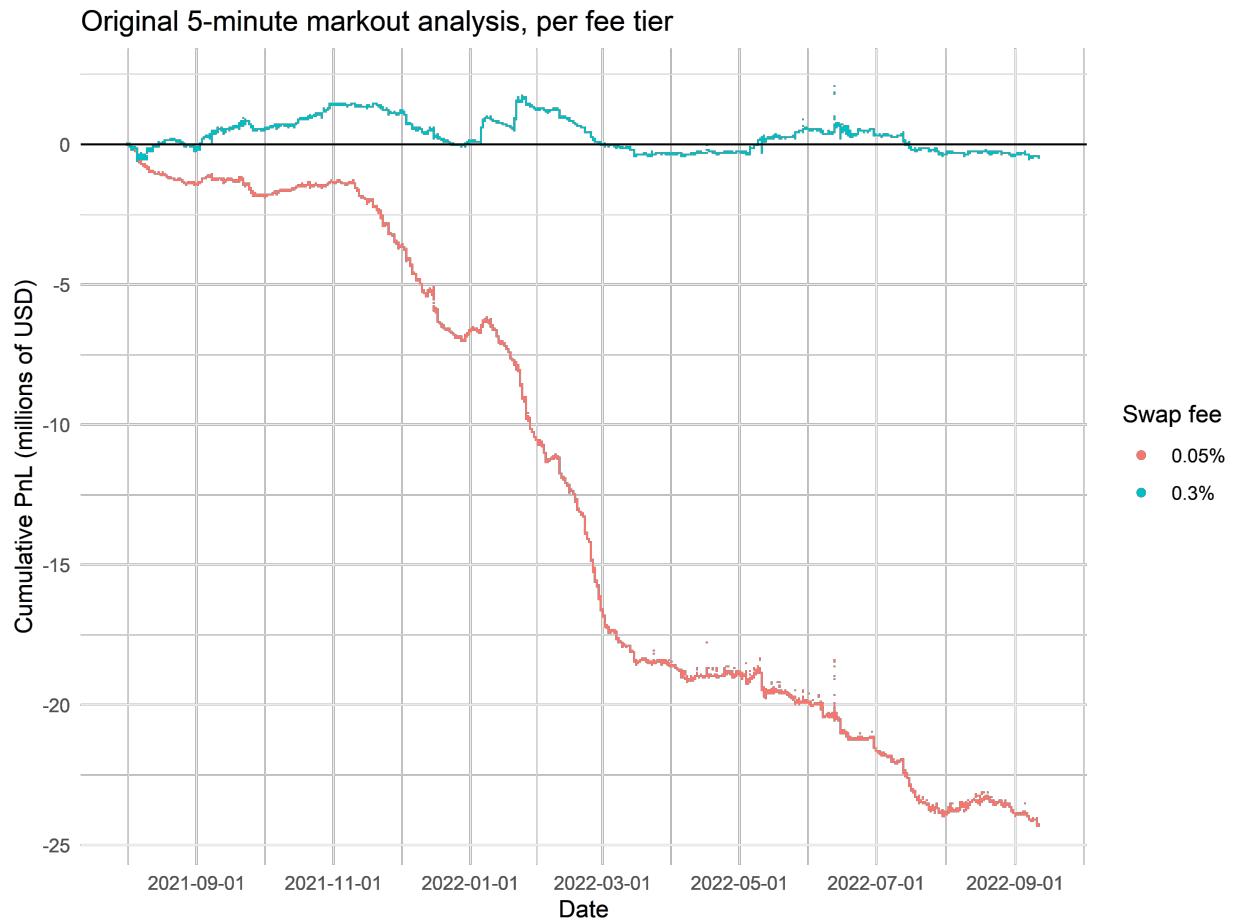
What happens if we make a single change: Instead of calculating the markout price using the average swap execution price of the last swap in each markout period, we use instead the price of the pool after that final swap? In theory, this should resolve both of the disagreements pointed out in the previous section. When we recalculate the cumulative PnL of all ETH/USDC LPs, we find that the results are *substantially* different:



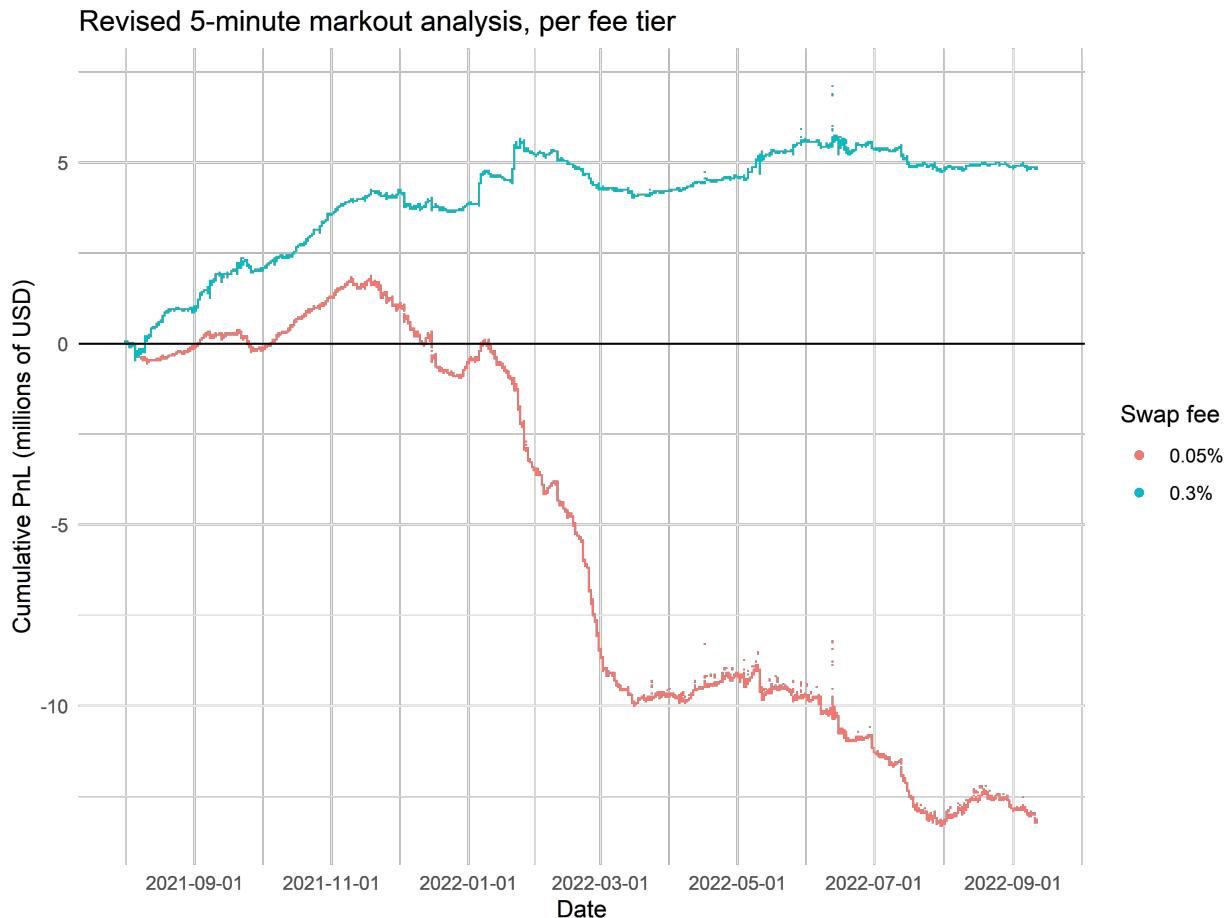
With the revised analysis, the losses at the end of the analysis period reach about -\$7m, nearly 75% lower than the figure of \$25m losses found in the original 5-minute markout analysis. Additionally, there is a long, multi-month period of profitability in the first half of the data! As we suspected, this seemingly small adjustment makes quite a large difference.

What if we look at performance of each fee tier separately? For simplicity, we will consider the 0.05% and 0.3% fee tiers. (The 1% fee tier receives a very small proportion of the swap volume, and in past analyses we have found that there are strange artifacts in the swap data, so we feel it is more straightforward to consider the 0.05% and 0.3% tiers alone.)

Using the original methodology, we see that the 0.05% pool is very unprofitable, and that the 0.3% pool tier is essentially flat throughout the analysis period. (That being said, it should be noted that the 0.05% pool has larger TVL and also receives the majority of ETH/USDC swap volume.)



Using the revised methodology, however, we find that the 0.05% pool's losses are significantly lesser, and that the 0.3% liquidity pool is actually net profitable throughout the analysis period:



These are roughly consistent with the results we obtained in a prior blog post! They are not identical, but that is only to be expected—the exact details of the analyses performed differ in many ways. What is important is that a small change made to the calculation of the markout price yields results which (1) corroborate previous analyses and (2) clearly indicate that it can be profitable to supply liquidity to certain fee tiers.

Longer markout periods

One might ask: What happens when we apply this revision to the 24-hour markout period, for which the original analysis demonstrates losses around ~\$100 million? Some have [argued](#), in fact, that a longer markout period more appropriately captures organic LP exposure, in the sense that if someone supplies liquidity to an ETH/USDC pool and the price of ETH is the same at the starting and ending points, they have strictly profited from fees, a result which is not reproduced if markout is calculated at intermediate points along the way.

We argue that a shorter markout period is in fact preferable. To be specific, we argue that the optimal markout period is one which captures as much *microstructural* properties as possible but as little *asset price drift* as possible.

For example, suppose that we use 1-month markouts for the price of ETH. Currently, the price of ETH is around \$1,200, and six months ago, the price of ETH was around \$3,400. That means that in those six months, the price of ETH has gone down nearly 3x!

It follows directly from the decrease in the price of ETH that overall, there was more selling activity than buying activity. Conversely, the liquidity pool bought more ETH on the way down than it sold (assuming liquidity remains roughly constant). Therefore, if we use 1-month markouts, it is very likely that in any given month, cumulative PnL is negative, because (1) the liquidity pool was a net buyer of ETH and (2) the price of ETH went down.

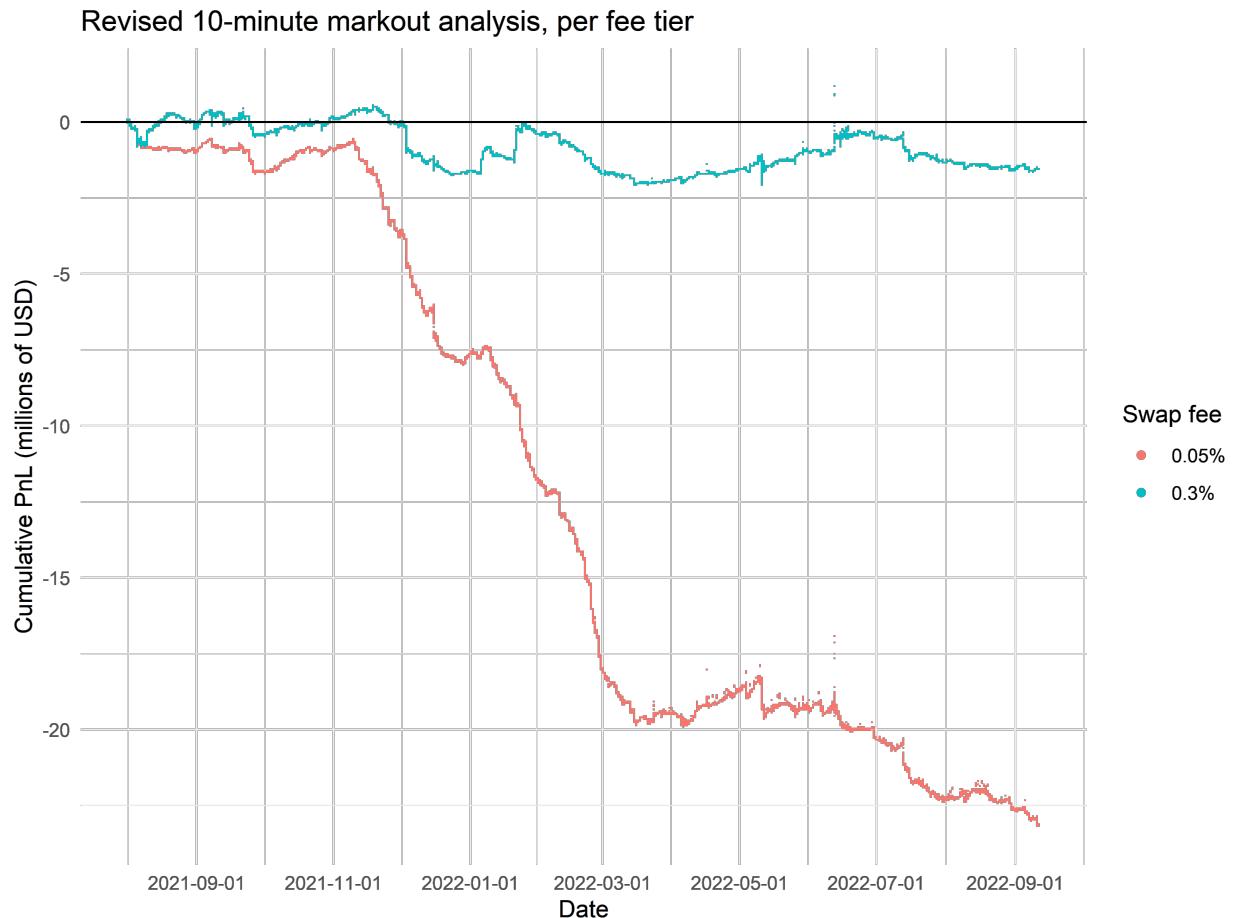
This thought experiment demonstrates exactly why we should not use 1-month or, arguably, even 24-hour markouts. Doing so means that the calculated PnL (with markouts) is dominated by exposure to the *drift* of the price of ETH rather than the balance of accrued fees vs. exposure to toxic flow! If we accept very long markout periods, we may as well simply plot the square root of the price of ETH rather than doing any markout analysis at all, given that the portfolio value for a supplier of ambient ETH/USDC liquidity scales *exactly* with that function.

It is arguably true that many suppliers of ETH/USDC liquidity do not hedge their deltas; however, that does not mean that there is any sense in doing a markout analysis in which we implicitly allow for the accumulation of unhedged delta exposure via long markout periods! *From a market-making standpoint*, the analysis that makes sense is one with a short markout period. Of course, our markout period cannot be *too* short. In the extreme case where we have a markout period of zero, and simply mark each trade to the infinitesimal price after that trade, PnL is generally extremely high, but obviously, this does not properly account for the toxicity of flow in the short term.

Our theoretical analysis still holds for longer markout periods. However, the revised methodology more or less converges to the original methodology as the markout period grows arbitrarily long, for the reason that unhedged delta exposure dominates PnL relative to the points of disagreement we previously outlined.

What is the correct markout period to choose? It is difficult to say for certain. However, for ETH/USDC, one of the most liquid and competitively traded markets in crypto, we believe that 5 minutes is sufficiently long to capture all microstructural features.

Suppose that we extend the markout period to 10 minutes. (One can very roughly argue that this corresponds approximately to a situation where a liquidity provider rehedges their deltas every 20 minutes.) We find substantial degradation in the cumulative PnL; for example, the net profitability of the 0.3% fee tier hovers at around zero even in the revised analysis:



That being said, it is important to note that remaining in a static fee tier is quite inefficient. As we have [pointed out](#) in prior work, a dynamic fee that toggles between different fee tiers based on statistical signals could help achieve substantial outperformance.