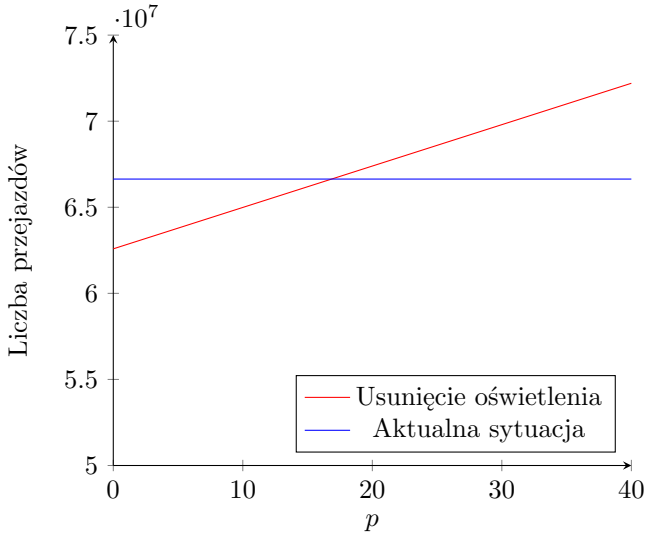
Oświetlenie miejskich rowerów

Szymon Pajzert

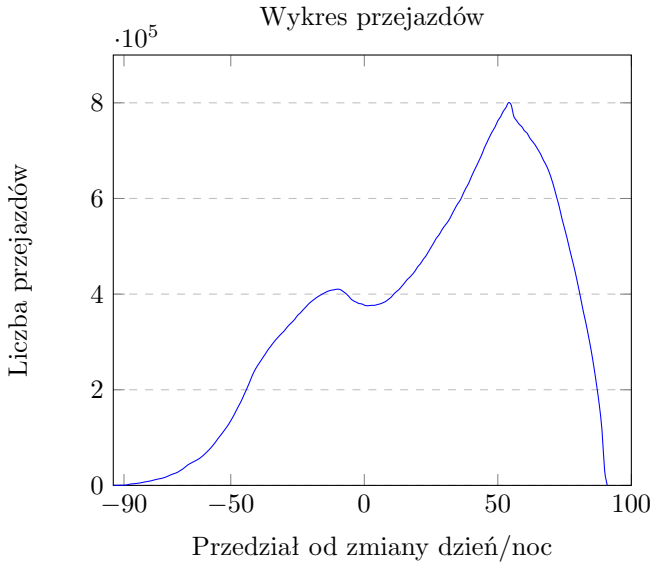
Z agregacji danych otrzymaliśmy, że w dzień odbyło się 48,142,817 a w nocy 18,495,221 początków lub końców przejazdów. Zakładając że ułamek p (z przedziału 0 do 1) przejazdów nadal będzie możliwych oraz zakładając globalny przyrost uczestników programu o 30% otrzymamy następujące wyniki. Okazuje się, że operacja ta przyniesie nam pozytywne skutki tylko gdy przynajmniej 20% przejazdów będzie mogło się odbyć po ciemku. Otrzymamy 10% zmianę dopiero przy 30%. Jeśli więc naszym priorytetem jest zwiększenie przejazdów, usunięcie oświetlenia raczej pozytywnie na ten aspekt nie wpłynie.



Poniżej znajduje się dokładniejszy opis mojej metodologi oraz użytych narzędzi.

Metodologia

W obliczeniach każdy przejazd jest liczony jako dwa przejazdy w jednym momencie - jego początek i koniec. Przejazdy są później akumulowane w pięciominutowe interwały, gdzie interwał 0 to pierwsze przejazdy odbywające się po zachodzie słońca lub ostatnie przed. Dajsze ujemne numery oznaczają numery w nocy, a dodatnie w dzień. Poniższy wykres dodatkowo wizualizuje udział tych interwałów w przejazdach. Do obliczenia wyników użyłem chmury BigQuery i dostępnej na niej możliwości dokonywania zapytań SQL.



Zapytania SQL

Do dokonania zapytań SQL, użyłem modelu ziemi jako kuli krążącej po okręgu wokół słońca. Do obliczeń użyłem modelu sunrise equation.

```
#standardSQL
-- Number of days in a year
CREATE TEMPORARY FUNCTION days(d DATE)
RETURNS INT64 AS (
    CASE
    WHEN
        (MOD(EXTRACT(YEAR FROM d),4) = 0
            AND MOD((EXTRACT(YEAR FROM d)),100) != 0)
        OR MOD((EXTRACT(YEAR FROM d)),400) = 0
        THEN 365
    ELSE 366
    END
);

-- Convert angle to radians
CREATE TEMPORARY FUNCTION rad(ang FLOAT64)
RETURNS FLOAT64 AS (3.141592 * ang / 180);

-- Calculate declination of the sun
CREATE TEMPORARY FUNCTION delta(d DATE)
RETURNS FLOAT64 AS (
    rad(-23.45)
    * cos(rad((360 / days(d))
    * (EXTRACT(DAYOFYEAR FROM d) + 10)))
);

-- Returns latitude of the New York
CREATE TEMPORARY FUNCTION psi()
RETURNS FLOAT64 AS (rad(40.748433));

-- Calculates hour angle of the earth from sun eguation
CREATE TEMPORARY FUNCTION sun_equation(d DATE)
RETURNS FLOAT64 AS (ACOS(- TAN(psi()) * TAN(delta(d))));

-- Converts time of the day to float in range from [1, to 24)
CREATE TEMPORARY FUNCTION time_to_float(t TIME)
RETURNS FLOAT64 AS (
    EXTRACT(HOUR FROM t)
    + EXTRACT(MINUTE FROM t) / 60 + EXTRACT(SECOND FROM t) / 3600
);
```

```
-- Calculates noon time due to daylight saving
CREATE TEMPORARY FUNCTION noon(d DATE)
RETURNS FLOAT64 AS (
    CASE
        WHEN EXTRACT(MONTH FROM d) < 3 OR EXTRACT(MONTH FROM d) > 10 THEN 12
        ELSE 13
    END
);

-- Calculate length of half of a day
CREATE TEMPORARY FUNCTION halfday(d DATE)
RETURNS FLOAT64 AS (sun_equation(d) * 12 / 3.1415);

-- Calculate time of sunrise
CREATE TEMPORARY FUNCTION sunrise(d DATE)
RETURNS FLOAT64 AS (noon(d) - halfday(d));

-- Calculate time of sunset
CREATE TEMPORARY FUNCTION sunset(d DATE)
RETURNS FLOAT64 AS (noon(d) + halfday(d));

-- Return smallest distance of t from a and b,
-- positive when t in (a, b)
-- negative otherwise
CREATE TEMPORARY FUNCTION closest(t FLOAT64, a FLOAT64, b FLOAT64)
RETURNS FLOAT64 AS (
    CASE
        WHEN t < a THEN t - a
        WHEN t > b THEN b - t
        ELSE
            (CASE WHEN t - a < b - t THEN t - a ELSE b - t END)
    END
);
```

Oto pod zapytania użyte do dokonania akumulacji wyników.

```
WITH
-- Extract times of start and stop
    times AS (
    SELECT * FROM (
        SELECT DATE(starttime) as d, starttime as timestamp,
            time_to_float(TIME(starttime, "America/New_York")) as time
        FROM `bigquery-public-data.new_york.citibike_trips`)
    UNION ALL (
        SELECT DATE(starttime) as d, stoptime as timestamp,
            time_to_float(TIME(stoptime, "America/New_York")) as time
        FROM `bigquery-public-data.new_york.citibike_trips`)),

-- Extract offset as number of five minutes intervals from sunset/sunrise of given start/sto
    data AS (
    SELECT d, timestamp, time,
        CAST(TRUNC(12 * closest(time, sunrise(d), sunset(d))) AS INT64) as offset
    FROM times),

-- Aggregate offsets
    offsets AS (
    SELECT count(*) as number, offset
    FROM data
    GROUP by offset),

-- Count given interval as daily or nightly
    gradation AS (
    SELECT offset,
        CAST(offset > 0 AS INT64) * number as day,
        CAST(offset <= 0 AS INT64) * number as night
    FROM offsets)

-- Accumulate results
SELECT sum(day), sum(night) from gradation;
```