

Содержание

- 1 Предобработка данных
- 2 Обзор данных
 - 2.1 Обзор float и int характеристик
 - 2.1.1 Средняя скорость
 - 2.1.2 Направление
 - 2.1.3 Широта
 - 2.1.4 Долгота
 - 2.1.5 Время
 - 2.2 Обзор object характеристик
 - 2.2.1 Маршруты
 - 2.2.2 Транспортные средства
 - 2.3 Рейсы
- 3 Визуализация рейсов
 - 3.1 Маршрут 3
 - 3.2 Маршрут 4
 - 3.3 Маршрут 291
 - 3.4 Маршрут 31
 - 3.5 Маршрут 6
 - 3.6 Маршрут 389
- 4 Выводы

Анализ данных транспортных маршрутов

Цель – проанализировать полученные данные.

Задачи:

- Предобработка данных
- Обзор и данных
- Визуализация и исследовательский анализ данных
- Выделение рейсов
- Поиск возможных ошибок в данных

Данные

- average_speed - Мгновенная скорость транспортного средства, полученная от приемника GPS, км/ч
- clid - Идентификатор участника программы
- direction - Направление движения в градусах (направление на север - 0 градусов). Диапазон значений 0-360

- lat - Широта точки в градусах.
- lon - Долгота точки в градусах.
- receive_time -
- route - Идентификатор маршрута.
- time - Дата и время получения координат точки от GPS-приемника (по Гринвичу).
- uuid - Уникальный идентификатор движущегося объекта (транспортного средства). Одинаковые uuid для разных типов транспорта (vehicle_type) недопустимы.
- vehicle_type - Тип общественного транспортного средства.

Предобработка данных

```
In [1]: # Импорт библиотек
import warnings
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import linkage, fcluster

pd.options.display.float_format = '{:.3f}'.format
warnings.filterwarnings('ignore')
```

```
In [2]: # импорт данных
df = pd.read_csv("test_task_log1.txt", sep="    ")
```

```
In [3]: # Функция обзора данных
def first_look(df):
    print('-----Первые 5 строк-----')
    display(df.head())
    print('\n')
    print('-----Тип данных-----')
    display(df.info())
    print('\n')
    print('-----Пропуски в данных-----')
    df_isna = df.isna().sum()
    if df_isna.sum().sum() > 0:
        display(df_isna.loc[df_isna > 0])
    else: print('Пропусков нет')
    print('\n')
    print('-----Дубликаты-----')
    if df.duplicated().sum() > 0:
        print('Дубликатов:', df.duplicated().sum())
    else:
        print('Дубликатов нет')
    print('\n')
```

```
In [4]: first_look(df)
```

```
-----Первые 5 строк-----
```

	average_speed	clid	direction	lat	lon	receive_time	route	time
0	4.444	c8g87gcr3	182.000	55.148	60.152	1701392963	3	1701392958
1	0.000	c8g87gcr3	0.000	55.135	60.149	1701393293	3	1701393286
2	4.722	c8g87gcr3	228.000	55.134	60.148	1701393323	3	1701393314
3	6.111	c8g87gcr3	229.000	55.134	60.148	1701393323	3	1701393318
4	8.333	c8g87gcr3	203.000	55.092	60.126	1701394067	3	1701394041

```

-----Тип данных-----
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 167305 entries, 0 to 167304
Data columns (total 10 columns):
#   Column          Non-Null Count  Dtype
---  -
0   average_speed    167305 non-null  float64
1   clid              167305 non-null  object
2   direction         167305 non-null  float64
3   lat               167305 non-null  float64
4   lon               167305 non-null  float64
5   receive_time      167305 non-null  int64
6   route             167305 non-null  object
7   time              167305 non-null  int64
8   uuid              167305 non-null  int64
9   vehicle_type      167305 non-null  object
dtypes: float64(4), int64(3), object(3)
memory usage: 12.8+ MB
None

```

```

-----Пропуски в данных-----
Пропусков нет

```

```

-----Дубликаты-----
Дубликатов: 69

```

Найдено 69 дубликатов. Их следует удалить, т.к. они не будут нести дополнительную информацию. Дубликаты могли появиться вследствие передачи данных чаще, чем за одну секунду.

```
In [5]: df = df.drop_duplicates()
print("Количество дубликатов: ", df.duplicated().sum())
```

Количество дубликатов: 0

Также следует удалить те строки, в которых `time` и `uuid` совпадают. Проверим есть ли такие.

```
In [6]: df.duplicated(["time", "uuid"]).sum()
```

Out[6]: 503

Их 503 штуки. Тоже удалим эти строки, т.к. вряд ли транспортное средство претерпело значительное изменение положения менее чем за секунду.

```
In [7]: df = df.drop_duplicates(["time", "uuid"])
print("Количество дубликатов: ", df.duplicated(["time", "uuid"]).sum())
```

Количество дубликатов: 0

Возможно достаточно много строк с нулевой скоростью транспорта, которые не помогут в решении стоящих перед нами задач. И их удалим тоже.

```
In [8]: df = df[df["average_speed"] != 0]
```

Других действий предобработки не требуется.

Обзор данных

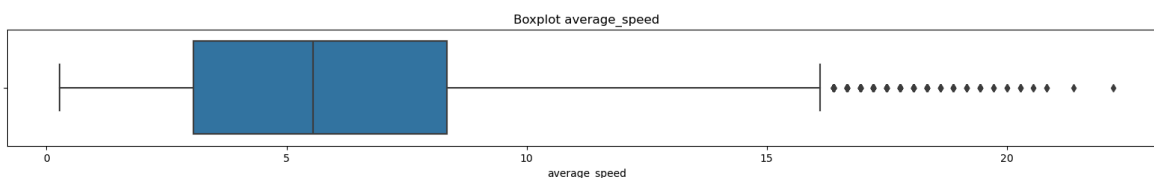
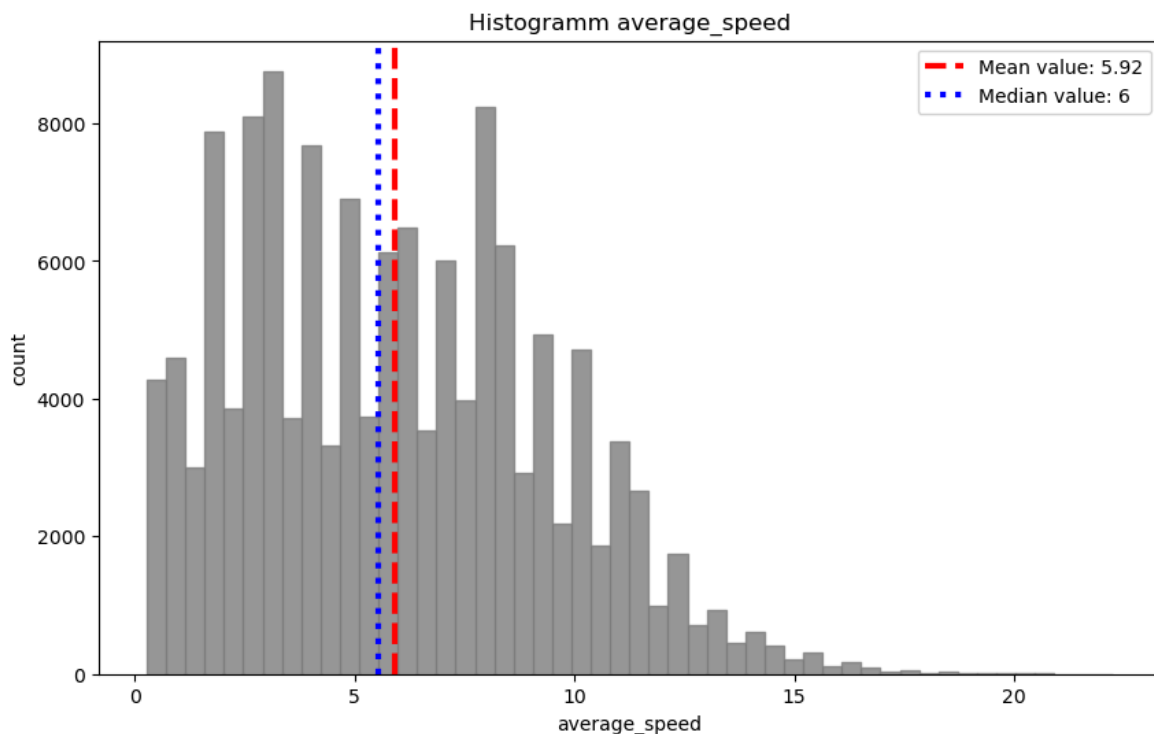
Обзор float и int характеристик

```
In [9]: # Функция для построения гистограммы и диаграммы размаха
def hist_maker(column, bins):
    print(column.describe())
    plt.figure(figsize=(10,6))
    plt.hist(column, bins=bins, color='grey', edgecolor='grey', alpha=0.8)
    plt.axvline(column.describe()['mean'], color='r', linestyle='dashed', linewidth=1)
    plt.axvline(column.describe()['50%'], color='b', linestyle='dotted', linewidth=1)
    plt.xlabel(column.name)
    plt.ylabel('count')
    plt.legend()
    plt.title('Histogramm {}'.format(column.name))
    plt.figure(figsize=(20,2))
    sns.boxplot(x=column)
    plt.title('Boxplot {}'.format(column.name))
```

Средняя скорость

```
In [10]: hist_maker(df["average_speed"], 50)
```

```
count    135973.000
mean         5.921
std         3.436
min         0.278
25%         3.056
50%         5.556
75%         8.333
max         22.222
Name: average_speed, dtype: float64
```

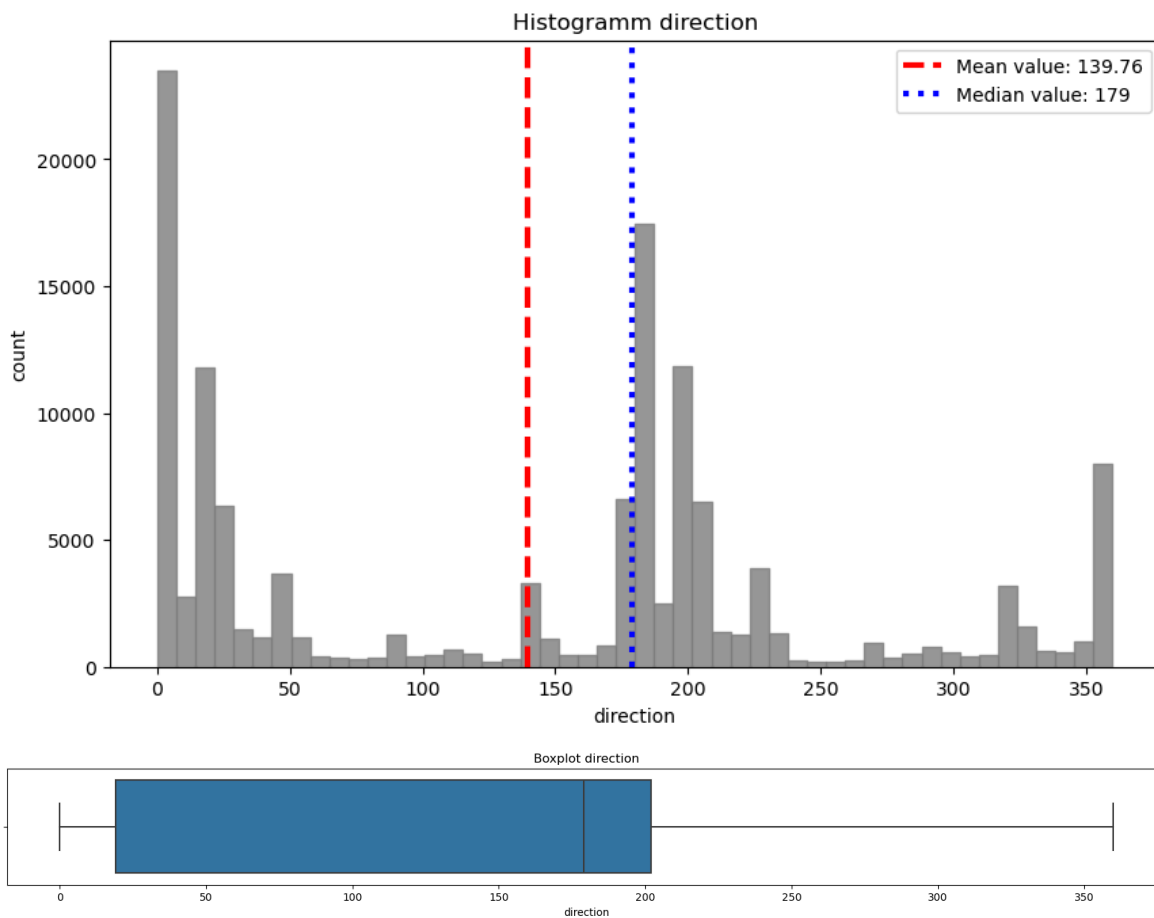


Средняя скорость составила 5.92. Выдающиеся значения есть, но не будем с ними ничего делать, т.к. в дальнейшем анализе значения скорости не понадобятся.

Направление

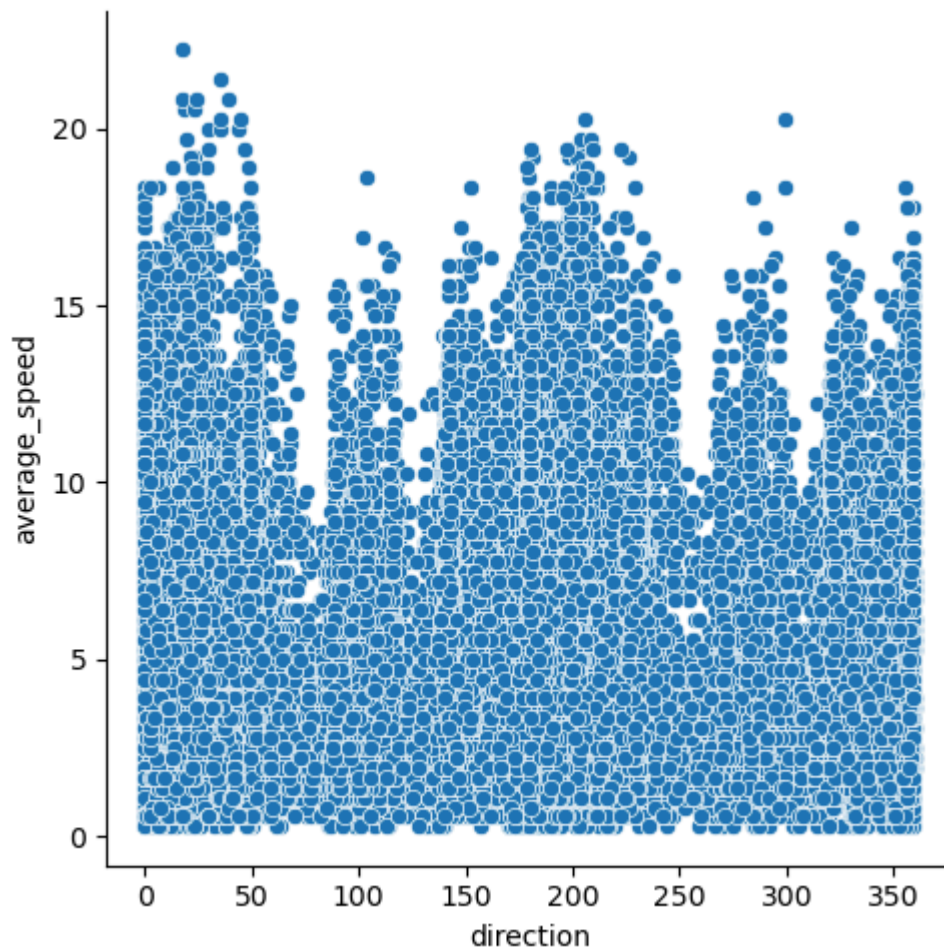
```
In [11]: hist_maker(df["direction"], 50)
```

```
count    135973.000
mean       139.761
std        113.251
min         0.000
25%        19.000
50%       179.000
75%       202.000
max       360.000
Name: direction, dtype: float64
```



Выделяются некоторые области направлений. Особенно много записей с околонулевыми значениями, что может быть связано с алгоритмами расчёта расстояний. Вероятней эти сигналы были записаны, когда скорость движения транспорта была низкой. Однако график направления и связи не подтверждает эту гипотезу.

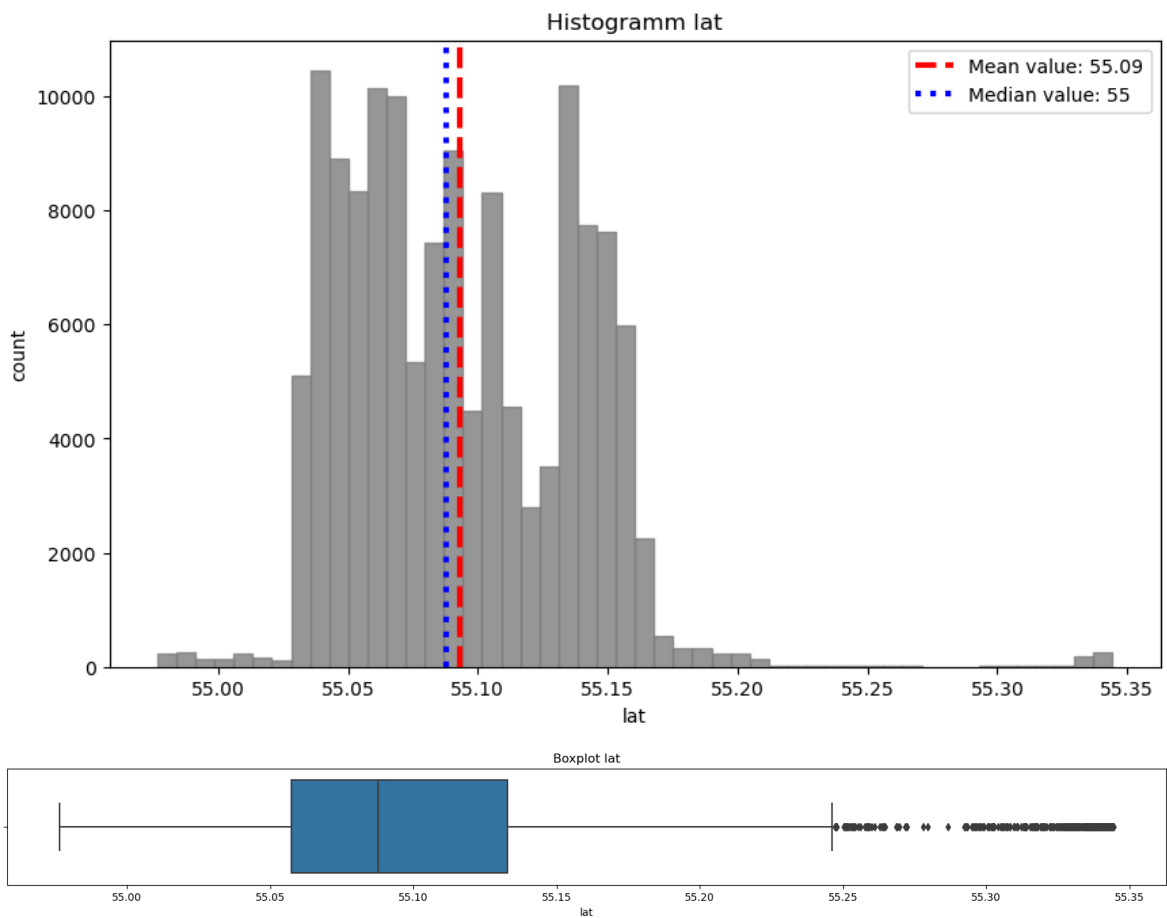
```
In [12]: sns.relplot(data=df, x="direction", y="average_speed");
```



Широта

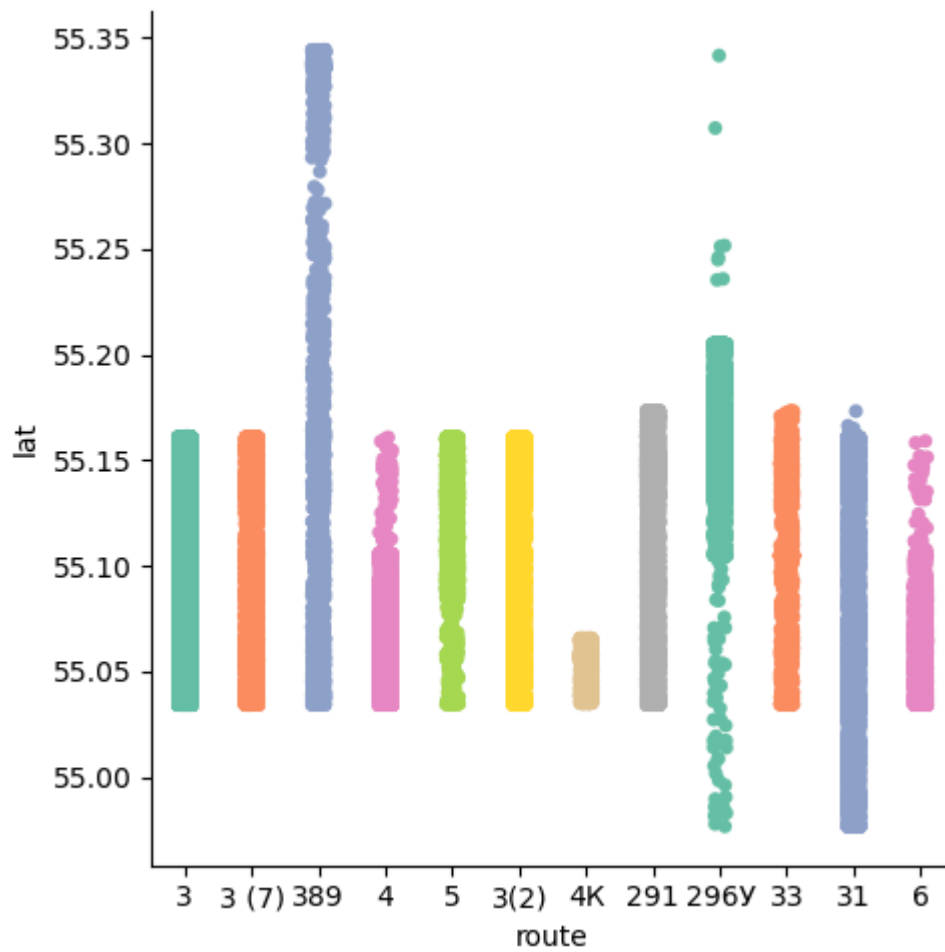
```
In [13]: hist_maker(df["lat"], 50)
```

```
count    135973.000
mean       55.093
std         0.044
min        54.977
25%        55.057
50%        55.088
75%        55.133
max        55.344
Name: lat, dtype: float64
```



Наблюдается выдающиеся значения. Они могут быть связаны как с отдельными маршрутами, так и с ошибками в данных.

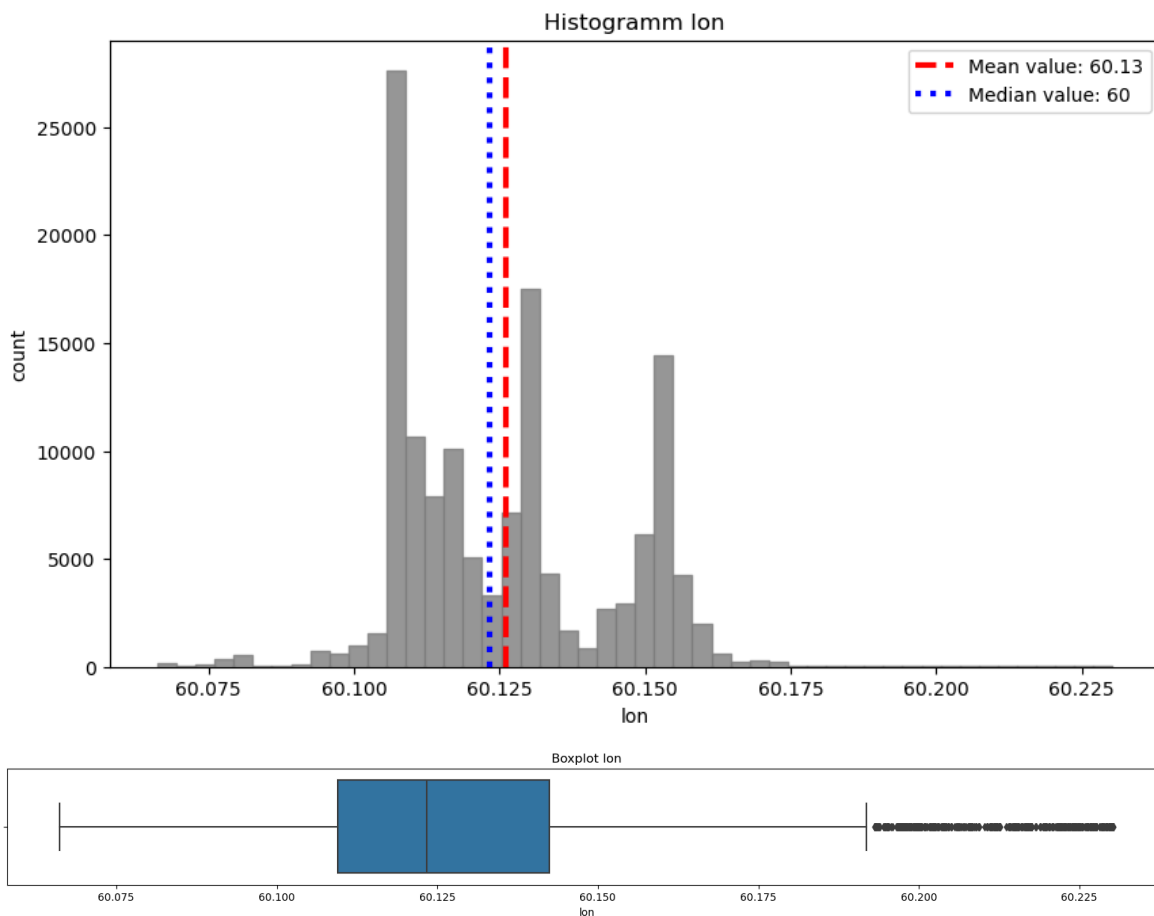
```
In [14]: sns.catplot(data=df, x="route", y="lat", palette="Set2");
```

Долгота

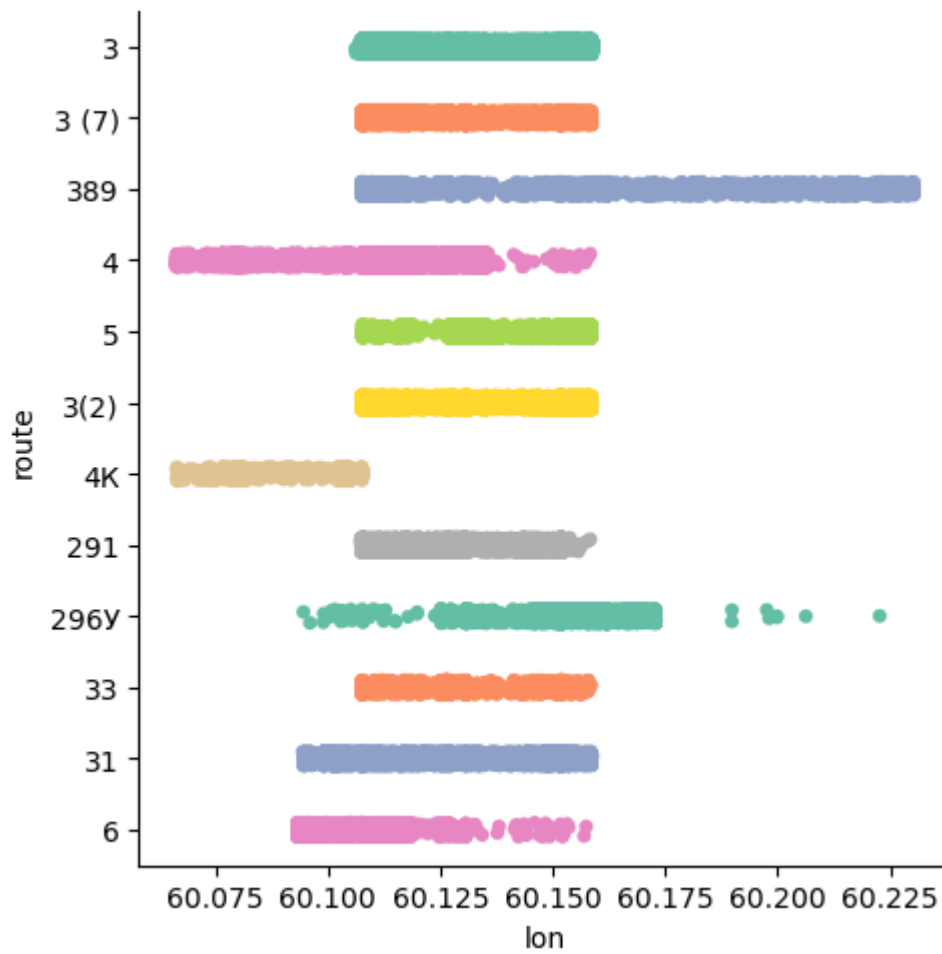
```
In [15]: hist_maker(df["lon"], 50)
```

```
count    135973.000
mean       60.126
std         0.019
min        60.066
25%        60.110
50%        60.123
75%        60.142
max        60.230
Name: lon, dtype: float64
```



По долготе выводы такие же как и по широте. На гистограмме отмечаются участки с узкими диапазонами значений долготы. Это может указывать на меридианальность пространственного положения маршрутов транспорта. Т.е. маршруты скорее всего проходят по дорожной инфраструктуре вытянутой с юга на север.

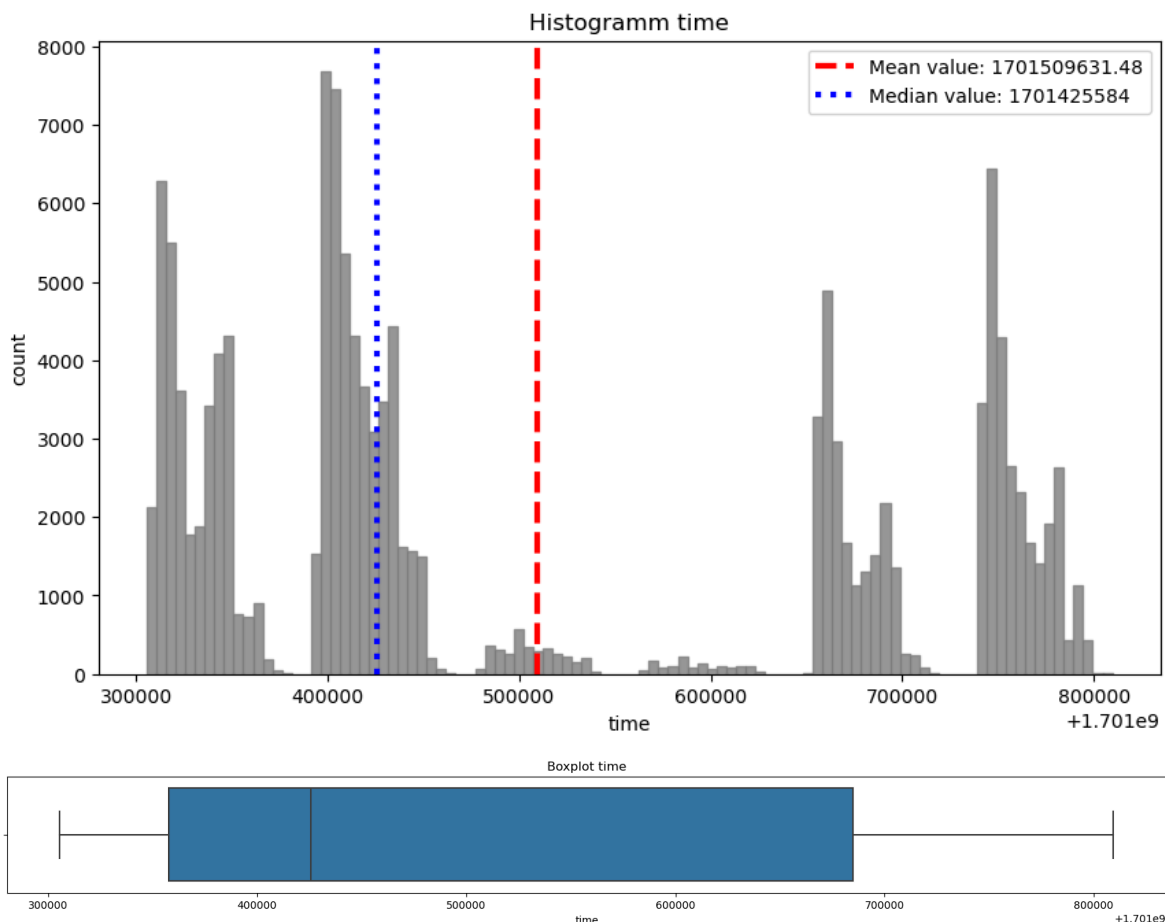
```
In [16]: sns.catplot(data=df, y="route", x="lon", palette="Set2");
```



Время

In [17]: `hist_maker(df["time"], 100)`

```
count      135973.000
mean    1701509631.484
std       170341.643
min      1701305555.000
25%      1701357631.000
50%      1701425584.000
75%      1701685070.000
max      1701809515.000
Name: time, dtype: float64
```



На гистограмме времени хорошо выделяется суточный ход записи. Увеличение количества строк соответствует дневному времени суток. Посмотрим за какое время были предоставлены данные.

```
In [18]: time_min = datetime.utcfromtimestamp(df["time"].min())
time_max = datetime.utcfromtimestamp(df["time"].max())

print("Период данных: ", time_min, " - ", time_max)
print("Номер дня недели: ", time_min.weekday(), " - ", time_max.weekday())
```

Период данных: 2023-11-30 00:52:35 - 2023-12-05 20:51:55
 Номер дня недели: 3 - 1

Мы также посмотрели и на дни недели. Наименьшая активность характерна для выходных дней.

Обзор object характеристик

Маршруты

Маршруты по которым передавались сигналы.

```
In [19]: print("Номера маршрутов: ", list(df["route"].unique()))
print("Количество маршрутов: ", len(df["route"].unique()))
```

Номера маршрутов: ['3', '3 (7)', '389', '4', '5', '3(2)', '4К', '291', '296У', '33', '31', '6']
 Количество маршрутов: 12

Транспортные средства

Сколько транспортных средств передавали данные по маршрутам?

```
In [20]: df.groupby(["route"])["uuid"].nunique().sort_values(ascending=False)
```

```
Out[20]: route
3         29
4         17
291        8
31         8
3 (7)       7
296Y        6
3(2)        6
6          5
33         4
4K         4
5          4
389        1
Name: uuid, dtype: int64
```

Сложим эти транспортные средства.

```
In [21]: df.groupby(["route"])["uuid"].nunique().sum()
```

```
Out[21]: 99
```

```
In [22]: print("Общее количество уникальных транспортных средств: ", df["uuid"].nunique())
```

Общее количество уникальных транспортных средств: 45

Можно сделать вывод о том, что одно и то же транспортное средство может ездить по разным маршрутам.

Рейсы

Согласно Википедии рейс это путь транспортного средства (корабля, судна, автомобиля, летательного аппарата и так далее) по определённом маршруту.

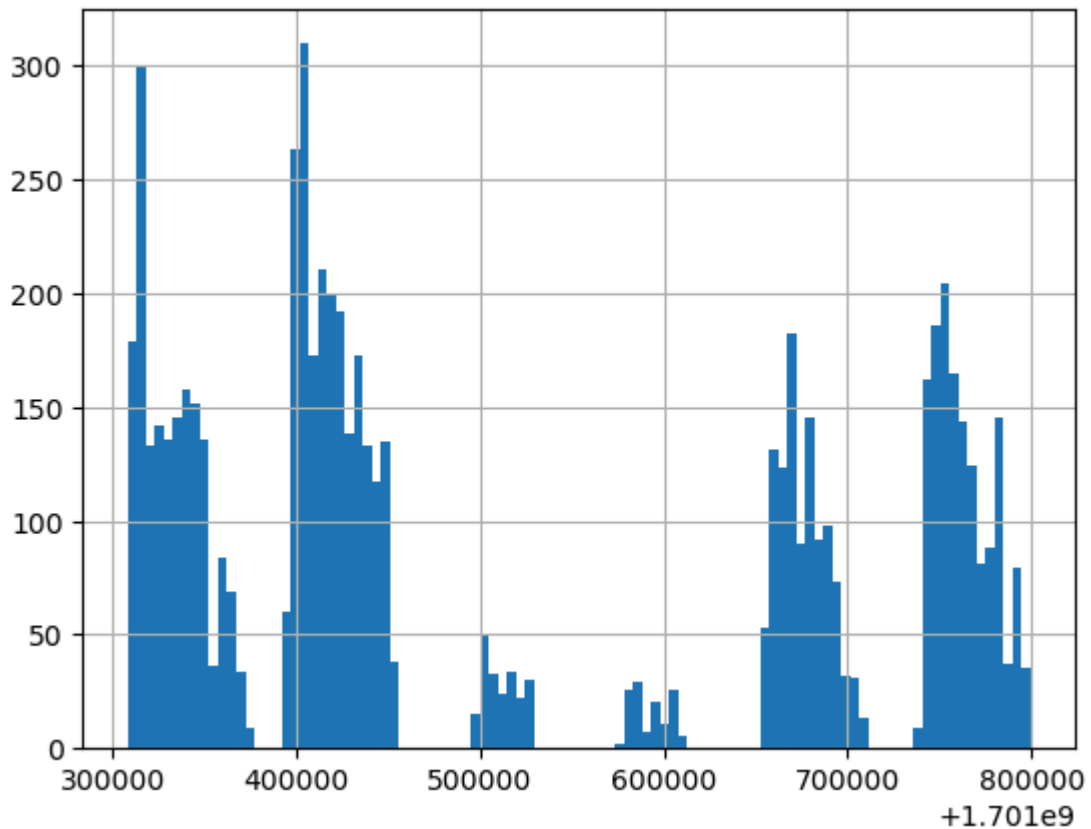
Рейсы в данных не выделены явно, поэтому в рамках тестового задания необходимо разработать методологию выделения рейса по доступным данным с целью выполнения последующих задач.

Пожалуй для этой цели лучше всего подходят данные о времени. На гистограмме ниже приведён пример для транспортного средства №1463178 маршрута №3. На нём достаточно хорошо выделяются периоды "активности".

```
In [23]: df[df["route"] == "3"]["uuid"].unique()
```

```
Out[23]: array([1463169, 1463178, 1463183, 1463538, 1504621, 1504623, 1504626,
                1504631, 1504827, 1504831, 1504833, 1504834, 1504839, 1504846,
                1504912, 1504916, 1504917, 9028617, 9058974, 1504624, 1504625,
                1504838, 1504848, 1480169, 1504830, 1504856, 1504841, 1504844,
                9058975], dtype=int64)
```

```
In [24]: df[(df["route"] == "3") & (df["uuid"] == 1463178)]["time"].hist(bins=100);
```



Подготовим функцию для выделения рейсов.

Описание работы функции:

- Сначала она создает пустой DataFrame `df_concat_route`, который будет содержать результаты кластеризации для всех маршрутов.
- Затем она итерируется по уникальным значениям маршрутов в столбце "route" переданного DataFrame `df`.
- Для каждого маршрута она создает новый пустой DataFrame `df_concat_uuid`, который будет содержать результаты кластеризации для всех уникальных `uuid` на данном маршруте.
- Затем она фильтрует исходные данные `df`, чтобы получить только записи, относящиеся к текущему маршруту, и извлекает уникальные значения `uuid` для этого маршрута.
- Далее она итерируется по уникальным `uuid` и для каждого из них фильтрует данные, чтобы получить только записи с этим `uuid` на текущем маршруте.
- Для каждого `uuid` она извлекает временной ряд `X`, стандартизирует его с использованием `StandardScaler`, затем выполняет кластеризацию с помощью метода `linkage`, используя критерий объединения 'ward'.
- Определяется пороговое значение расстояния (`threshold`), и кластерные метки вычисляются с помощью `fcluster`.
- К кластерным меткам добавляются метки, содержащие информацию о маршруте и `uuid`, и сохраняются в столбец `clusters`.

- Результаты кластеризации для данного uuid на текущем маршруте добавляются к df_concat_uuid.
- Наконец, результаты для всех uuid на текущем маршруте объединяются с результатами для других маршрутов в df_concat_route, и возвращается итоговый DataFrame

```
In [25]: def get_raise(df):
df_concat_route = pd.DataFrame()
route_list = df["route"].unique()
for route in route_list:
    df_concat_uuid = pd.DataFrame()
    uuid_list = df[df["route"] == route]["uuid"].unique()
    df_route = df[df["route"] == route]

    for uuid in uuid_list:
        df_uuid = df_route[df_route["uuid"] == uuid]
        X = df_uuid['time'].reset_index()

        # стандартизируем данные
        scaler = StandardScaler()
        scaler.fit(X) # обучите scaler на обучающей выборке методом fit
        X_st = scaler.transform(X) # стандартизируем обучающую выборку метод
        linked = linkage(X_st, method='ward')

        # определяем кластеры на основе порогового значения расстояния
        threshold = 15.0 # подобрам эмперическим путём
        cluster_labels = fcluster(linked, threshold, criterion='distance')

        cluster_labels_route = []

        for i in cluster_labels:
            name = str(route) + "_" + str(uuid) + "_" + str(i)
            cluster_labels_route.append(name)

        df_uuid['raises'] = cluster_labels_route

        df_concat_uuid = pd.concat([df_concat_uuid, df_uuid])

    df_concat_route = pd.concat([df_concat_route, df_concat_uuid])


    return df_concat_route
```

```
In [26]: df_raise = get_raise(df)
```

```
In [27]: df_raise.head()
```

Out[27]:

	average_speed	clid	direction	lat	lon	receive_time	route	time
0	4.444	c8g87gcr3	182.000	55.148	60.152	1701392963	3	1701392958
2	4.722	c8g87gcr3	228.000	55.134	60.148	1701393323	3	1701393314
3	6.111	c8g87gcr3	229.000	55.134	60.148	1701393323	3	1701393318
4	8.333	c8g87gcr3	203.000	55.092	60.126	1701394067	3	1701394041
5	9.167	c8g87gcr3	199.000	55.076	60.115	1701394397	3	1701394387



Сохраняем датафрейм в csv для последующей визуализации рейсов с помощью QGIS.

In [28]: `df_raise.to_csv("bus_routes.csv", index=False)`

Ниже представлен массив рейсов в виде "route_uuid_raise". Можно сделать вывод о том, что одно и тоже транспортное средство может встречаться на разных маршрутах.

In [29]: `df_raise["raises"].unique()`

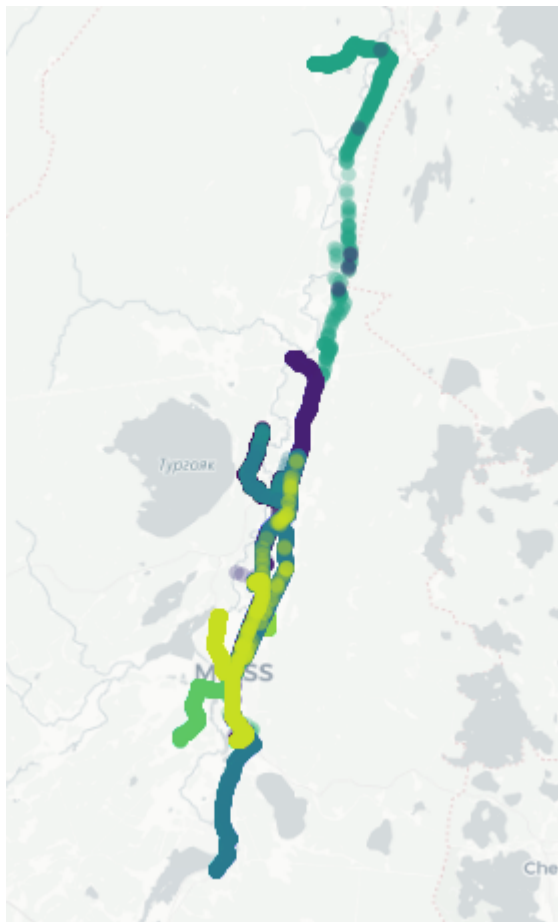

```
Out[29]: array(['3_1463169_1', '3_1463169_3', '3_1463169_2', '3_1463178_3',
               '3_1463178_5', '3_1463178_4', '3_1463178_1', '3_1463178_2',
               '3_1463183_3', '3_1463183_2', '3_1463183_1', '3_1463538_2',
               '3_1463538_3', '3_1463538_4', '3_1463538_1', '3_1504621_3',
               '3_1504621_2', '3_1504621_1', '3_1504623_1', '3_1504623_3',
               '3_1504623_2', '3_1504626_3', '3_1504626_5', '3_1504626_4',
               '3_1504626_1', '3_1504626_2', '3_1504631_1', '3_1504631_3',
               '3_1504631_4', '3_1504631_2', '3_1504827_1', '3_1504827_2',
               '3_1504827_3', '3_1504831_1', '3_1504831_2', '3_1504831_3',
               '3_1504833_1', '3_1504833_2', '3_1504833_3', '3_1504834_2',
               '3_1504834_4', '3_1504834_3', '3_1504834_1', '3_1504839_1',
               '3_1504839_2', '3_1504839_3', '3_1504846_2', '3_1504846_4',
               '3_1504846_3', '3_1504846_1', '3_1504912_3', '3_1504912_2',
               '3_1504912_1', '3_1504916_2', '3_1504916_5', '3_1504916_3',
               '3_1504916_4', '3_1504916_1', '3_1504917_3', '3_1504917_2',
               '3_1504917_1', '3_9028617_2', '3_9028617_3', '3_9028617_4',
               '3_9028617_1', '3_9058974_3', '3_9058974_5', '3_9058974_4',
               '3_9058974_1', '3_9058974_2', '3_1504624_1', '3_1504625_2',
               '3_1504625_1', '3_1504838_2', '3_1504838_3', '3_1504838_1',
               '3_1504848_1', '3_1504848_2', '3_1480169_4', '3_1480169_3',
               '3_1480169_1', '3_1480169_2', '3_1504830_1', '3_1504830_3',
               '3_1504830_2', '3_1504856_2', '3_1504856_3', '3_1504856_1',
               '3_1504841_2', '3_1504841_3', '3_1504841_1', '3_1504844_2',
               '3_1504844_3', '3_1504844_1', '3_9058975_1', '3_9058975_2',
               '3 (7)_1480169_3', '3 (7)_1480169_2', '3 (7)_1480169_1',
               '3 (7)_1504841_2', '3 (7)_1504841_3', '3 (7)_1504841_1',
               '3 (7)_1504623_2', '3 (7)_1504623_1', '3 (7)_1504624_2',
               '3 (7)_1504624_1', '3 (7)_1504838_2', '3 (7)_1504838_3',
               '3 (7)_1504838_1', '3 (7)_1504856_1', '3 (7)_1504856_2',
               '3 (7)_1504848_2', '3 (7)_1504848_3', '3 (7)_1504848_1',
               '389_1500432_2', '389_1500432_4', '389_1500432_3', '389_1500432_1',
               '4_1504624_2', '4_1504624_1', '4_1504838_2', '4_1504838_1',
               '4_1504844_3', '4_1504844_2', '4_1504844_1', '4_1504856_2',
               '4_1504856_3', '4_1504856_1', '4_1504907_1', '4_1504907_3',
               '4_1504907_2', '4_1504827_3', '4_1504827_2', '4_1504827_1',
               '4_1504841_1', '4_1504621_1', '4_1504621_2', '4_1504625_2',
               '4_1504625_1', '4_1504848_2', '4_1504848_3', '4_1504848_1',
               '4_1504921_1', '4_1504921_2', '4_1504830_2', '4_1504830_4',
               '4_1504830_3', '4_1504830_1', '4_1504909_1', '4_1504909_3',
               '4_1504909_2', '4_1504908_1', '4_1504908_2', '4_1504833_1',
               '4_1504833_2', '4_1504833_4', '4_1504833_3', '4_1504839_3',
               '4_1504839_2', '4_1504839_1', '4_1504920_2', '4_1504920_3',
               '4_1504920_1', '5_1504625_3', '5_1504625_2', '5_1504625_1',
               '5_1504844_3', '5_1504844_2', '5_1504844_1', '5_1504848_2',
               '5_1504848_3', '5_1504848_1', '5_1504624_2', '5_1504624_3',
               '5_1504624_1', '3(2)_1504830_1', '3(2)_1504830_2',
               '3(2)_1504848_2', '3(2)_1504848_3', '3(2)_1504848_1',
               '3(2)_1504841_1', '3(2)_1504841_2', '3(2)_1504917_2',
               '3(2)_1504917_3', '3(2)_1504917_1', '3(2)_1504846_2',
               '3(2)_1504846_3', '3(2)_1504846_1', '3(2)_1504912_3',
               '3(2)_1504912_2', '3(2)_1504912_1', '4K_1504907_1', '4K_1504907_2',
               '4K_1504921_1', '4K_1504921_2', '4K_1504909_1', '4K_1504909_2',
               '4K_1504920_1', '4K_1504920_2', '291_1504908_1', '291_1504908_2',
               '291_1504908_3', '291_1504911_3', '291_1504911_2', '291_1504911_1',
               '291_2412958_3', '291_2412958_2', '291_2412958_1', '291_2417908_3',
               '291_2417908_1', '291_2417908_2', '291_1504920_2', '291_1504920_1',
               '291_1504910_2', '291_1504910_3', '291_1504910_1', '291_1504926_1',
               '291_1504926_2', '291_2417915_2', '291_2417915_1',
               '296Y_1504910_2', '296Y_1504910_4', '296Y_1504910_3',
               '296Y_1504910_1', '296Y_1504921_1', '296Y_1504921_3',
```

```
'296Y_1504921_2', '296Y_1504911_1', '296Y_1504911_2',
'296Y_1504920_1', '296Y_1504920_3', '296Y_1504920_2',
'296Y_1504926_1', '296Y_1504907_2', '296Y_1504907_1',
'33_1504914_1', '33_2417915_2', '33_2417915_3', '33_2417915_1',
'33_1504925_1', '33_1504925_2', '33_2412959_1', '33_2412959_2',
'31_1504920_1', '31_1504920_2', '31_1504920_3', '31_2412959_1',
'31_2412959_4', '31_2412959_3', '31_2412959_2', '31_2417908_1',
'31_1504909_1', '31_1504909_2', '31_2412958_1', '31_2412958_2',
'31_1504907_2', '31_1504907_3', '31_1504907_1', '31_1504914_1',
'31_1504914_2', '31_2422524_2', '31_2422524_3', '31_2422524_1',
'6_1504925_2', '6_1504925_3', '6_1504925_4', '6_1504925_1',
'6_1504907_2', '6_1504907_1', '6_1504911_1', '6_1504911_3',
'6_1504911_2', '6_1504921_1', '6_1504921_2', '6_1504908_4',
'6_1504908_3', '6_1504908_1', '6_1504908_2'], dtype=object)
```

Визуализация рейсов

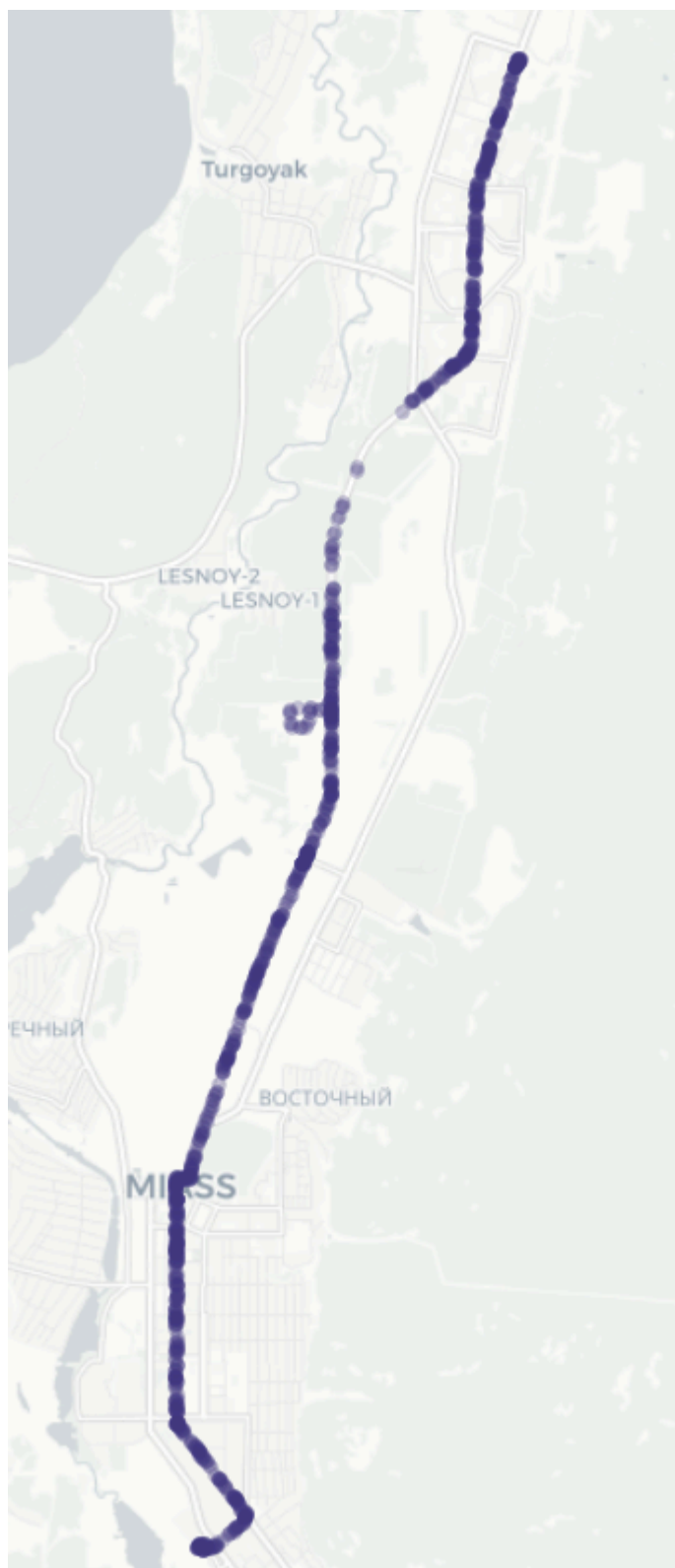
Наиболее удобный инструмент визуализации подобных данных это ПО QGIS.

На рисунке ниже приведена карта с отфильтрованными точками. Таким образом, можно увидеть, что транспортные маршруты расположены в Челябинской области "агломерации" Миасс.



Ниже будет приведена визуализация некоторых рейсов.

Маршрут 3



Это троллейбусный маршрут, согласно предоставленным данным. На карте что частично запись есть и на территории троллейбусного депо. Достаточно мало точек наблюдается в "лесистой местности". Вероятно мобильный сигнал и сигнал GPS/ГЛОНАС в этом районе ловит хуже, чем в городе или транспортное средство в этой части едет быстрее среднего. В будущем можно провести доп анализ.

```
In [30]: df_raise[df_raise["route"] == "3"][:1]
```

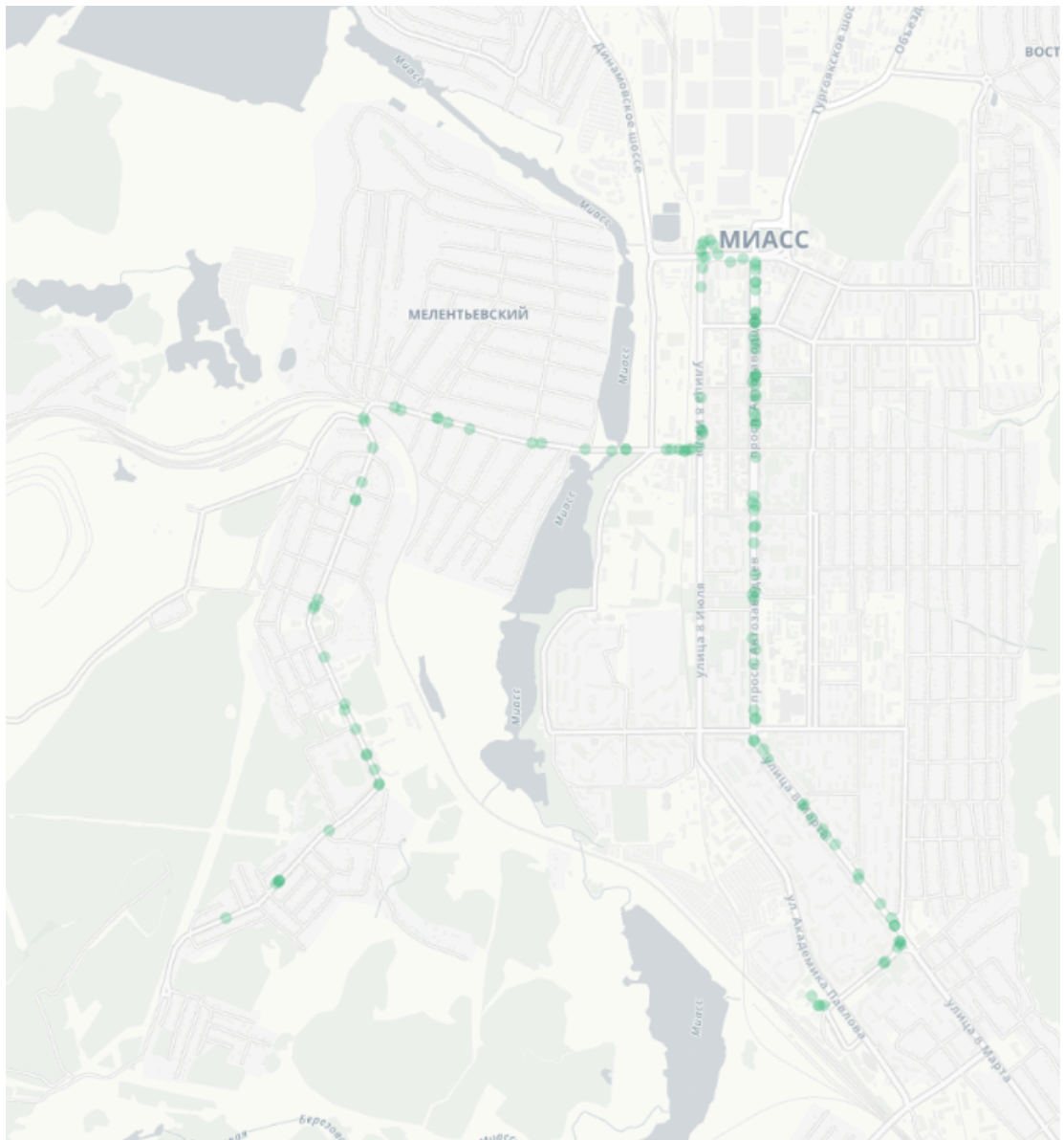
Out[30]:

	average_speed	clid	direction	lat	lon	receive_time	route	time
0	4.444	c8g87gcr3	182.000	55.148	60.152	1701392963	3	1701392958



Маршрут 4





В данных возможна ошибка. Т.к. встречаются разные маршруты, хотя по данным свободных источников этот маршрут должен соответствовать второму рисунку данного раздела. Это может быть связано с тем, что одно и то же транспортное средство может следовать по нескольким маршрутам. При этом информация о маршруте в транспортном средстве может не переключаться.

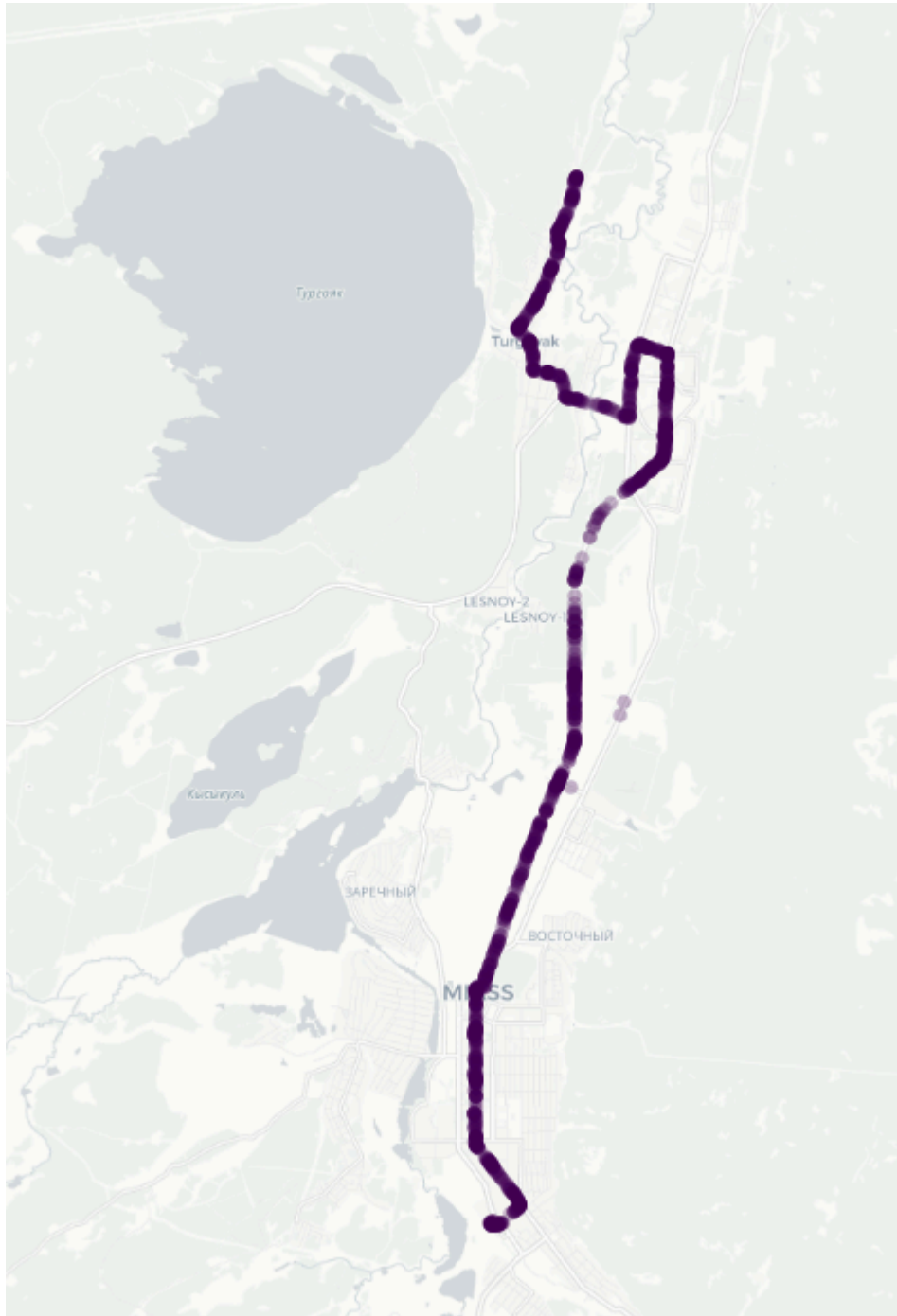
Пример строки маршрута 4:

```
In [31]: df_raise[df_raise["route"] == "4"][:1]
```

```
Out[31]:
```

	average_speed	clid	direction	lat	lon	receive_time	route
13353	6.111	c8g87gcr3	0.000	55.061	60.108	1701394401	4 170135

Маршрут 291



```
In [32]: df_raise[df_raise["route"] == "291"][1:]
```

```
Out[32]:
```

	average_speed	clid	direction	lat	lon	receive_time	route
36942	0.278	c8g87gcr3	0.000	55.140	60.152	1701393233	291 170139



По предоставленным данным маршрут 291 это троллейбусный маршрут, однако, судя по карте это не так. Согласно свободным источникам это автобусный маршрут. Наблюдаются выскочившие точки из основного маршрута, что может быть связано с ошибками в определении местоположения транспортного средства.

Маршрут 31



In [33]: `df_raise[df_raise["route"] == "291"][:1]`

Out[33]:

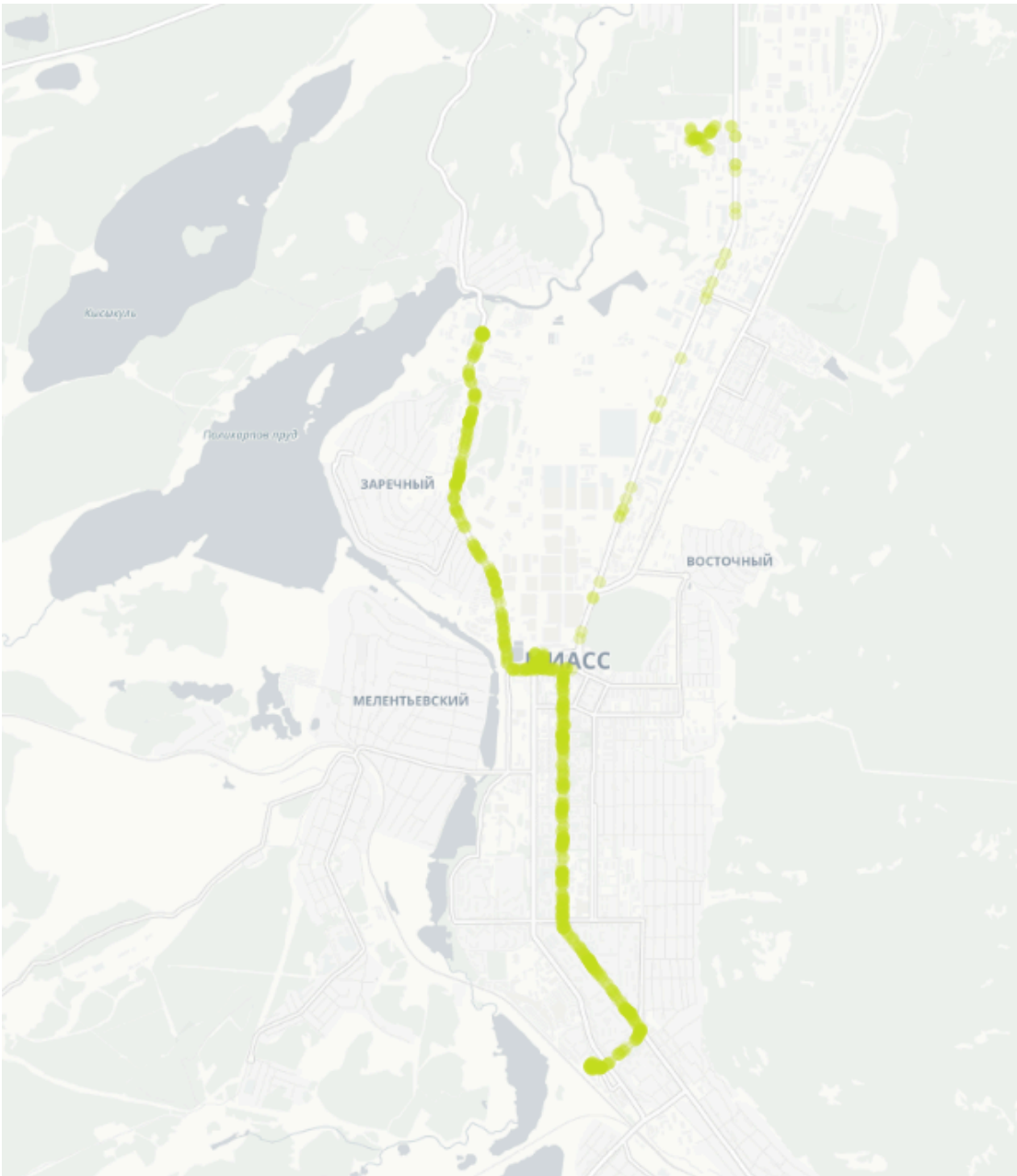
	average_speed	clid	direction	lat	lon	receive_time	route
36942	0.278	c8g87gcr3	0.000	55.140	60.152	1701393233	291 170139



По свободным данным это автобусный маршрут. По предоставленным - троллейбусный. По карте видно, что транспорт заезжает в троллейбусное депо. Там

он может базироваться и выезжать из депо оттуда. Возможно, в Миасе от троллейбусов отказались, а классификация в предоставленных данных осталась старая, как и название депо (троллейбусное.)

Маршрут 6



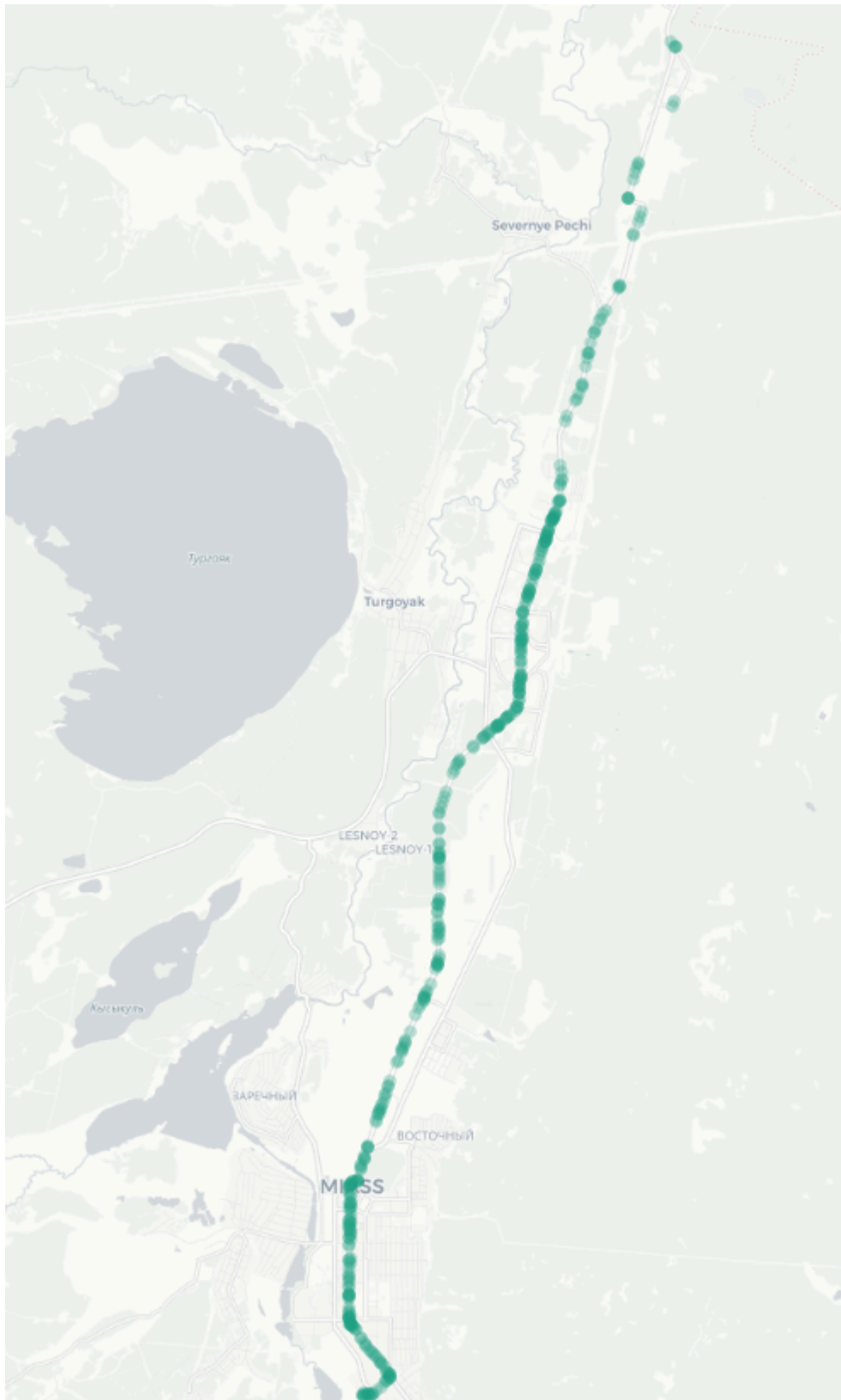
```
In [34]: df_raise[df_raise["route"] == "6"][:1]
```

Out[34]:

	average_speed	clid	direction	lat	lon	receive_time	route
49233	3.056	c8g87gcr3	0.000	55.059	60.108	1701393323	6 170139

По свободным источникам маршрут 6 - автобусный. Также видно что транспорт выезжает на маршрут из троллейбусного депо.

Маршрут 389



```
In [35]: df_raise[df_raise["route"] == "389"][:1]
```

```
Out[35]:
```

	average_speed	clid	direction	lat	lon	receive_time	route
10476	2.778	c8g87gcr3	85.000	55.034	60.111	1701399022	389 170139



Согласно карте междугородний маршрут. Возможно имеет самое редкое расписание, т.к. по данным было выделено всего лишь 4 рейса.

Выводы

- Предобработка данных
 - Удалены дубликаты
 - Удалены строки с нулевой скоростью
- Обзор данных
 - Период представленных данных: 2023-11-30 00:52:35 – 2023-12-05 20:51:55
 - Дни недели: четверг - вторник
 - Одно и то же транспортное средство может ездить по разным маршрутам
 - Проведено разделение точек на рейсы по столбцу время
- Визуализация данных
 - Данные на территорию города Миасса
 - С помощью QGIS были визуализированы семь рейсов шести маршрутов
- Ошибки в данных
 - Неточность в определении местоположения
 - Наблюдаются "выскакивающие" от пути маршрута точки
 - Иногда передача информации о местоположении достаточно редка
 - В типе транспортного средства наблюдаются ошибки - возможно информация в данных устаревшая
 - Могут наблюдаться несколько рейсов разных маршрутов в рамках одного маршрута, что может быть связано с использованием одного транспортного средства на нескольких маршрутах
 - Благодаря визуализации видно, что на некоторых рейсах кроме пути маршрута фиксируется и путь до депо - в большинстве случаев до троллейбусного депо Миасса.