Visualization with Altair

02/27/2020

Overview

- Introduction to Altair: Basic components of visualization
- Basic Charts:
  - Bar Chart
  - Line Chart
  - Scatterplot
- Practice and Exercise

Data, Library Installation

We are going to use the same nyc taxi dataset as what we explored last week.

Download it here: https://github.com/nyuvis/visual_analytics_course/blob/master/labs/lab2_Preprocessing/sampled_taxi.csv

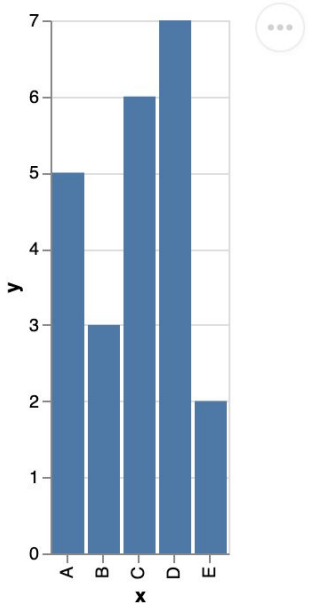
Install Altair and example datasets:

```
pip install altair vega_datasets
```

A sample plot

```
import altair as alt
import pandas as pd

data = pd.DataFrame({'x': ['A', 'B', 'C', 'D',
'E'],
                     'y': [5, 3, 6, 7, 2]})
alt.Chart(data).mark_bar().encode(
    x='x',
    y='y',
)
```



Basic Components

Data:
Initialize the visualization with data for plotting

Marks:
What visual attributes should be shown on screen.

```
alt.Chart(data).mark_bar()
.encode(
    x='x',
    y='y',
)
```

Encoding:
Map visual properties to data columns

Basic Component: Data

In general, we can initialize a visualization with

- pandas DataFrame
- altair Data object
- a url of json/csv file
- or geo information data (Geopandas, GeoDataFrame, etc.)

* We can also generate data for plotting. There are sequence generator, sphere generator, etc.

Wide-form and Long-form Data

```
1  wide_form = pd.DataFrame({'Date': ['2007-10-01', '2007-11-01', '2007-12-01'],
2                            'AAPL': [189.95, 182.22, 198.08],
3                            'AMZN': [89.15, 90.56, 92.64],
4                            'GOOG': [707.00, 693.00, 691.48]})
5  print(wide_form)
```

	Date	AAPL	AMZN	GOOG
0	2007-10-01	189.95	89.15	707.00
1	2007-11-01	182.22	90.56	693.00
2	2007-12-01	198.08	92.64	691.48

```
10 print(long_form)
```

	Date	company	price
0	2007-10-01	AAPL	189.95
1	2007-11-01	AAPL	182.22
2	2007-12-01	AAPL	198.08
3	2007-10-01	AMZN	89.15
4	2007-11-01	AMZN	90.56
5	2007-12-01	AMZN	92.64
6	2007-10-01	GOOG	707.00
7	2007-11-01	GOOG	693.00
8	2007-12-01	GOOG	691.48

Just like many other plotting libraries, altair works better with long-form data where each row is a single observation.

However, we usually have wide-form data in most datasets.

Converting B/w Long-form and Wide-form Data

From wide to long:

```
wide_form.melt('Date', var_name='company', value_name='price')
```

From long to wide:

```
long_form.pivot(index='Date', columns='company', values='price').reset_index()
```

* Check more details about melt and pivot in pandas documentation.

Basic Components: Marks

Altair provides a number of basic mark properties →

* Full list of marks:
https://altair-viz.github.io/user_guide/marks.html

** Example Visualization Gallery:
https://altair-viz.github.io/gallery/index.html

Mark Name	Method	Description
area	`mark_area()`	A filled area plot.
bar	`mark_bar()`	A bar plot.
circle	`mark_circle()`	A scatter plot with filled circles.
geoshape	`mark_geoshape()`	A geographic shape
image	`mark_image()`	A scatter plot with image markers.
line	`mark_line()`	A line plot.
point	`mark_point()`	A scatter plot with configurable point shapes.
rect	`mark_rect()`	A filled rectangle, used for heatmaps
rule	`mark_rule()`	A vertical or horizontal line spanning the axis.
square	`mark_square()`	A scatter plot with filled squares.
text	`mark_text()`	A scatter plot with points represented by text.
tick	`mark_tick()`	A vertical or horizontal tick mark.

Basic Components: Encodings

Encoding Data Type:

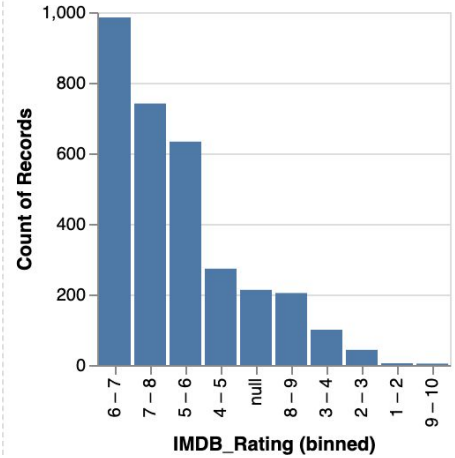
Data Type	Shorthand Code	Description
quantitative	`Q`	a continuous real-valued quantity
ordinal	`O`	a discrete ordered quantity
nominal	`N`	a discrete unordered category
temporal	`T`	a time or date value
geojson	`G`	a geographic shape

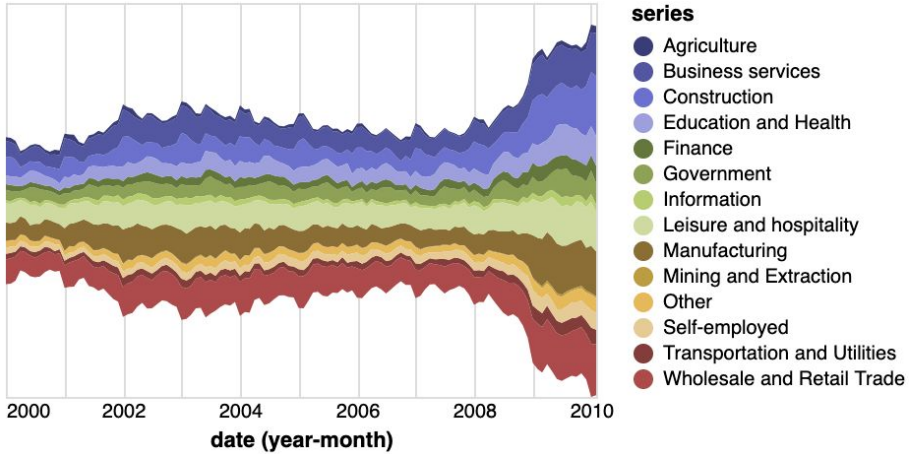
Channels can be customized:

- Position Channels: x, y, longitude, latitude, etc.
- Mark Property Channels: color, opacity, shape, size, stroke, etc.
- etc.

Encoding Channel Options

"Each encoding channel allows for a number of additional options to be expressed; these can control things like axis properties, scale properties, headers and titles, binning parameters, aggregation, sorting, and many more."



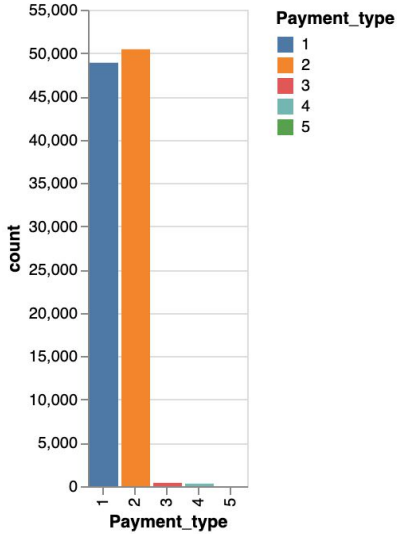


Check more details: https://altair-viz.github.io/user_guide/encoding.html

Some encoding can be used frequently

- Define the sorting of x/y axis.
  - x = alt.X('day:N', sort='ascending')
- Define the scaling of x/y axis
  - y = alt.Y('fare_amount:Q', scale=alt.Scale(type='linear', domain=[10, 10000])),
- Define the color the marks
  - color = 'color:N'

```
alt.Chart(df.groupby('Payment
_type').size()
      .reset_index(name='count
'))
.mark_bar()
.encode(
      x='Payment_type:N',
      y='count:Q',
      color='Payment_type:N'
)
```



Question Answering Time!

Let’s go back to the questions we answered last week.

Question 1:

How do the payment types compare in terms of number of trips?

Bar Chart: Count the occurance

Approach 1: plot transformed data

- Step 1: generate data for plotting

```
to_plot = df['Payment_type'].value_counts() or
to_plot2 = df.groupby('Payment_type').size()
```

- Step 2: transform `to_plot` to a DataFrame

```
to_plot = to_plot.rename_axis('Payment_type').reset_index(name='count') or
to_plot2 = to_plot2.reset_index(name='count')
```

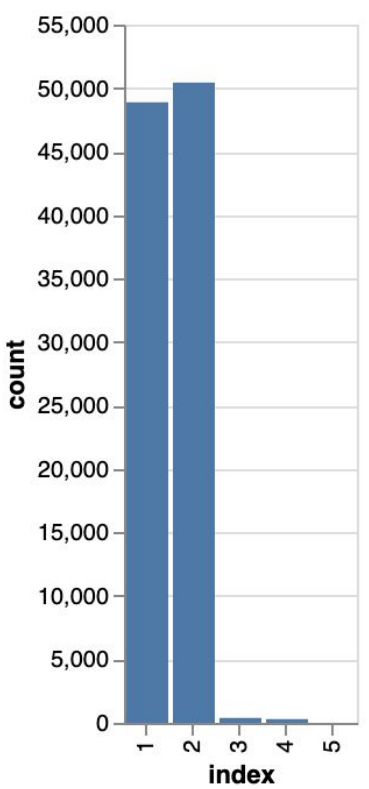
- Step 3: plot

```
alt.Chart(to_plot).mark_bar().encode(
   x='Payment_type:N',
   y='count:Q'
)
```

Q1: How do the payment types compare in terms of number of trips?

* Of course you can combine the three steps into one such as:

```
alt.Chart(df.groupby('Payment_type').size()
     .reset_index(name='count'))
.mark_bar()
.encode(...)
```



Q1: How do the payment types compare in terms of number of trips?

We can set y-axis as an aggregation value.

Class `altair.Y` allows more detailed descriptions of what to be shown for y-positions.

For Example,

```
alt.Y(aggregate='count', type='quantitative')
```

Or `alt.Y('count():Q')`

* a shorthand string contains information of field, aggregate, and type

Bar Chart: Count the occurance

Approach 2: transform data during plotting

```
alt.Chart(df).mark_bar().encode(
    x='Payment_type:N',
    y='count(Payment_type):Q'
)
```

You will get a MaxRowsError after running the code above directly.

Workaround:

- Run `alt.data_transformers.enable(max_rows=100000)` before plotting, or
- Run `alt.data_transformers.disable_max_rows()` . Be careful with this operation.

Bar Chart: Count the occurance

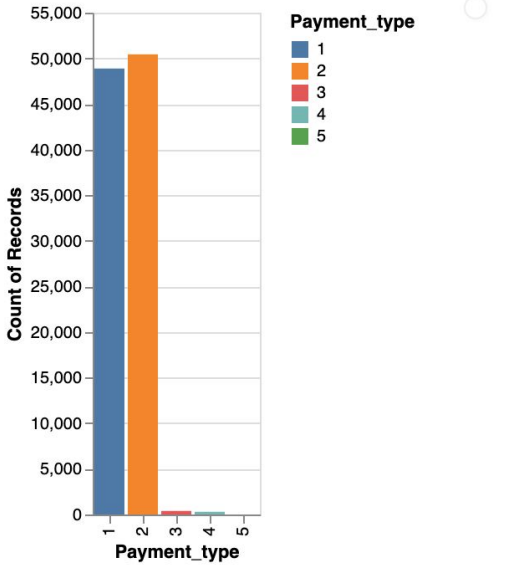
Approach 2: transform data during plotting

```
alt.Chart(df).mark_bar().encode(
    alt.X('Payment_type:N', sort=alt.EncodingSortField(field= 'Payment_type',
op='count', order='descending')),
    alt.Y(field= 'Payment_type', aggregate='count', type='quantitative')
)
```

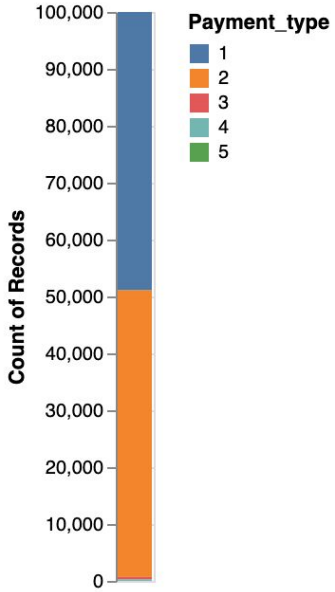
Compare the visualization results here and the one generated from the code in the previous page. What are the differences in the plot?

Bar chart, stacked bar chart, grouped bar chart

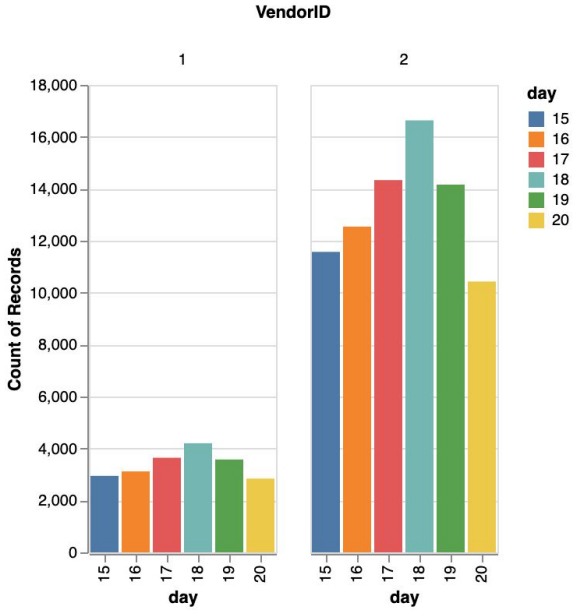
```
alt.Chart(df).mark_bar().encode(
    x='Payment_type:N',
    y='count(Payment_type):Q',
    color= 'Payment_type:N',
)
```




```
alt.Chart(df).mark_bar().encode(
    y='count(Payment_type):Q',
    color= 'Payment_type:N',
)
```

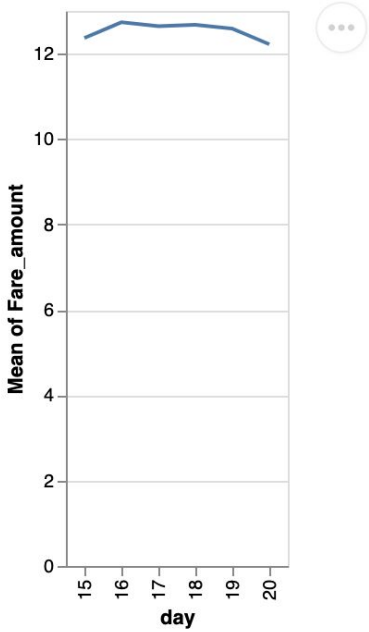



```
alt.Chart(df).mark_bar().encode(
    y='count(Fare_amount):Q',
    x='day:N',
    color= 'day:N',
    column='VendorID:N',
)
```

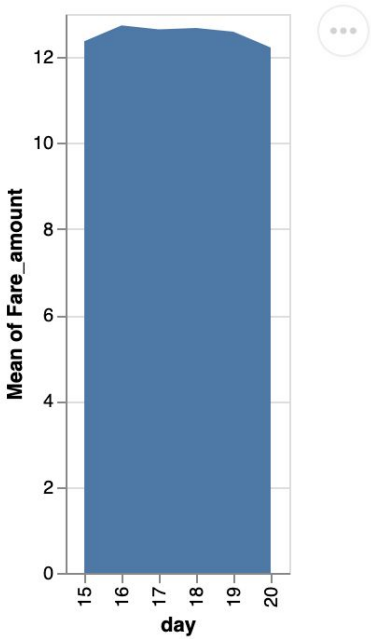


Line Chart

```
alt.Chart(df).mark_line().encode(
    y='mean(Fare_amount):Q',
    x='day:N',
)
```

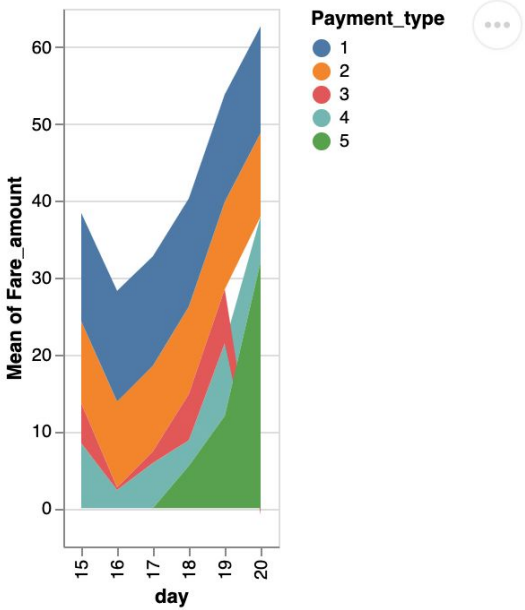



```
alt.Chart(df).mark_area().encode(
    y='mean(Fare_amount):Q',
    x='day:N',
)
```

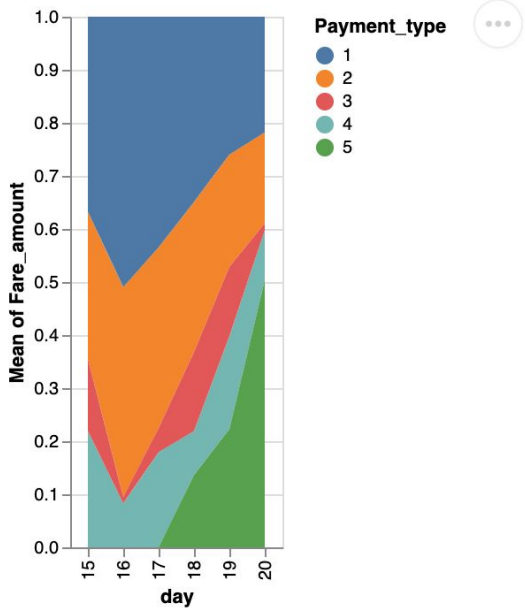


Stacked Line Chart

```
alt.Chart(df).mark_area().encode(
    y=alt.Y('mean(Fare_amount):Q',),
    x='day:N',
    color= 'Payment_type:N',
)
```

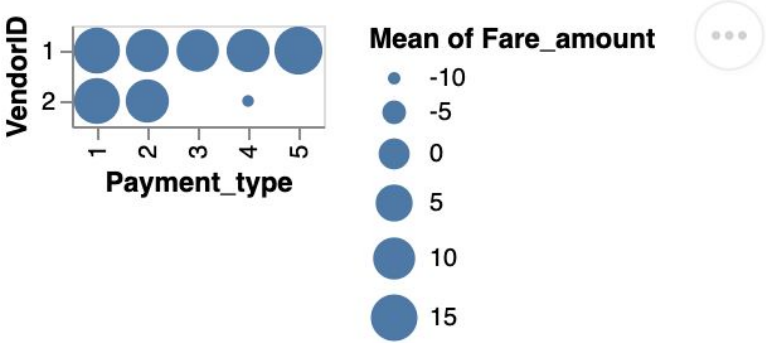



```
alt.Chart(df).mark_area().encode(
    y=alt.Y('mean(Fare_amount):Q', stack='normalize'),
    x='day:N',
    color= 'Payment_type:N',
)
```

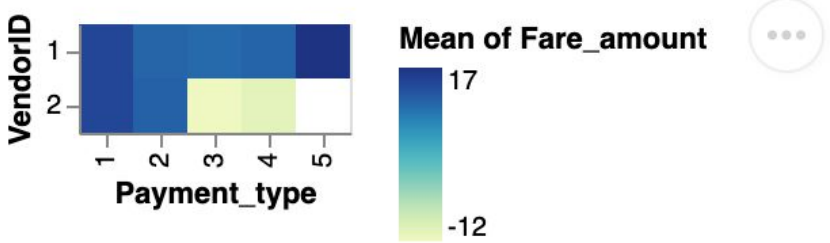


Matrix, Heatmap

```
alt.Chart(df).mark_circle().encode(
    y='VendorID:N',
    x='Payment_type:N',
    size= 'mean(Fare_amount)',
)
```




```
alt.Chart(df).mark_rect().encode(
    y='VendorID:N',
    x='Payment_type:N',
    color= 'mean(Fare_amount)',
)
```



Practice 1:

Try to plot:

How does the number of trips change over time on June 16 (by hour)?

Which mark should I use?

Practice 2:

How are the vendors different from each other in terms of the **average** trip distance every day?

- Which mark should I use?
- How to differentiate between two vendors?
- Should I show the distance from zero?

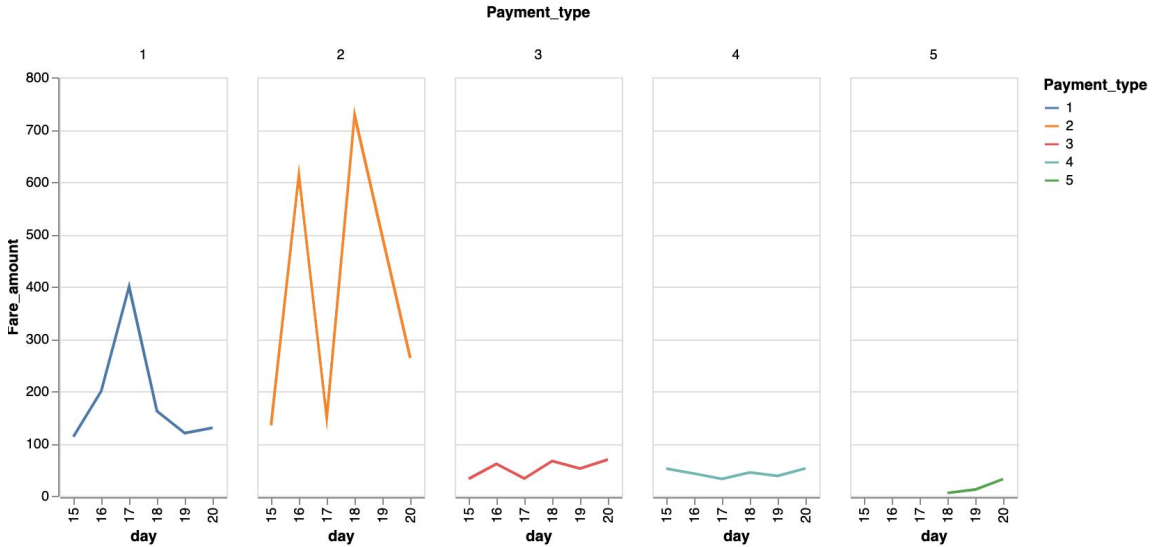
Practice 3

How are the payment type different from each other in terms of the **max** fare amount of each type per day?

Multiple charts

Add one encoding rule to plot multiple charts in a row:

```
column= 'Payment_type:N',
```

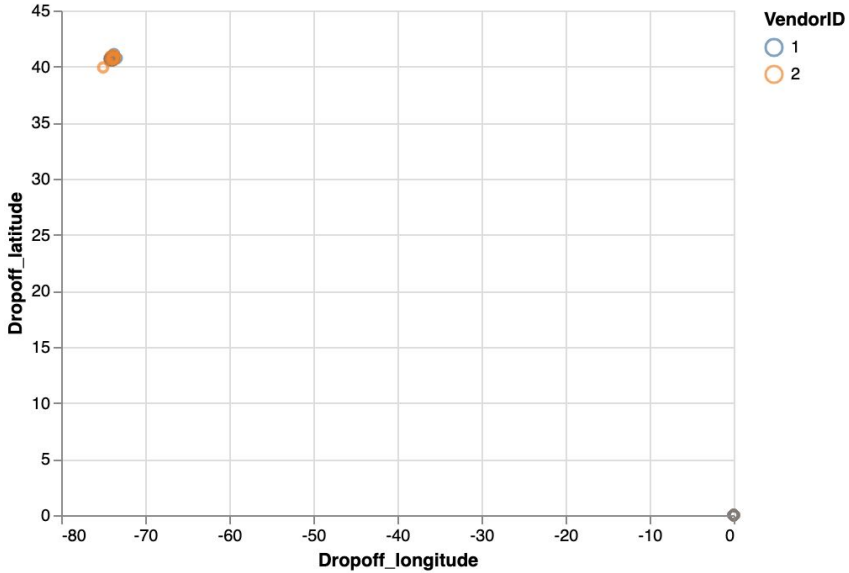


Practice 4: scatterplot

How are the vendors different from each other in terms the drop off locations (longitude, latitude) distributed on June 17?

- What do you think the visualization will look like?
- How does your scatter plot look like?

I got this...



How should I interpret this visualization?

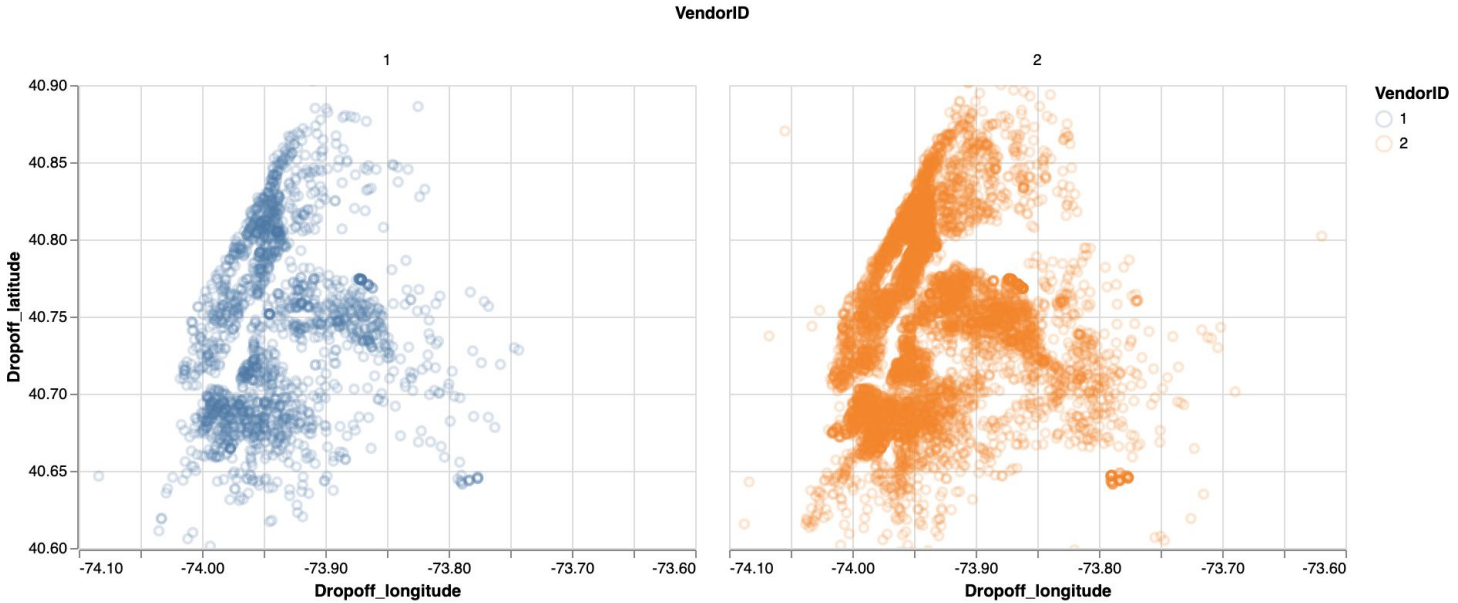
Visualization Customization

- Adjust Axis Limits
  - set the domain of scales for x and y axis (revisit page 12)
  - deal with marks beyond the scale (remove them, or show at the edge)
    - Try `.mark_point(clip=True)`
    - Or `scale=alt.Scale(domain=(min, max), clamp=True)`
- Set the opacity of marks when there are too many to be shown
  - set the opacity in encoding: `opacity=alt.value(.2)`

Re-render the scatterplot

Set the limit of x-axis: [-74.1, -73.6], y-axis: [40.6, 40.9]

Can you observe the contour of Manhattan?



Practice

Film Permits: https://data.cityofnewyork.us/City-Government/Film-Permits/tg4x-b46p

Motor Vehicle Collisions: https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95

Restaurant Inspections:
https://data.cityofnewyork.us/Health/DOHMH-New-York-City-Restaurant-Inspection-Results/43nn-pn8j

NYC Jobs: https://data.cityofnewyork.us/City-Government/NYC-Jobs/kpav-sd4t