

```
import geopandas as gpd
alt.data_transformers.disable_max_rows()

df = pd.read_csv('uc.csv')
df.head()
```

```

df.head()

df2 = pd.read_csv('uss.csv')
df2.head()

ucor = gpd.read_file('ukraine_2.json')

# Для візуалізацій я використовувала не повний маршрут, а лише точки відправлення та прибуття.
# Це не дуже добре сказалось на детальності даних, багато малих міст (проміжних зупинок) проігноровано.

# Отримаємо відстань, час прибуття та відправлення
k = 1
for i in range(3, 31752):
    row = df2.loc[i]
    if pd.isnull(row['Назва автобусного маршруту']):
        pass
    else:
        df2['відправлення'][i] = df2['відправлення'][i+1]
        df2['прибуття'][k] = df2['прибуття'][i-1]
        df2['відстань'][k] = df2['відстань'][i-1]
        k = i
#print(df2)

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/ipykernel_launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
    if __name__ == '__main__':
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
    # Remove the CWD from sys.path while we load stuff.

# Відсікаємо всі села. Цей спосіб не є дуже добрий, але він був потрібен для того щоб уникнути дублікатів.
# Можна було не видаляти всі села, лише в разі дублікатів лишати те, в якого вона найбільша.
# Але дані про поуляцію не є повтоцінними.
df = df[df['type'] != 'village']
df.reset_index(drop=True, inplace=True)

print(df)

      koatuu          town      type      obl \
0     110100000  м. Сімферополь    city  Автономна Республіка Крим
1     110165300  смт Аерофлотський   smt  Автономна Республіка Крим
2     110165600  смт Гресівський   smt  Автономна Республіка Крим
3     110165601      с. Бітумне small_village  Автономна Республіка Крим
4     110165800  смт Комсомольське   smt  Автономна Республіка Крим
...       ...         ...
2507  80000000000      м. Київ      city           Київ
2508  85000000000      м. Севастополь    city  Севастополь
2509  8536310300      м. Інкерман      town  Севастополь
2510  8536965300      смт Кача       smt  Севастополь
2511  8536990203  с. Сонячний small_village  Севастополь

      dis oldname      lat      lon  pop_2014  pop_2015 \
0  Сімферополь     NaN  44.952139  34.102457  338319.0      NaN
1  Сімферополь     NaN  45.019551  34.001051    2359.0      NaN
2  Сімферополь     NaN  45.010178  34.026487   11509.0      NaN
3  Сімферополь     NaN  45.019471  34.045061      NaN      NaN
4  Сімферополь     NaN  45.019896  34.021467   4892.0      NaN

```

	pop_2010	pop_ratec
0	NaN	338319.0
1	NaN	2359.0
2	NaN	11509.0
3	NaN	221.0
4	NaN	4892.0

```

...
2507 2906569.0 2906569.0
2508      NaN 344853.0
2509      NaN 12028.0
2510      NaN 5137.0
2511      NaN 1778.0

[2512 rows x 12 columns]

# Готуємо координати міст до роботи.
cor = {}
cor['T'] = df['town'].apply(lambda x: ' '.join(x.split(' ') [1:]))
cor['lat'] = df['lat']
cor['lon'] = df['lon']
cor['obl'] = df['obl']
cor = pd.DataFrame.from_dict(cor)

cor.drop_duplicates('T', keep='first', inplace=True)
cor.reset_index(drop=True, inplace=True)

print(cor)

          T      lat      lon          obl
0   Сімферополь  44.952139  34.102457  Автономна Республіка Крим
1   Аерофлотський  45.019551  34.001051  Автономна Республіка Крим
2   Гречівський  45.010178  34.026487  Автономна Республіка Крим
3    Бітумне  45.019471  34.045061  Автономна Республіка Крим
4  Комсомольське  45.019896  34.021467  Автономна Республіка Крим
...
2040      ...     ...
2040      Київ  50.450107  30.524050          ...
2041      Севастополь  44.605443  33.522084  Севастополь
2042      Інкерман  44.618104  33.604327  Севастополь
2043      Кача  44.775508  33.543754  Севастополь
2044      Сонячний  44.786110  33.619190  Севастополь

[2045 rows x 4 columns]

# Лишаємо лише дані про початкову та кінцеві станції. Приводимо їх до ладу.
df2_test = df2[df2['Назва автобусного маршруту'].isnull() == False]

df2_m = {}
df2_m['from'] = df2_test['Назва автобусного маршруту'].apply(lambda x: x.split('-')[0].strip())
df2_m['to'] = df2_test['Назва автобусного маршруту'].apply(lambda x: x.split('-')[1].strip())
df2_m['start_t'] = df2_test['відправлення']
df2_m['end_t'] = df2_test['прибуття']
df2_m['l'] = df2_test['відстань']

df2_m = pd.DataFrame.from_dict(df2_m)

df2_m.reset_index(drop=True, inplace=True)

print(df2_m)

          from          to      start_t      end_t      l
0       Черкаси        Київ  відправлення  прибуття  відстань
1  Дебальцеве  Красний Луч           8:45    10:00      52
2  Городенка    Чернівці           7:35    9:15      64
3       Бар    Віньківці           6:10    7:20      38
4  Скадовськ   Київ           6:45   21:30     707
...
3905  Кам'янець_Подільський    Затока           ...
3906  Хорошів (Іршанськ)      Київ           5:00    8:20      205
3907  Хорошів (Іршанськ)      Київ           13:30   17:00      205
3908      Могилів  Подільський           9:20   17:00     386
3909      Олевськ  Хмельницький           3:00  прибуття  відстань

```

```
In [6]: # Так як дані не ідеальні, то деякі з них при конвертації ламались, тому робимо перевірки на відність.  
df2_m['from_lat'] = df2_m['from']  
fr_lat = []  
fr_lon = []  
to_lat = []
```

```
to_lon = []

ex = []
for i in range (0, 3910):
    row = df2_m.loc[i]
    fr = row['from']
    to = row['to']

    if (cor[cor['T'] == fr]['lat']).empty:
        fr_lat.append(0.0)
    else:
        fr_lat.append(float(cor[cor['T'] == fr]['lat']))

    if (cor[cor['T'] == fr]['lon']).empty:
        fr_lon.append(0.0)
    else:
        fr_lon.append(float(cor[cor['T'] == fr]['lon']))

    if (cor[cor['T'] == to]['lat']).empty:
        to_lat.append(0.0)
    else:
        to_lat.append(float(cor[cor['T'] == to]['lat']))

    if (cor[cor['T'] == to]['lon']).empty:
        to_lon.append(0.0)
    else:
        to_lon.append(float(cor[cor['T'] == to]['lon']))

df2_m['from_lat'] = fr_lat
df2_m['from_lon'] = fr_lon
df2_m['to_lat'] = to_lat
df2_m['to_lon'] = to_lon

#print(df2_m)

test = df2_m[df2_m['from_lat']!=0]
test = test[test['from_lon']!=0]

test = test[test['to_lat']!=0]
test = test[test['to_lon']!=0]

points= alt.Chart(test).mark_point(filled=True, color="black", size = 9).encode(
    longitude='to_lon:Q',
    latitude='to_lat:Q',
    size = alt.Size('count(to)+ count(from):Q', title = 'Кількість відправлень:'),
    tooltip = alt.Tooltip('to')
).properties(
    title='',
    width=600,
    height=600)

flows = alt.Chart(test).mark_rule(strokeCap = 'round', color ='orange', opacity= 0.3).encode(
    longitude = alt.Longitude('from_lon:Q'),
    latitude = alt.Latitude('from_lat:Q'),
    longitude2 = alt.Longitude2('to_lon:Q'),
    latitude2 = alt.Latitude2('to_lat:Q'),
    tooltip = alt.Tooltip(['from', 'to'])
).properties(
    title='Щоденні автобусні відправлення-прибуття по Україні, 2020 р.',
    width= 500,
    height=300
)

alt.layer( flows + points).properties(width = 500, height = 300).configure_legend(orient = 'bottom-left').config



# Ця візуалізація показує які з якого міста та в яке є маршрути.  
# Помаранчевими лініями я хотіла показати наскільки багато маршрутів відбувається на щодень на Україні.  
# Поинти дозволяють зрозуміти лідерів по кількості відправлень + приїздів та співставити це з реальними містами  
# Ця візуалізація для загального розуміння картини, детальні дані з неї дізнатись важко.



```
test['fr']= test['from']

input_ratio1 = alt.binding_select(options = sorted(test.to.unique()))
input_ratio11 = alt.binding_select(options = sorted(test.fr.unique()))

select_city_to = alt.selection_single(name="FROM", fields = ['to'], bind = input_ratio1, empty = 'all', init =
select_city_from = alt.selection_single(name="TO", fields = ['fr'], bind = input_ratio11, empty = 'all', init =
```



```
ukraine = alt.Chart(ucor).project().mark_geoshape(color ='#b3cffc').encode(
 tooltip = alt.Tooltip('NAME_1:N'),
 #opacity = alt.condition
 detail = alt.Detail('NAME_1:N')
)

flow_f_to = alt.Chart(test).mark_rule(strokeCap = 'round', color ='orange', opacity= 0.9).encode(
 longitude = alt.Longitude('from_lon:Q'),
 latitude = alt.Latitude('from_lat:Q'),
 longitude2 = alt.Longitude2('to_lon:Q'),
 latitude2 = alt.Latitude2('to_lat:Q'), tooltip = alt.Tooltip('from:N')
).transform_filter(select_city_to).properties(
 title='Щоденні автобусні прибуття по Україні, 2020 р.',
 width=600,
 height=600
```



```
home_to = alt.Chart(test).mark_text().encode(
longitude='to_lon:Q',
latitude='to_lat:Q', text = alt.Text('to:N')).add_selection(select_city_to).transform_filter(select_city_t
```



```
flow_f_from = alt.Chart(test).mark_rule(strokeCap = 'round', color ='orange', opacity= 0.9).encode(
 longitude = alt.Longitude('from_lon:Q'),
 latitude = alt.Latitude('from_lat:Q'),
 longitude2 = alt.Longitude2('to_lon:Q'),
 latitude2 = alt.Latitude2('to_lat:Q'), tooltip = alt.Tooltip('to:N')
).transform_filter(select_city_from).properties(
 title='Щоденні автобусні відправлення по Україні, 2020 р.',
 width=600,
 height=600
```



```
points2_from = alt.Chart(test).mark_point(filled=True, color="black", size = 9).encode(
 longitude='to_lon:Q',
 latitude='to_lat:Q',
 size = alt.Size('count(to):Q', title= 'Кількість рейсів'),
 tooltip = alt.Tooltip('to')
).add_selection(select_city_from
).transform_filter(select_city_from
```


```

```
        width=600,  
        height=600  
    )  
  
points_to = alt.Chart(test).mark_point(filled=True, color="black", size=1000)
```

```
longitude='from_lon:Q',
latitude='from_lat:Q',
size = alt.Size('count(from):Q',title= 'Кількість рейсів'),
tooltip = alt.Tooltip('from')
).transform_filter(select_city_to
).properties(
    title='',
    width=600,
    height=600
)

home_from = alt.Chart(test).mark_text().encode(
longitude='from_lon:Q',
latitude='from_lat:Q', text = alt.Text('from:N') ).transform_filter(select_city_from)

pl1 = ukraine + flow_f_to + points_to + home_to
pl2 = ukraine + flow_f_from + points2_from + home_from

alt.hconcat(pl1.properties(width = 450, height = 300), pl2.properties(width = 100, height = 300))
```

Щоденні автобусні прибуття по Україні, 2020 р.

Щоденні автобусні відправлення по Україні, 2020 р.

TO_fr FROM_to

```
# Цією візуалізацією я хотіла показати міста відправлення - прибуття так, щоб їх можна було зробити вибраними. Людина може вибирати прибуття місто та подивитись те, куди з нього ідуть прямі рейси. Та навпаки, вибирати відправлення місто та подивитись, з чимими містами з цим містом є прямі рейси. Але я думаю, що це буде дуже складно зробити, оскільки в Україні багато міст, і вони не упорядковані. Тому я буду зробити це з дуже великим списком міст.
```

```
# З візуалізації можна зрозуміти, що автобуси, які виїждають з міста А в місто Б не ідуть у зворотному напрямку. Так як графіки прибуття - відправлення дуже відрізняються між собою.
```

```
# Також можна зробити висновок, що автобусні перевезення здебільшого двох типів:
```

```
# 1 переїзд у межах області та ії сусідів по периметру
```

```
In [11]: df2_m = df2_m[df2_m['end_t'] != 'прибуття ']
df2_m = df2_m[df2_m['l'] != 'Бобринець АС']
df2_m = df2_m[df2_m['start_t'].notna()]
df2_m = df2_m[df2_m['end_t'].notna()]
```

```
df2_m['l'] = df2_m['l'].apply(lambda x: int(x))
df2_m['start_t'] = df2_m['start_t'].apply(lambda x: int(x.split(':')[0]))
df2_m['end_t'] = df2_m['end_t'].apply(lambda x: int(x.split(':')[0]))
```



```
ch1= alt.Chart(df2_m).mark_rect().encode(
    x = alt.X('start_t:N', title='Час відправлення'),
    y = alt.Y('mean(l):Q', title= 'кількість відправлень'),
    #color = alt.Color('l:Q', scale = alt.Scale(scheme = 'pinkyellowgreen'))
).properties(width = 400,title = 'Залежність відстані маршруту від часу відправлення')

ch2=alt.Chart(df2_m).mark_rect().encode(
    x = alt.X('end_t:N', title='Час прибуття'),
    y = alt.Y('mean(l):Q', title = 'кількість відправлень'),
    #color = alt.Color('l:Q', scale = alt.Scale(scheme = 'pinkyellowgreen'))
).properties(width = 400,title = 'Залежність відстані маршруту від часу прибуття')

ch3 = alt.Chart(df2_m).mark_rect().encode(
    x = alt.X('start_t:N', title='Час відправлення'),
    y = alt.Y('count(start_t):Q', title= 'кількість відправлень'),
```

```
ch4 = alt.Chart(df2_m).mark_rect().encode(
    x = alt.X('end_t:N', title='Час прибуття'),
    y = alt.Y('count(end_t):Q', title = 'кількість відправлень'),
    #color = alt.Color('l:Q', scale = alt.Scale(scheme = 'pinkyellowgreen'))
).properties(width = 400,title = 'Розподіл часу прибуття')
```

The figure consists of two side-by-side bar charts. Both charts have a y-axis labeled 'Кількість' (Quantity) ranging from 350 to 500. The left chart is titled 'Залежність відстані маршруту від часу відправлення' (Relationship between route distance and departure time). It shows four bars representing different departure times: 10:00, 11:00, 12:00, and 13:00. The values are approximately 400, 400, 470, and 470 respectively. The right chart is titled 'Залежність відстані маршруту від часу прибуття' (Relationship between route distance and arrival time). It shows five bars representing different arrival times: 11:00, 12:00, 13:00, 14:00, and 15:00. The values are approximately 480, 470, 480, 460, and 450 respectively.

A bar chart titled 'Відповіді на перші питання' (Responses to the first questions). The y-axis is labeled 'кількість відпов.' (number of responses) and ranges from 150 to 250. The x-axis lists 15 categories. The bars show varying response counts, with the highest bar reaching approximately 260.

Категорія	Кількість відпов.
1	260
2	240
3	180
4	200
5	210
6	220
7	190
8	170
9	160
10	150
11	160
12	170
13	190
14	200
15	210
16	220
17	230
18	250
19	260

The figure consists of two side-by-side bar charts. The left chart, titled 'Час відправлення' (Departure time), shows the frequency of departures per hour from 0 to 23. The right chart, titled 'Час прибуття' (Arrival time), shows the frequency of arrivals per hour from 0 to 21. Both charts have a y-axis ranging from 0 to 100. The bars are blue.

The figure consists of two side-by-side bar charts. The left chart has a y-axis ranging from 400 to 550 with increments of 50. It has two bars: one at approximately 505 and another at approximately 425. The right chart has a y-axis ranging from 250 to 350 with increments of 50. It has five bars with heights of approximately 290, 310, 305, 310, and 240 respectively.

The figure consists of two side-by-side bar charts. Both charts have 'Кількість відправлень' (Number of messages) on the y-axis. The left chart's y-axis ranges from 0 to 350 with increments of 50. It has 10 bars representing friend counts from 0 to 9. The distribution is highly skewed, with the highest bar at 10 friends (~360 messages) and a long tail extending down to 0 friends (~150 messages). The right chart's y-axis ranges from 0 to 250 with increments of 50. It has 10 bars representing friend counts from 0 to 9. The distribution is also highly skewed, with the highest bar at 20 friends (~230 messages) and a long tail extending down to 0 friends (~120 messages).

Час відправлення	Кількість
0	10
1	10
2	10
3	20
4	98
5	100
6	100
7	100
8	100
9	100
10	100
11	100
12	100
13	100
14	100
15	100
16	100
17	105
18	105
19	70
20	45
21	35
22	35
23	35

```
# Розглянемо графіки залежності часу відправлення/прибуття та довжини поїздки  
# В мене була гіпотеза, що найдовші рейси відбуваються або до початку або після робочого дня  
# А приходять зранку наступного дня, або пізно ввечері.  
# Як ми бачимо, це підтвержується.
```

```
# Прибуття здебільшого посеред дня, рідше вночі.  
  
# Я хотіла зобразити розподіл, barchart добре підходять  
  
# Хотіла зробити графік зі слайдером (час поїздки) та на основі його будувати графіки частоти, але він може  
# показувати дані лише для конкретної відстані, не для інтервалу (100-200 км). Тому він не є коректним.
```

```
test = df2_m[df2_m['from_lat']!=0]
test = test[test['from_lon']!=0]
```

```
time_slider = alt.binding_range(min=0, max=24, step=1)
time_selection = alt.selection_single(bind=time_slider, fields=['start_t'], name="Start_time")

points= alt.Chart(test).mark_point(filled=True, color="black", size = 9).encode(
    longitude='to.lon:Q'
```

```
longitude='to_lon:Q',
latitude='to_lat:Q',
size=alt.Size('count(to)+count(from):Q', title='Кількість відправлень:'),
tooltip=alt.Tooltip('to')
).add_selection(time_selection).transform_filter(time_selection).properties(
    title='',
    width=600,
```

```
flows = alt.Chart(test).mark_rule(strokeCap = 'round', color ='orange', opacity= 0.3).encode(
    longitude = alt.Longitude('from_lon:Q'),
    latitude = alt.Latitude('from_lat:Q'),
    longitude2 = alt.Longitude2('to_lon:Q'),
    latitude2 = alt.Latitude2('to_lat:Q'),
```

```
    tooltip = alt.Tooltip(['from', 'to'])
).transform_filter(time_selection).properties(
    title='Щоденні автобусні відправлення по Україні, 2020 р.',
    width= 500,
    height=300
)
```

Out [42]:

Щоденні автобусні відправлення по Україні, 2020 р.

Кількість відправлень:

- 100
- 200
- 300
- 400

...

A network graph illustrating connections between nodes. The size of the black dots represents the value of the node, with a legend on the right indicating four levels: 400, 500, 600, and 700.

