Desafio CIVITAS

Filipe S P Prates
filipespprates@gmail.com

30 de junho de 2024

1 Introdução

Este artigo apresenta os métodos e resultados obtidos em resposta ao Desafio Civitas, demonstrando minhas habilidades em análise de dados. Utilizarei os dados da tabela 'rj-cetrio.desafio.readings_2024_06', que contém leituras de radares de trânsito do município do Rio de Janeiro.

Para mais informações, visite o repositório do desafio CIVITAS no github.

2 Objetivo

O objetivo do desafio é realizar uma análise exploratória dos dados, identificar inconsistências, além de identificar placas de veículos que foram possivelmente clonadas, usando as informações disponíveis.

3 Análise Exploratória dos Dados

3.1 Esquema dos Dados

Os dados fornecidos representam instâncias de capturas de radares de trânsito. Cada entrada na tabela contém informações referentes à uma captura: um veículo em uma posição em um momento do tempo. As colunas disponíveis estão descritas na Tabela 2.

3.2 Informações Gerais dos Dados

O dataset disponibilizado contém informações sobre 36.358.536 capturas de radares. Realizadas entre "2024-06-06 00:00:00 UTC"(5 de Junho de 2024, 9 horas da noite no Rio de Janeiro), até "2024-06-13 14:31:56 UTC"(13 de Junho de 2024, 11 horas e 31 minutos da manhã no Rio de Janeiro).

3.3 Coluna Categóricas

Das colunas da tabela, *placa*, *empresa*, *tipoveiculo* e *camera_numero* são colunas categóricas, ou seja, possuem uma série de valores pré-fixados que podem receber. Nesta subseção analisamos para tais

Coluna	Tipo	Descrição
datahora	TIMESTAMP	Data e hora da detecção do radar
datahora_captura	TIMESTAMP	Data e hora do recebimento dos dados
placa	BYTES	Placa do veículo capturado
empresa	BYTES	Empresa do radar
tipoveiculo	BYTES	Tipo do veículo
velocidade	INTEGER	Velocidade do veículo
camera_numero	BYTES	Número identificador do radar
camera_latitude	FLOAT	Latitude do radar
camera_longitude	FLOAT	Longitude do radar

Tabela 1: Descrição das colunas da tabela.

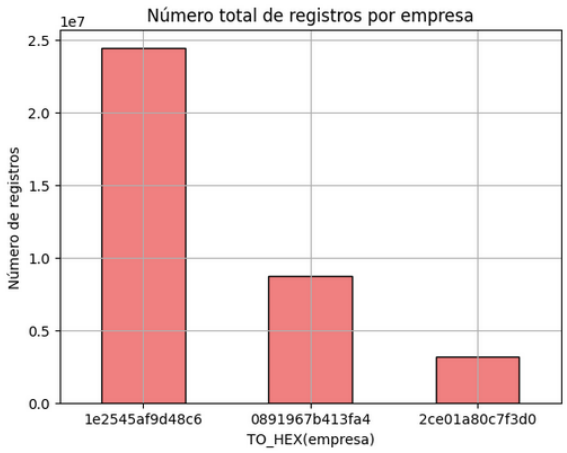

Figura 1: Número de registros por empresa responsável. A empresa 1e2545af9d48c6, com 24 milhões de registros, tem responsabilidade sobre aproximadamente 67% do Dataset. Essa distribuição é similar ao número de câmeras por empresa.

Coluna	Número de valores distintos
placa	7984610
empresa	3
tipoveiculo	4
camera_numero	1421

Tabela 2: Valores distintos de cada coluna categóricas.

colunas, quantos valores distintos para cada categoria existem, e como estão os dados distribuídos entre eles.

São representados no dataset 7.984.610 *placas* diferentes, com um total de 36.358.536 capturas, resultando em uma média de aproximadamente 4.55 capturas para cada placa.

Apenas 3 *empresas* são responsáveis por todos os registros de capturas no Dataset (Figura 1).

O sistema classifica os veículos refistrados em 4 *tipoveiculo*s diferentes, observando as velocidades médias de cada tipo de veículo, podemos propôr hipóteses de qual *tipoveiculo* refere a qual tipo de veículo real (Figura 2).

Utilizaremos à frente á fim de otimização de cáculos o fato de todos os 36 milhões de registros estarem agrupados geograficamente em apenas 1421 radares fixos.

```sql
SELECT COUNT(DISTINCT placa) AS unique_placas
FROM `rj-cetrio.desafio.readings_2024_06`;

SELECT COUNT(DISTINCT empresa) AS unique_empresas
FROM `rj-cetrio.desafio.readings_2024_06`;

SELECT COUNT(DISTINCT tipoveiculo) AS unique_tipoveiculo
FROM `rj-cetrio.desafio.readings_2024_06`;

SELECT COUNT(DISTINCT camera_numero) AS unique_radares
FROM `rj-cetrio.desafio.readings_2024_06`;
```

Listing 1: Query SQL para contar quantidade de valores distintos para as colunas categóricas do dataset

3.3.1 Número de registros por cada câmera de radar

Observamos na Figura 3 que a câmera de radar identificada por f6b6eff6c3c578 é a que mais registrou veículos no Dataset, com aproximadamente o dobro de registros da segunda com mais registros.

Ao analisar todas as câmeras exceto a f6b6eff6c3c578, percebemos uma distribuição similar à uma exponencial. Onde poucas câmeras são responsáveis pela maioria dos registros, e muitas câmeras

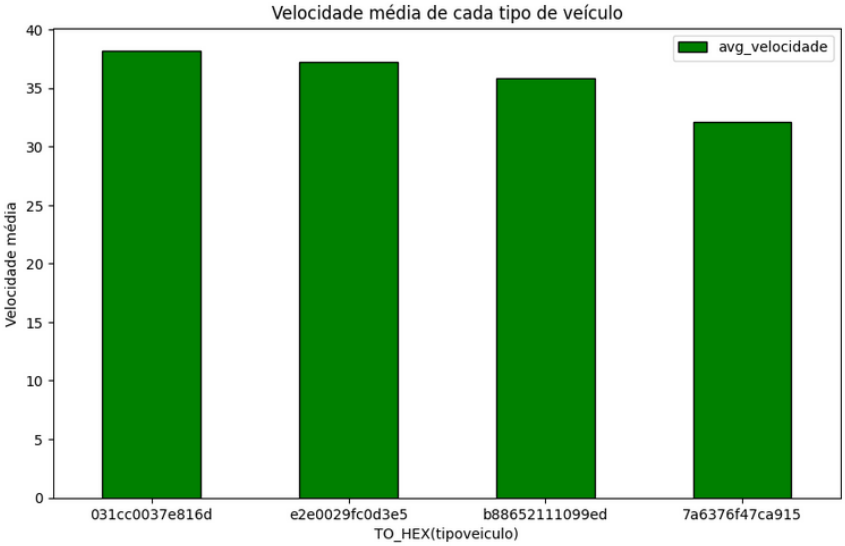

Figura 2: Velocidades médias de cada tipo de veículo. Podemos observar uma variação entre as velocidade médias dependendo do tipo de veículo. De 32.1 km/h para veículos do tipo 7a6376f47ca915, até 38.2km/h para veículos do tipo 031cc0037e816d. Os dados são consistentes em 7a6376f47ca915 representar veículos de carga como caminhões, enquanto 031cc0037e816d e e2e0029fc0d3e5 representam motos e veículos de passeio.

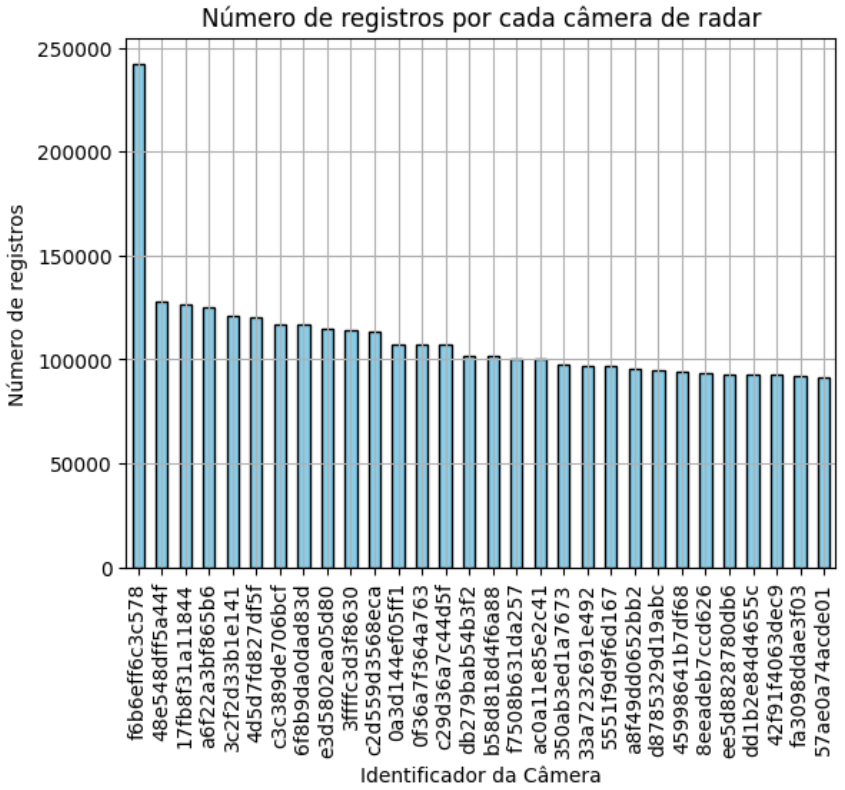

Figura 3: As 30 câmeras de radares com mais registros, e seu número de registros, do Município do Rio de Janeiro.

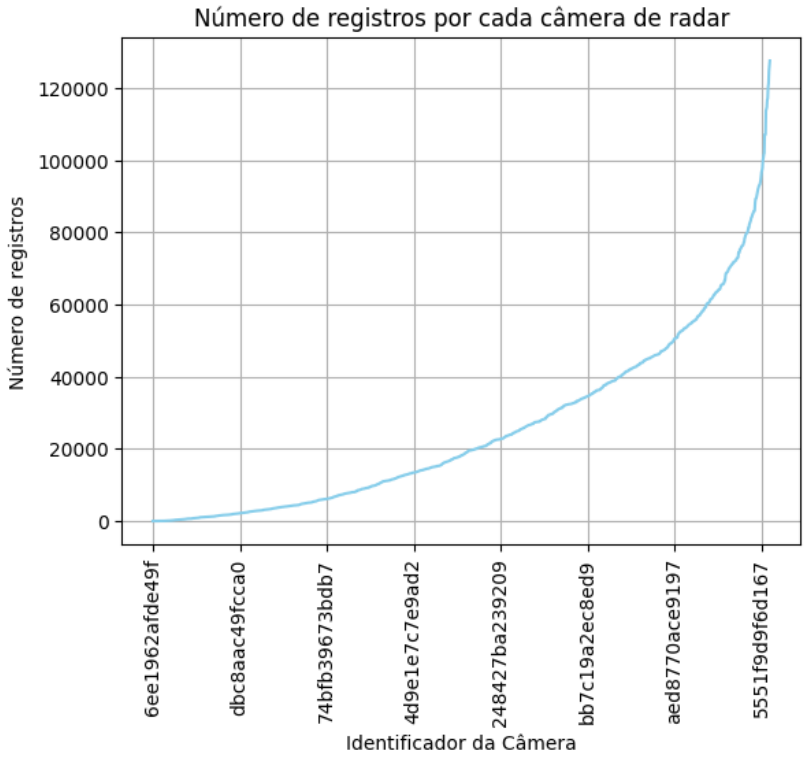

Figura 4: Número de registros por câmera de radares, ordenados da câmera com menos registros (6ee1962afde49f, 457ee59cde6d92 e bb059f56ccf5e3, com um registro cada) até a câmera com mais registros (48e548dff5a44f, já que foi excluida a outlier camera_numero f6b6eff6c3c578). Observamos um comportamento de regime exponencial - onde câmeras localizadas em vias com grande fluxo de carros geram uma parte muito maior dos registros se comparadas à radares em ruas menos frequentadas.

possuem poucos registros. Isto é consistente com uma rede complexa de ruas, onde algumas vias são muito utilizadas, e outras, mais capilares, pouco utilizadas. Figura 4.

3.4 Histograma de Velocidades

As velocidades dos veículos registradas no dataset foram consideradas consistentes com o Município do Rio de Janeiro, vide Figura 5.

3.5 Inconsistência nos dados de Latitude e Longitude

Algumas das câmeras de radares registram latitudes e longitudes inconsistentes. Por exemplo, no caso da *camera_numero* CKF78vkvCg==, todos os seus registros indicam *camera_latitude* e *camera_longitude* 0. Ao todo **259 casos** de câmeras que não registram latitudes e longitudes consistentes com o território do Município do Rio de Janeiro. Neste artigo tais registros serão desconsiderados, porém com acesso ao endereço de tais câmeras, seria possível corrigir os valores e então acrescentar mais registros para análise.

```
SELECT
    distinct camera_numero,
FROM
    `rj-cetrio.desafio.readings_2024_06`
WHERE
    camera_latitude >= -22.8
  OR camera_latitude <= -23.0
  OR camera_longitude <= -43.8
  OR camera_longitude >= -43.1
```

Listing 2: Query SQL para identificar se existem casos inconsistentes onde câmeras registram valores

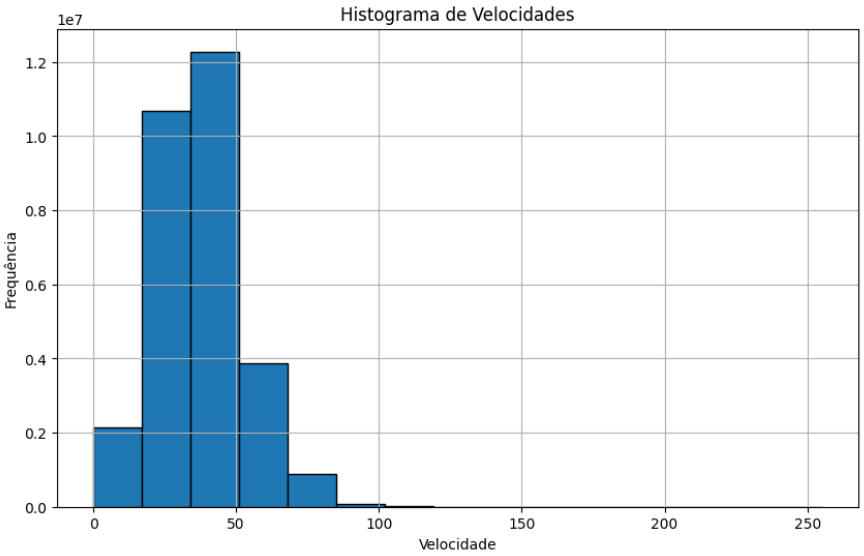

Figura 5: Histograma com as frequencias que as diferentes velocidades foram registradas pelo radares do Rio de Janeiro. Observamos uma distribuição normal em volta da média de aproximadamente 35km/h, consistente com ruas urbanas, com alguns outliers de até 255 km/h. Tais valores são altos, porém, devido à sua pequena frequência, considerados ainda consistentes com uma metrópole com vias expressas e milhões de habitantes.

de latitude e longitude fora do território do Município do Rio de Janeiro. Existem 259 casos de câmeras com problemas sistemáticos neste quesito.

Percebemos também, ao visualizar os dados que não se encontram dentro dos limites esperados, casos de sinal trocado, como latitude "22.9", o qual provavelmente podemos assumir que deveria ser o valor -22.9", dentro dos limites de latitude do município. Tais dados não serão considerados neste estudo exploratório, porém podem ser tratados e incluidos futuramente.

```sql
SELECT
    camera_numero,
    COUNT(DISTINCT camera_latitude) AS unique_latitudes,
    COUNT(DISTINCT camera_longitude) AS unique_longitudes
FROM
    `rj-cetrio.desafio.readings_2024_06`
GROUP BY
    camera_numero
HAVING
    COUNT(DISTINCT camera_latitude) > 1 OR COUNT(DISTINCT camera_longitude) > 1;
```

Listing 3: Query SQL para identificar se existem casos inconsistentes onde registros feitos pela mesma câmera de radar apresentam valores diferentes de latitude e longitude entre si. Não existem.

3.6 Limpeza nos dados de Latitude e Longitude

Podemos observar nos dados alguns valores com latitude ou longitude muito distante do esperado para radares no município do Rio de Janeiro. Conseguimos através do Google Maps determinar que, em grosso modo, os limites do município se encontram nas latitudes (-22.773 S, -43.152 W) ponto extremo nordeste do Municipio, e (-23.061 S, -43.765 W) ponto extremo sudoeste do Município. Podemos então nos conter com os dados que se encaixam dentro desses limites inferiores e superiores.

Para diminuir o uso posterior de recursos da máquina, excluímos, diretamente na SQL, qualquer valor de latitude ou longitude fora do esperado, chegando assim a 29.949.144 instâncias de dados de captura de radares dentro das condições estabelecidas 4.

```sql
SELECT
  TO_HEX(placa) AS placa_hex,
```

```
  TO_HEX(empresa) AS empresa_hex,
  TO_HEX(tipoveiculo) AS tipoveiculo_hex,
  velocidade,
  TO_HEX(camera_numero) AS cameranumero_hex,
  camera_latitude,
  camera_longitude,
  datahora,
  datahora_captura
FROM `rj-cetrio.desafio.readings_2024_06`
WHERE camera_latitude < -22.8
  AND camera_latitude > -23.0
  AND camera_longitude > -43.8
  AND camera_longitude < -43.1
```

Listing 4: Query SQL para recuperar dados de radares dentro de um modelo retângular do Município do Rio de Janeiro

3.7 Visualizando os dados no Município

Podemos utilizar as colunas de latitude e longitude para criar um gráfico de dispersão, onde cada ponto indica uma localização da captura armazenada na tabela. Como 30 milhões de pontos excedem o limite para um gráfico de dispersão, iremos utilizar diferentes quantidades de dados, de 1000 entradas até 100.000 entradas, esperando que um mapa coerente com a geografia e as ruas do Município do Rio de Janeiro apareça cada vez mais claramente. Figura 8.

Sobrepondo alguns dos pontos em um mapa do município obtemos 9.

4 Placas Clonadas

4.1 Mesma placa em dois veículos de tipos diferentes

Uma maneira simples de detectar placas potencialmente clonadas é verificar se existem dois (ou mais) registros da mesma placa na tabela com valores de *tipoveiculo* distintos. Isso indica que uma mesma placa está sendo utilizada, ou, potencialmente, que houve algum erro no processo de detecção do *tipoveiculo* no momento da captura.

Para identificar este segundo caso, podemos analisar os identificadores dos radares, verificando assim se existe algum padrão que indique falhas sistemáticas em um ou mais dos radares, o que geraria falsos positivos.

```
SELECT placa FROM `rj-cetrio.desafio.readings_2024_06`
GROUP BY placa HAVING COUNT(DISTINCT tipoveiculo) > 1;
```

Listing 5: Query SQL para identificar inconsistências de placa com tipo de veículo. São retornadas todas as placas as quais existem dois (ou mais) registros na tabela com valores de *tipoveiculo* distintos.

Seguindo este critério, encontramos 441.652 placas inconsistentes.

4.2 Distância entre dois registros

Outra maneira de identificar placas clonadas é verificar a existência de dois registros da mesma placa, distantes um do outro, com uma pequena diferença de *datahora_captura*. Podemos assumir que, dada uma distância suficientemente grande e uma diferença entre *datahora_captura* (Δt) suficientemente pequena, é impossível que seja o mesmo veículo em ambos os locais.

4.2.1 Distância em linha reta

Dadas a latitude e longitude de dois registros, podemos calcular sua distância em kilômetros utilizando a fórmula de "haversine". Para melhor performance, calculamos diretamente na query SQL para apenas buscar os dados relevantes para análises subsequentes.

```
 6371 * ACOS(
      COS(RADIANS(t1.camera_latitude)) * COS(RADIANS(t2.camera_latitude)) *
      COS(RADIANS(t2.camera_longitude) - RADIANS(t1.camera_longitude)) +
```

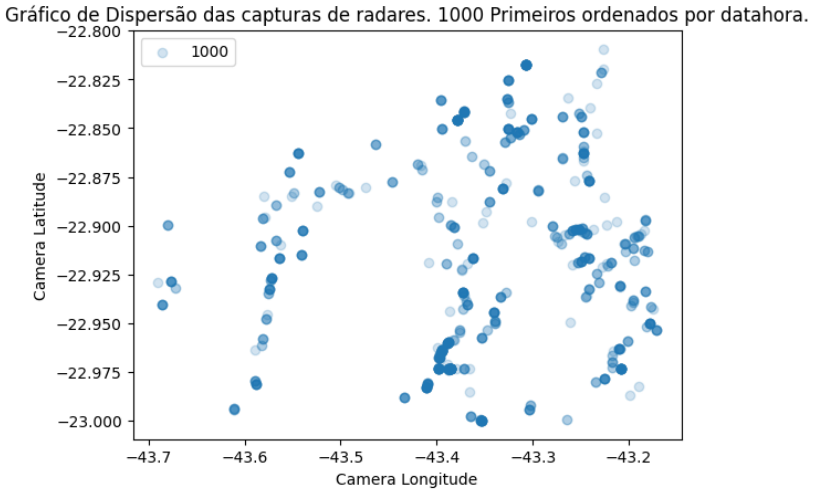

Figura 6: Gráfico de Dispersão das capturas de radares. 1000 Primeiras capturas ordenados por *datahora*.

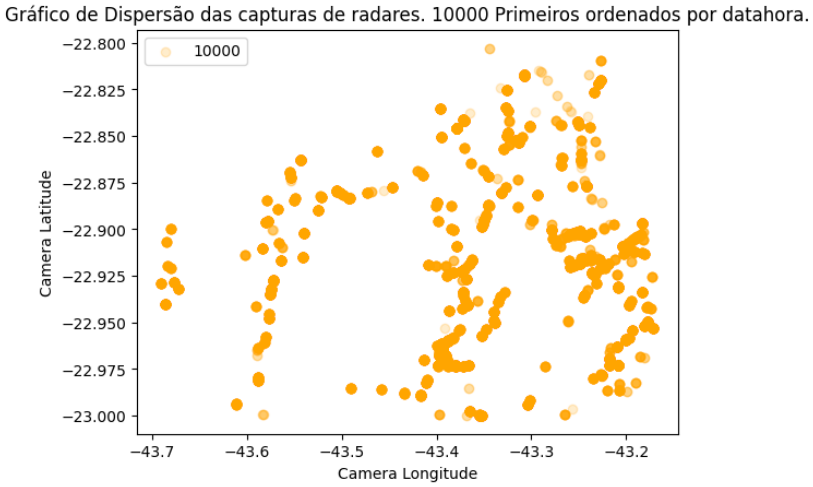

Figura 7: 10000 Primeiras capturas ordenados por data e hora.

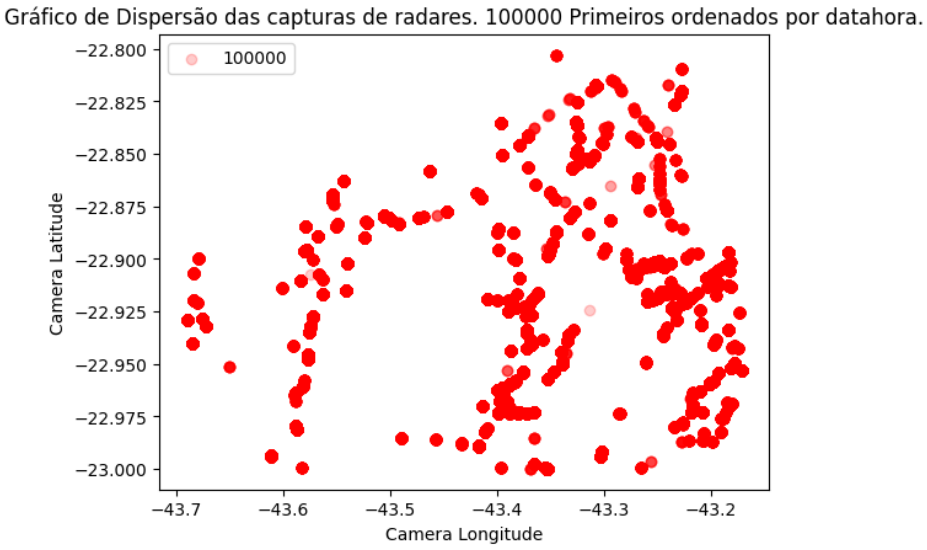

Figura 8: 100000 Primeiras capturas ordenados por data e hora. Podemos observar claramente a Zona Sul no canto inferior direito, a costa oeste da Baia de Guanabara, a Linha Amarela, a Barra na parte inferior, até a Estrada das Furnas/ Av. Édison Passos, cortando o Parque Nacional da Tijuca.

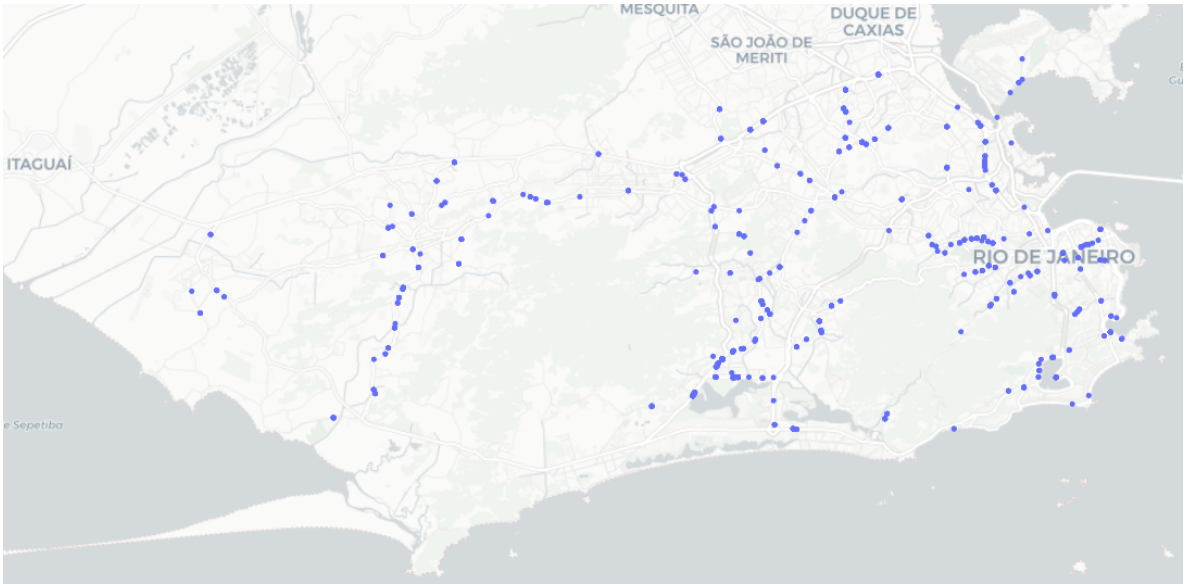

Figura 9: Dados das primeiras 1000 detecções disponíveis da tabela, sobrepostos no mapa do Município do Rio de Janeiro.

```
      SIN(RADIANS(t1.camera_latitude)) * SIN(RADIANS(t2.camera_latitude))
    ) AS distance_km
```

Listing 6: Fórmula de haversine para calcular distâncias em kilometros na superfície da Terra, através de latitudes e longitudes.

```
6371 * ACOS(LEAST(GREATEST(
      COS(t1.camera_latitude * 3.141592 / 180) * COS(t2.camera_latitude * 3.141592 /
          180) *
      COS(t2.camera_longitude * 3.141592 / 180 - t1.camera_longitude * 3.141592 / 180)
          +
      SIN(t1.camera_latitude * 3.141592 / 180) * SIN(t2.camera_latitude * 3.141592 /
          180),-1),1)) AS distance_km
```

Listing 7: Fórmula de haversine para calcular distâncias em kilometros na superfície da Terra, através de latitudes e longitudes. Sem acesso à função RADIANS() nem PI() e resolvendo erro de float point precision na função ACOS().

Com essa função de distância, podemos observar alguns casos de mesma placa com registros de distâncias grandes, em um curto período de tempo.

```
WITH distance_time_check AS (
  SELECT
    t1.placa,
    t1.datahora_captura AS datahora1,
    t2.datahora_captura AS datahora2,
    t1.camera_latitude AS lat1,
    t1.camera_longitude AS lon1,
    t2.camera_latitude AS lat2,
    t2.camera_longitude AS lon2,
    ABS(TIMESTAMP_DIFF(t1.datahora_captura, t2.datahora_captura, MINUTE)) AS
        time_diff_minutes,
    6371 * ACOS(LEAST(GREATEST(
      COS(t1.camera_latitude * 3.141592 / 180) * COS(t2.camera_latitude * 3.141592 /
          180) *
      COS(t2.camera_longitude * 3.141592 / 180 - t1.camera_longitude * 3.141592 / 180)
          +
      SIN(t1.camera_latitude * 3.141592 / 180) * SIN(t2.camera_latitude * 3.141592 /
          180),-1),1)) AS distance_km
  FROM
    `rj-cetrio.desafio.readings_2024_06` t1
```

```sql
  JOIN
    `rj-cetrio.desafio.readings_2024_06` t2
  ON
    t1.placa = t2.placa
    AND t1.datahora_captura < t2.datahora_captura
  WHERE t1.camera_latitude < -22.8 AND t2.camera_latitude < -22.8
  AND t1.camera_latitude > -23.0  AND t2.camera_latitude > -23.0
  AND t1.camera_longitude > -43.8  AND t2.camera_longitude > -43.8
  AND t1.camera_longitude < -43.1  AND t2.camera_longitude < -43.1
)
SELECT
  placa,
  datahora1,
  datahora2,
  lat1,
  lon1,
  lat2,
  lon2,
  time_diff_minutes,
  distance_km
FROM
  distance_time_check
WHERE
  time_diff_minutes <= 3 -- Threshold para diff em minutos
  AND distance_km >= 6   -- Threshold para diff em kilometros
ORDER BY
  placa, datahora1;
```

Listing 8: Exemplo de query SQL para detectar inconsistências e retornar casos de mesma placa com registros distantes um do outro, segundo a fórmula de haversine, mas com diferença de tempo entre eles de curta. Considerando inconsistente percorrer mais de 6km em menos de 3 minutos (120km/h de média). Como discutido no artigo, podemos substituir última condicional WHERE para "time_diff_minutes $\leq$ distance_km/5"para melhor resultados.

TO_HEX(placa)	lat1	long1	lat2	long2	Distância (km)	Δt (min)
AAEd0Z1N8R0nW6PjvFdY3uE=	-22.949	-43.177	-22.903	-43.245	8.684	2
AAFHNUra1M70d1Cw4bhU5VU=	-22.954	-43.193	-22.817	-43.307	19.215	0
AAAHLRM713ANhx0EEbTttcx8=	-22.880	-43.331	-22.835	-43.245	9.628	0
AAHtSwsIG2PODaQXtXb9gQs=	-22.940	-43.367	-22.994	-43.303	8.889	2
AAJbkMjCdI3+qr77FZOTSRo=	-22.936	-43.333	22.979	-43.588	26.583	3

Tabela 3: Tabela com alguns dos dados de retorno do SQL 8. É impossível que um mesmo veículo percorra distâncias como 26,3km em 3 minutos, ou até 19,2 km em 0 minuto. Logo são casos claros de falha nos radares e/ou placas clonadas.

Utilizando tal abordagem, com limites de "*time_diff_minutes* ≤ 3 E *distance_km* ≥ 6", conseguimos detectar 77222 casos de placas distintas com indícios de clonagem.

Podemos deixar mais genérica a condição de inconsistência, já que podem existir casos de registros que percorrem, por exemplo, 8km em 4min, a mesma velocidade incosistente, porém a SQL atual não os mostra. Para detectar todos os casos onde a distância e o Δt são potencialmente inconsistentes, podemos utilizar a condicional "*time_diff_minutes* $\leq$ *distance_km*/2". Assim capturamos todos os casos que acarretariam uma velocidade média de mais de 120km/h, num total de 369752 placas distintas suspeitas. Esse número elevado pode apontar um número significativo de falsos positivos.

Para garantir um número menor de falsos positivos, já que numa via expressa é possível alguns veículos alcaçarem tal velocidade, utilizaremos "*time_diff_minutes* $\leq$ *distance_km*/3", necessitando de 180km/h de média para percorrer a distância entre registros no tempo indicado. Utilizando tal condicional temos um resultado final de 344816 casos distintos de placas com registros inconsistentes.

Para finalizar, e garantir nenhum falso positivo, podemos utilizar uma condicional mais extrema como **"*time_diff_minutes* $\leq$ *distance_km*/5"**, só possível se o veículo mantivesse **300km/h**, numa linha reta entre os dois ponto, já que não é possível, identificamos então **322245 casos de placas clonadas** utilizando este critério.

4.2.2 Distância no trânsito

Para uma estimativa mais precisa do tempo esperado para percorrer uma distância entre dois pontos, o ideal seria considerar, ao invés de uma linha reta, a geografia e o urbanismo da Cidade. Identificando o endereço referente ao ponto geolocalizado (por latitude e longitude) e então utilizar bilbliotecas de estimativa de tempo mínimo no trânsito para percorrer a distância entre os dois registros. Assumir que se uma placa foi registrada nesses dois pontos em um tempo significativamente menor que tal estimativa, um erro de registro ou placa clonada existe.

Tal abordagem seria claramente mais custosa, porém como os registros são fixos nos radares, e temos apenas 1421 radares únicos, podemos calcular a distância par-a-par entre os radares (2017820 valores), que podem ser utilizados para estimar o Δt mínimo para todos os pares de registros de cada *placa*.

Para uma precisão ainda melhor, podemos utilizar a datahora de captura e os dados de trânsito do momento da caputura, porém não poderíamos mais aproveitar do agrupamento dos mais de 30 milhões de valores nos 1421 radares, requerendo novamente executar um chamado a biblioteca de estimativa de tempo no trânsito para cada par de registros suspeitos, o que poderia ser computacionalmente inviável.

5 Ferramentas utilizadas

Para o desenvolvimento da análise documentada neste artigo, foram utilizadas as seguintes ferramentas:

1. Google BigQuery para acesso à tabela "rj-cetrio.desafio.readings_2024_06", execução de queries SQL, e execução de notebook Python conectado à tabela,
2. Python e Bibliotecas de análise de dados - pandas para analisar DataFrames com grande quantidade de dados, matplotlib e plotly para análises gráficas,
3. Overleaf - Editor LaTeX, para documentar algumas das SQL usadas e tabelar valores, além de facilitar a organização e apresentação do trabalho,
4. Excalidraw, para organização de ideias relativas ao desafio (Figura 10).

6 Próximos Passos

Nesta análise assumimos uma série de simplificações, tais como:

1. Desconsiderar totalmente valores de latitude/longitude fora de um modelo retângular do Rio de Janeiro.
2. Distância em linha reta entre pontos geolocalizados para estimativa Δt mínimo,
3. Considerar todas as câmera de radares que registram latitudes e longitudes corretas como confiáveis.

Como proximos passos, para refinar os resultados, poderíamos:

1. Verificar a possibilidade de corrigir valores de latitude/longitude incosistentes: Por exemplo, populando pelo valor de latitude/longitude de outro registro de mesmo "camera_numero", ou obter de fonte confiável os valores de cada um dos 1421 radares, desta forma obtendo 20% mais registros para análise.
2. Obter uma distância entre dois registros seguindo as ruas e tempos históricos de trânsito no Município: Por exemplo, uma função que retorne o endereço dada sua latitude e longitude, e outra função que recebe dois endereços e retorna o tempo esperado utilizando bibliotecas (ex. google.maps.MapsLibrary) externas.
3. Analisar a possibilidade de uma ou poucas câmeras de radar estarem gerando dados errados, com placas ou tipo de veículo mal identificados, ou com datahora e/ou latitude/longitude equivocadas. Se identificados radares com tais problemas, remover os registros associados à ele do dataset. Assim diminuiríamos significativamente a quantidade de falsos positivos na solução proposta.

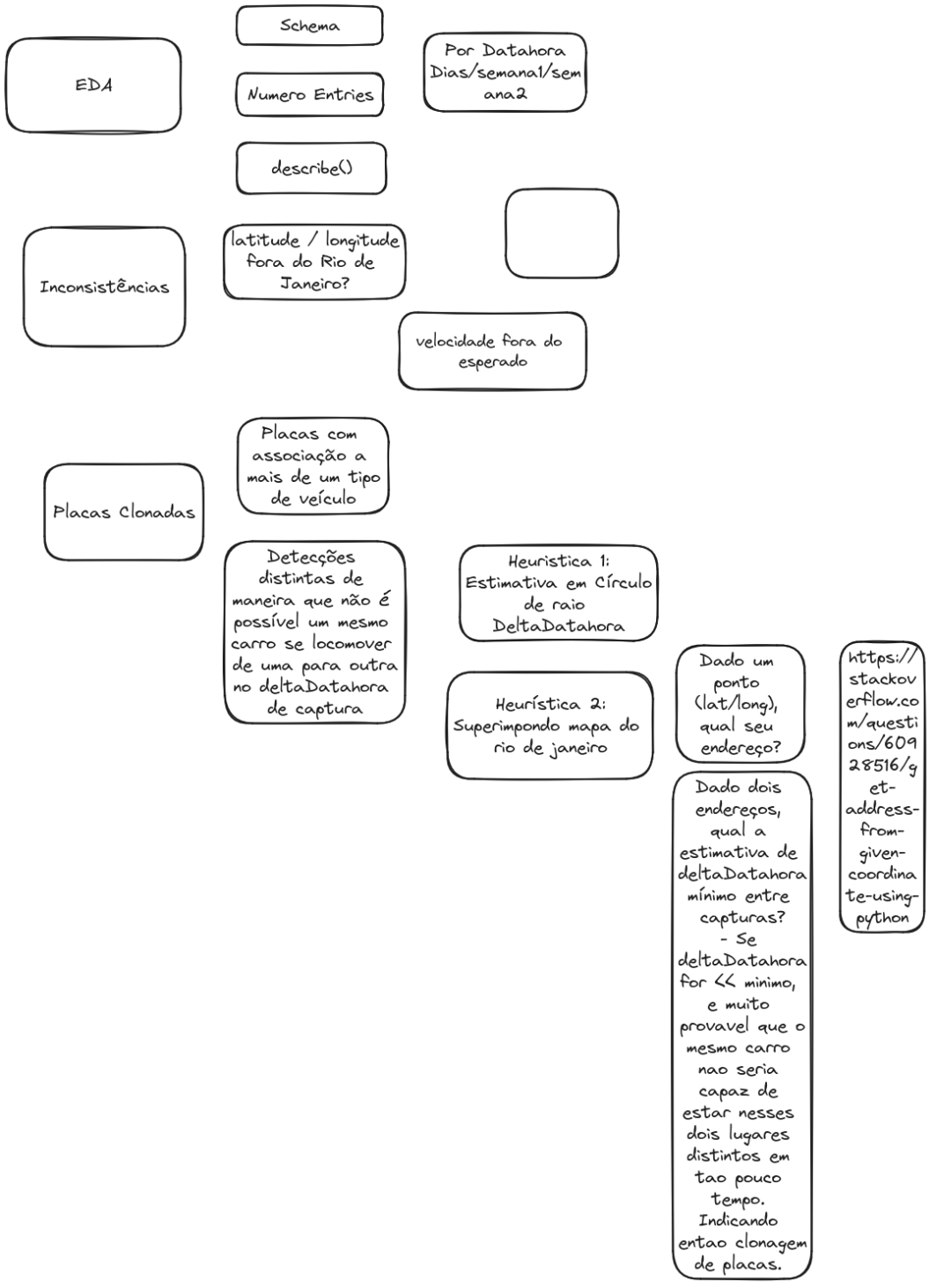

Figura 10: Artefato da ferramenta Excalidraw, utilizado na organização prévia e durante o desenvolvimento deste artigo.