Brent Nakashima
Professor Pierce
CSC 369

Week 5: An Analysis of User Pixel Placement Trends and Behaviors

Background

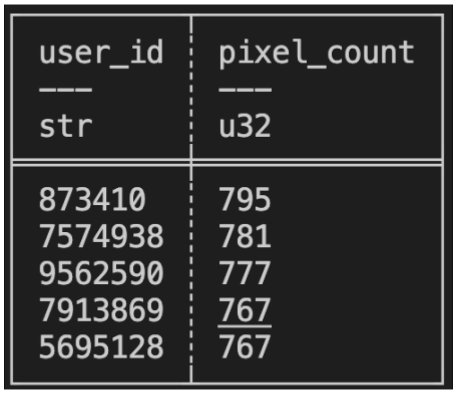
The purpose of the r/place pixel war was to compete or work with other users to establish pieces of art on the canvas. Often times, there would be different communities fighting on the same piece of art (for example One Piece Jolly Roger and Sans). This led me to want to examine how users in the r/place pixel war behaved regarding how they placed their pixels.

Assumptions

The initial assumption was that users would be interested in painting the exact same pixel repeatedly every few minutes. This would create a divide and conquer strategy within communities, where users only needed to focus on a small number of pixels to maintain control of the canvas and change the pixel back when needed. Allocating a few users to each small section of the artwork would theoretically allow for full coverage of the entire piece.

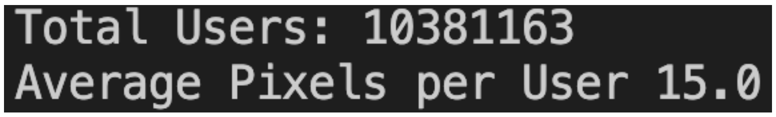
Painting the same pixel also touches on a piece of human psychology, where humans often form habits from repeat actions and enjoy the consistency of performing an action repeatedly.

Analysis



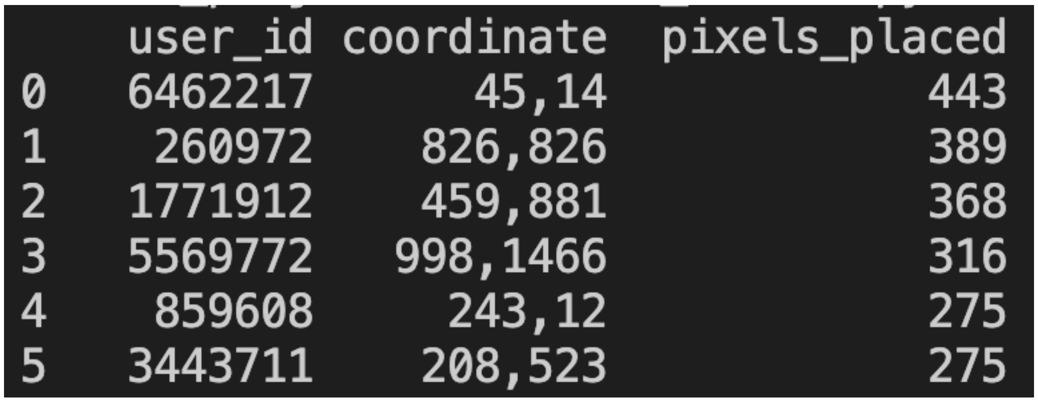
```
┌──────────┬─────────────┐
│ user_id  ┆ pixel_count │
│ ---      ┆ ---         │
│ str      ┆ u32         │
╞══════════╪═════════════╡
│ 873410   ┆ 795         │
│ 7574938  ┆ 781         │
│ 9562590  ┆ 777         │
│ 7913869  ┆ 767         │
│ 5695128  ┆ 767         │
└──────────┴─────────────┘
```

At first, I calculated how many pixels each user placed and sorted by pixel_count. However, only looking at the top 5 users may not be an accurate representation of most users, so I decided to look at the mean.



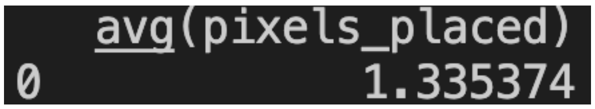
```
Total Users: 10381163
Average Pixels per User 15.0
```

I found out that on average, each user only placed 15 pixels throughout the entire event. This was surprising because the event lasted 3.5 days, giving plenty of time for users to place pixels. My theory here was that users were more interested in claiming they helped out their communities with a few pixels, and did not have the attention span to wait for the cooldown time in between pixels. 5 minutes is a long time to wait to place the next pixel, so most users probably paused after placing the first initial few.



```
   user_id coordinate  pixels_placed
0  6462217      45,14            443
1   260972    826,826            389
2  1771912    459,881            368
3  5569772   998,1466            316
4   859608     243,12            275
5  3443711    208,523            275
```

The next thing I did was try and identify if users placed their average 15 pixels in the same place. The image above captures the top 5 users who placed the most pixels in the same spot. For example, the first user placed 443 pixels at location (45, 14). However, these top users placed a lot more pixels than the average user would have placed given the average was 15.



```
   avg(pixels_placed)
0            1.335374
```

On average, a user placed about 1.3 pixels in the same coordinate. This goes against my initial assumption, where it seems that actually most users placed pixels on different coordinates.

Users wanting to place pixels in different spots touches on another aspect of people in general, where humans do not like to be bored. This brings up another issue where did users place their pixels in a centralized area if they were focused on helping out their community.

Conclusion

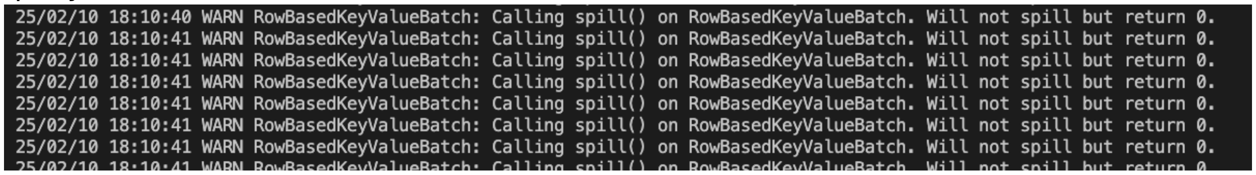
In the r/place 2022 pixel war, user behavior was not centered around repetition and users often did not place their pixels in the same location. These were my additional conclusions as a result from analysis:

- Typically, a user did not place that many pixels throughout the event. This was because human attention span is extremely limited, and there was a significant cooldown of 5 minutes in between pixel placement.
- User pixel placement was skewed right, with most users placing their amount of pixels towards the lower end of 15 pixels compared to the top user of 795 placed pixels.

DuckDB vs. PySpark

Initial Approach

Since both libraries have the ability to run SQL queries, I used the same query for both. The main difference was that in PySpark I had to create a spark session and then a temporary view table from the parquet file to use for my query. I used a sample of the data to verify that my queries ran in both correctly and can confirm they worked. However, when running the PySpark query on the full Parquet file, I kept getting a memory leak error and my query did not work.



```
25/02/10 18:10:40 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
25/02/10 18:10:41 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
25/02/10 18:10:41 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
25/02/10 18:10:41 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
25/02/10 18:10:41 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
25/02/10 18:10:41 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
25/02/10 18:10:41 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
25/02/10 18:10:41 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
```

Optimization

In order to solve this problem, I first tried applying predicate pushdown to my data frame, only selecting the ‘user_id’ and ‘coordinate’ columns. I could not apply any filtering because I had to use the whole dataset, and taking only 1 days’ worth of data did not seem like an appropriate representation for the entire thing.

Scaling

PySpark is used specifically when data analysis needs to be scaled horizontally. Unfortunately, I don’t have very many options as I am just working on my local laptop, which is why I might be running into the spill issue. I do not have means to increase scaling on my laptop, and do not want to create an instance on GCP or AWS for this assignment. I have learned that you should only use PySpark when you really need to, and other options (DuckDB, Polars, etc.) can be a much cheaper and simpler method for businesses to answer data-driven questions.