Worksheet 5

This worksheet is due Monday night of the following week. You are encouraged to work in groups of up to 3 total students, but each student should submit their own file. (It's fine for everyone in the group to upload the same file.)

These questions refer to the attached vending machines csv file, `data_groups.csv`. This is a very nice (artificial) dataset for demonstrating the usefulness of data visualization.

Put the full names of everyone in your group (even if you're working alone) here. (This makes grading easier.)

- **Names**: Ilyas

Part 1 - Without data visualization

- Load the data using pandas as a DataFrame stored as `df` and look at the first few rows.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv("../Data/data_groups.csv")
df.head()
```

Out[2]:

	grouping	x	y
0	2	57.614674	33.822448
1	2	67.881720	24.236018
2	1	42.820500	32.564100
3	8	72.998186	59.665645
4	5	78.249916	17.883241

In [3]:
```
groups = sorted(df["grouping"].unique())

for gp in groups:
    df_sub = df[df["grouping"] == gp]
    print("The group is", gp) # if you know f-strings, those could be used instead
    print("The mean of x is", df_sub["x"].mean(),
            "The standard deviation of x is", df_sub["x"].std())
    print("The mean of y is", df_sub["y"].mean(),
            "The standard deviation of y is", df_sub["y"].std())
    print()
```

```
The group is 0
The mean of x is 54.267848823661964 The standard deviation of x is 16.76675894771805
The mean of y is 47.83589633112676 The standard deviation of y is 26.936104931679978

The group is 1
The mean of x is 54.26327323943663 The standard deviation of x is 16.765142039116792
The mean of y is 47.832252816901416 The standard deviation of y is 26.935403486939116

The group is 2
The mean of x is 54.26734110478873 The standard deviation of x is 16.768959216194453
The mean of y is 47.83954522535211 The standard deviation of y is 26.93027468808843

The group is 3
The mean of x is 54.26992723091549 The standard deviation of x is 16.769958611325386
The mean of y is 47.836987988408445 The standard deviation of y is 26.937683806980512

The group is 4
The mean of x is 54.26880527950704 The standard deviation of x is 16.766704015934764
The mean of y is 47.83545020401409 The standard deviation of y is 26.939997961411027

The group is 5
The mean of x is 54.268730022394365 The standard deviation of x is 16.769239493454403
The mean of y is 47.83082315530282 The standard deviation of y is 26.935726689918788

The group is 6
The mean of x is 54.265881785422536 The standard deviation of x is 16.768852670828494
The mean of y is 47.831495652323945 The standard deviation of y is 26.93860807087184

The group is 7
The mean of x is 54.26015033415494 The standard deviation of x is 16.76995769550748
The mean of y is 47.8397172794507 The standard deviation of y is 26.93000168716234

The group is 8
The mean of x is 54.26691630119718 The standard deviation of x is 16.769999617573024
The mean of y is 47.83160198797183 The standard deviation of y is 26.937901927731797

The group is 9
The mean of x is 54.26144178316901 The standard deviation of x is 16.765897903899337
The mean of y is 47.830251913661975 The standard deviation of y is 26.93987622043797

The group is 10
The mean of x is 54.266099784295776 The standard deviation of x is 16.769824954043756
The mean of y is 47.83472062494366 The standard deviation of y is 26.9397434192671

The group is 11
The mean of x is 54.26030345169014 The standard deviation of x is 16.767735488473807
The mean of y is 47.83982920901408 The standard deviation of y is 26.93019151853346

The group is 12
The mean of x is 54.26731970598592 The standard deviation of x is 16.76001265980608
```

```
The mean of y is 47.837717267253524 The standard deviation of y is 26.9300360878382
```

- Here is another way to get the same `mean` information. Look over this code and its output and the output above, and see how they're related. (Again, you don't need to write anything for this part.)

In [4]:
```
df.groupby("grouping").mean()
```

Out[4]:

grouping	x	y
0	54.267849	47.835896
1	54.263273	47.832253
2	54.267341	47.839545
3	54.269927	47.836988
4	54.268805	47.835450
5	54.268730	47.830823
6	54.265882	47.831496
7	54.260150	47.839717
8	54.266916	47.831602
9	54.261442	47.830252
10	54.266100	47.834721
11	54.260303	47.839829
12	54.267320	47.837717

- How do you think you get the same standard deviation information using `groupby`? Try it, and check that the information does match what we got using the for loop.

In [5]:
```
df.groupby("grouping").std()
```

Out[5]:

grouping	x	y
0	16.766759	26.936105
1	16.765142	26.935403
2	16.768959	26.930275
3	16.769959	26.937684
4	16.766704	26.939998
5	16.769239	26.935727
6	16.768853	26.938608
7	16.769958	26.930002
8	16.770000	26.937902
9	16.765898	26.939876
10	16.769825	26.939743
11	16.767735	26.930192
12	16.760013	26.930036

- In a markdown cell, describe, what is the main takeaway from the above outputs, in terms of what they tell you about the dataset? Feel free to make more computations if they are helpful.

The outputs tell you the average value for each group, along with the spread of the group, the mean and standard deviation.

Part 2: Visualizing the data all at once

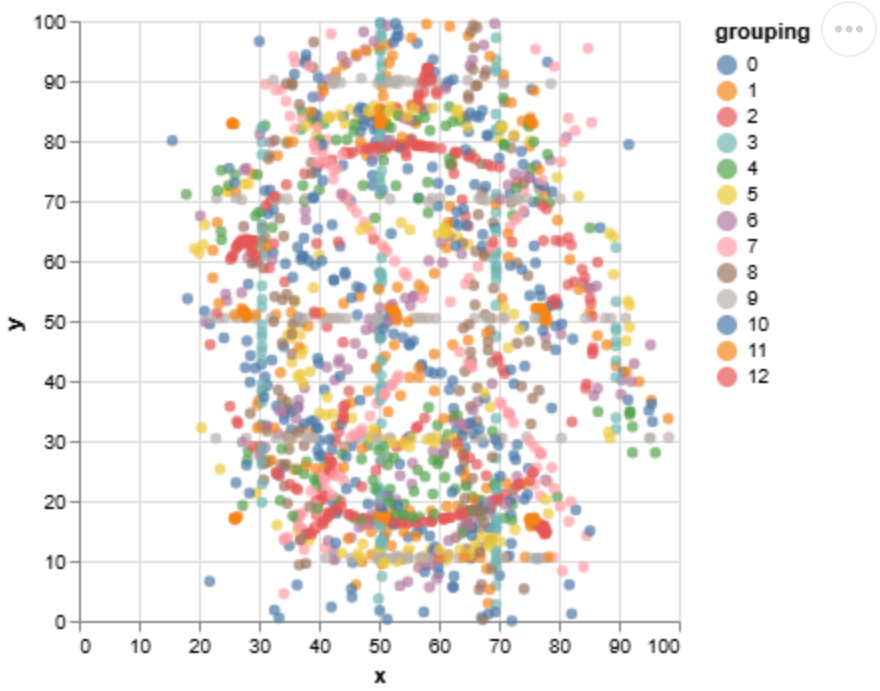
- Using Altair, plot the data in `df` using a scatter plot (`mark_circle`). Encode the "x" column in the x-axis, the "y" column in the y-axis, and the "grouping" in the color. Use `:N` after specifying "grouping" to tell Altair that these groupings are categories, not quantitative values.

(The resulting plot should look like a mess.)

In [6]:
```
import altair as alt
import seaborn as sns
```

In [7]:
```
alt.Chart(df).mark_circle().encode(x = "x", y = "y", color = "grouping:N")
```

Out[7]:



Part 3: Visualizing the 13 groupings individually

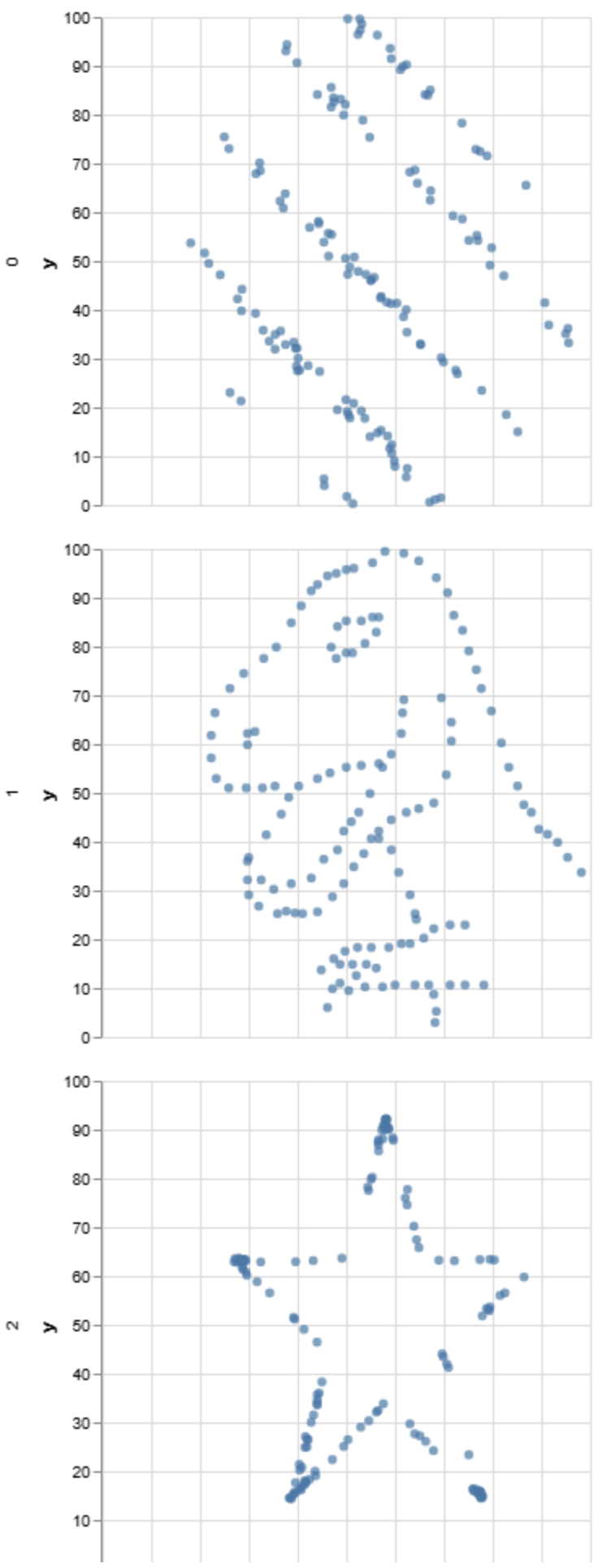
We will see that these 13 groups of data are fundamentally different by plotting them separately. We don't have to do this by hand. Altair, Seaborn, and Plotly Express all have their own way to do this separate plotting automatically.

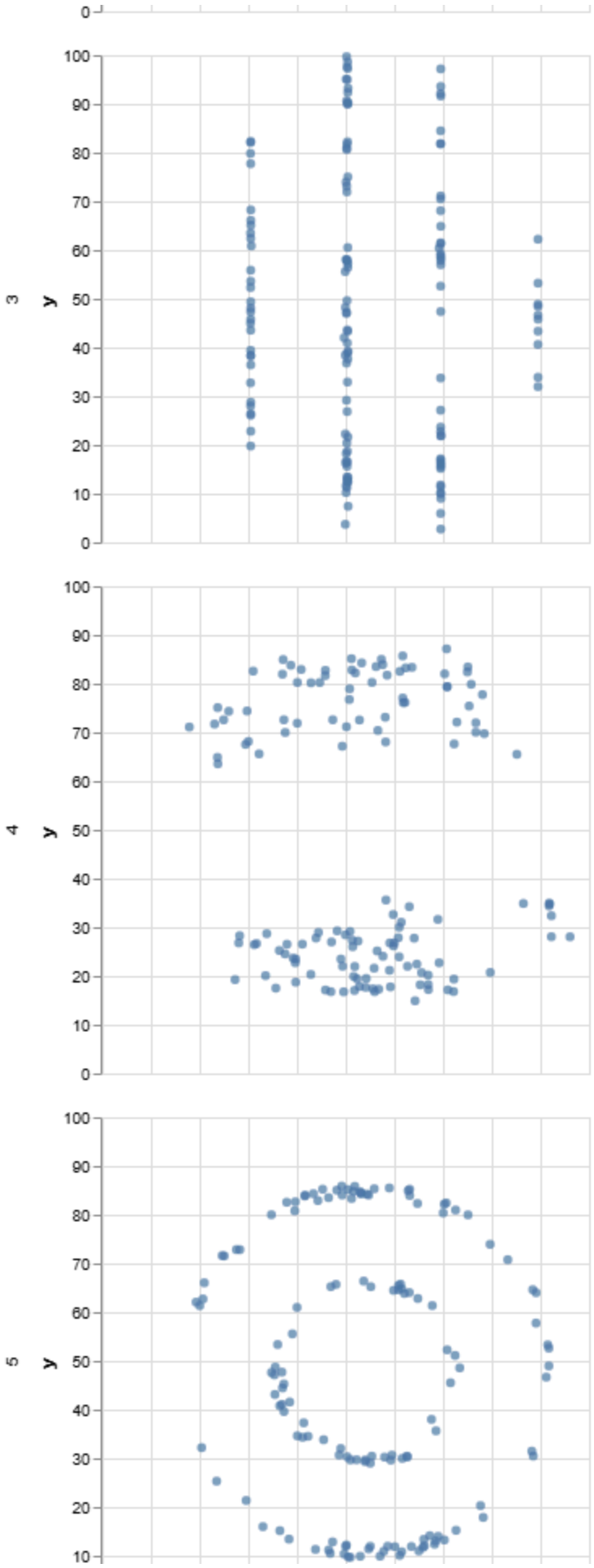
Using Altair

- Make a facet chart in Altair by taking the same `Chart` definition as above, and including `row="grouping:N"` within the encoding. This tells Altair to put each grouping in its own row. Reference 1 (but don't look at the first for loop example, scroll below that). Reference 2

In [8]:
```
alt.Chart(df).mark_circle().encode(x = "x", y = "y", row = "grouping:N")
```

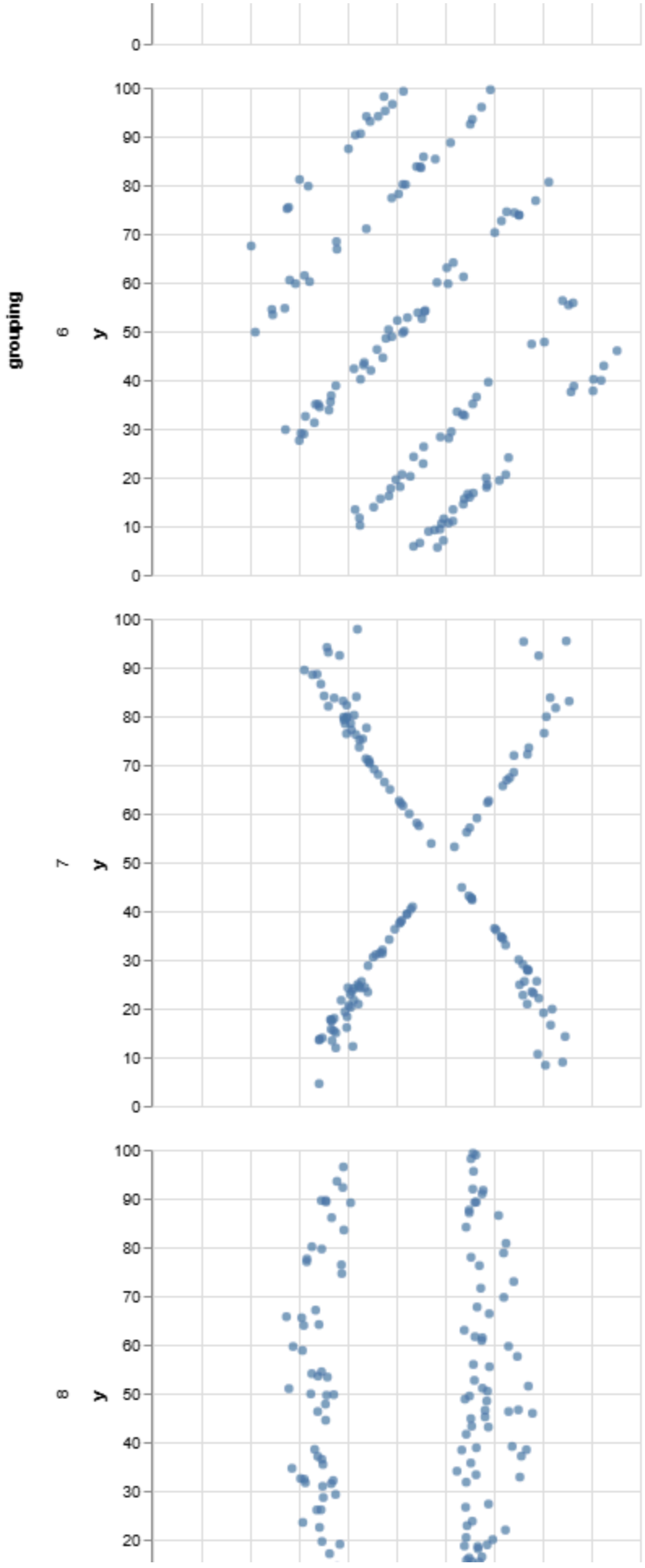
Out[8]:

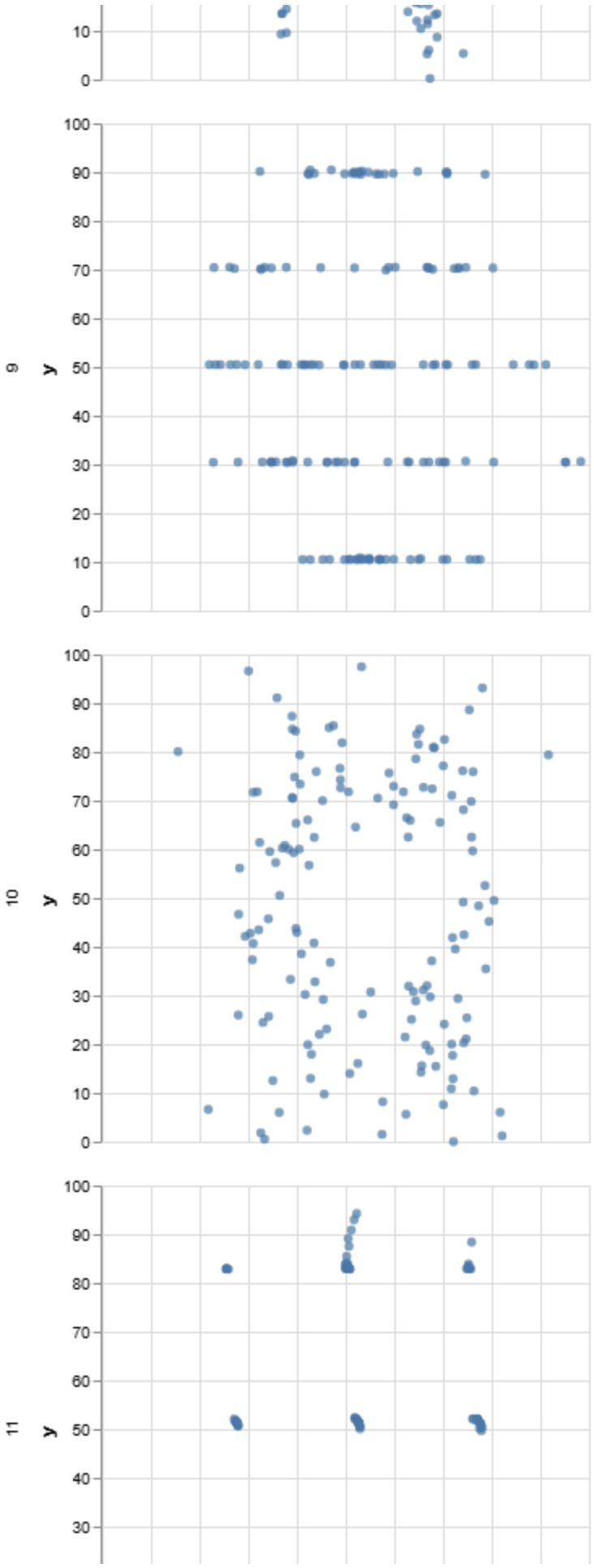

0
y
100
90
80
70
60
50
40
30
20
10
0
1
y
100
90
80
70
60
50
40
30
20
10
0
2
y
100
90
80
70
60
50
40
30
20
10

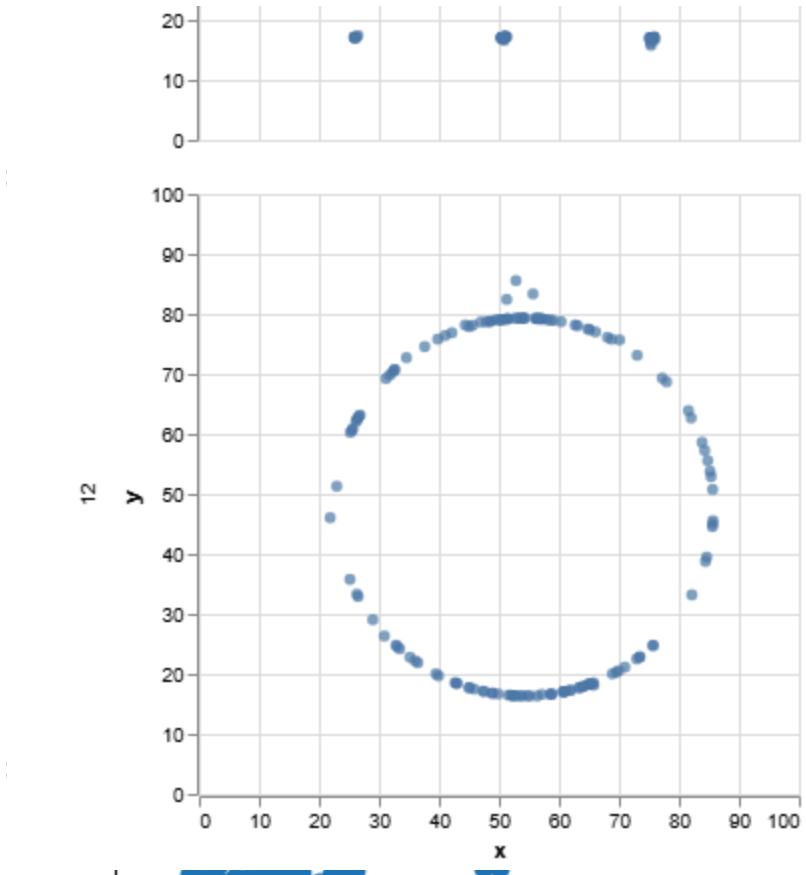

3
y
4
y
5
y

(Scroll down to the
ould use `df` instead of
used with
e used with `map`).

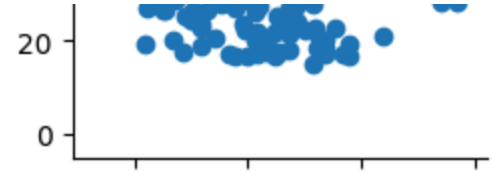
In [9]:



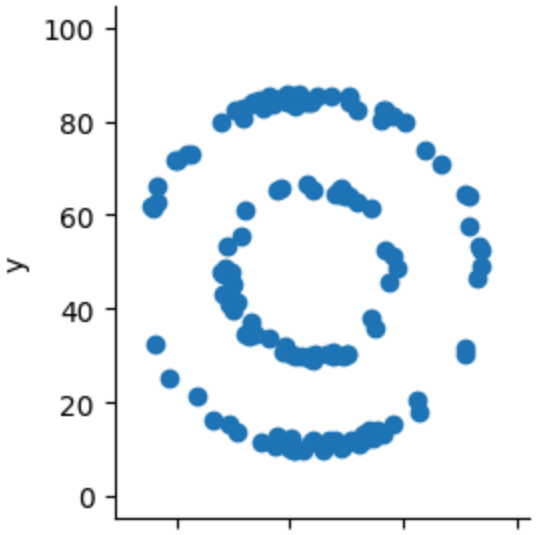

9
10
11
y
0
10
20
30
40
50
60
70
80
90
100



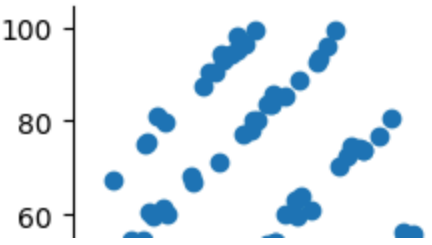
12

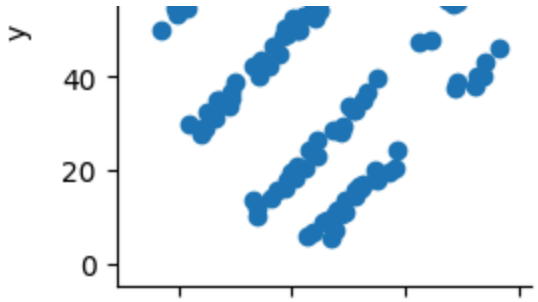


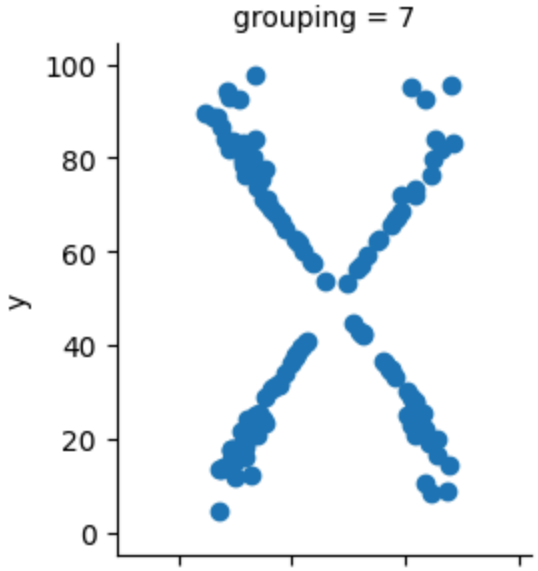
grouping = 5

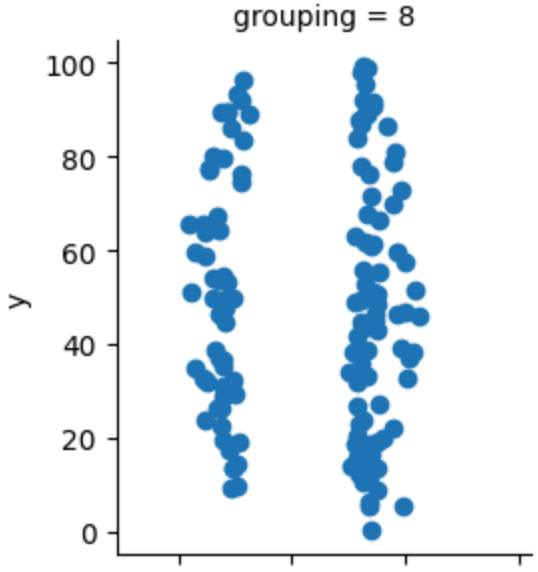


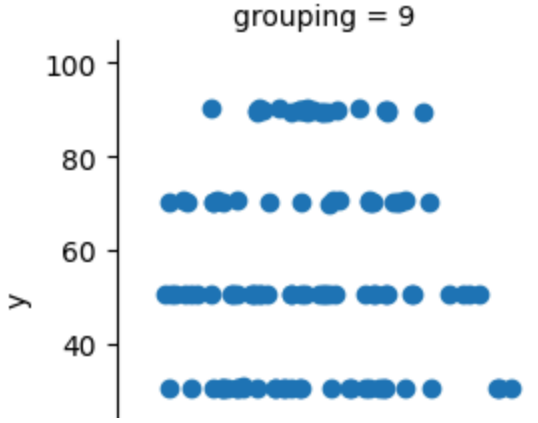
grouping = 6

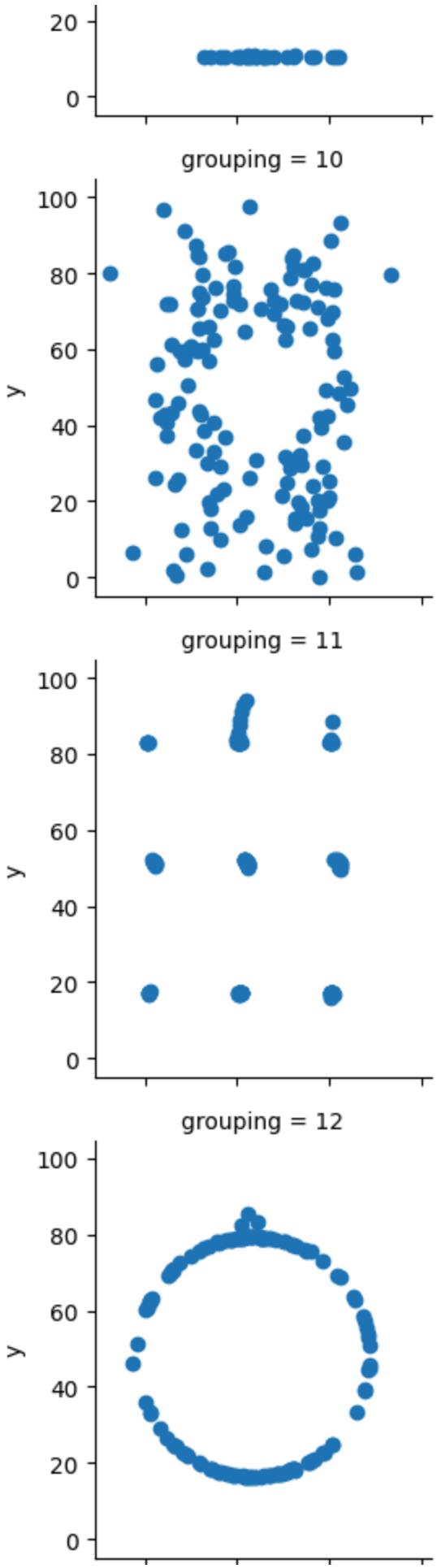


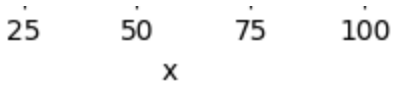

y
40
20
0


grouping = 7
y
100
80
60
40
20
0


grouping = 8
y
100
80
60
40
20
0


grouping = 9
y
100
80
60
40

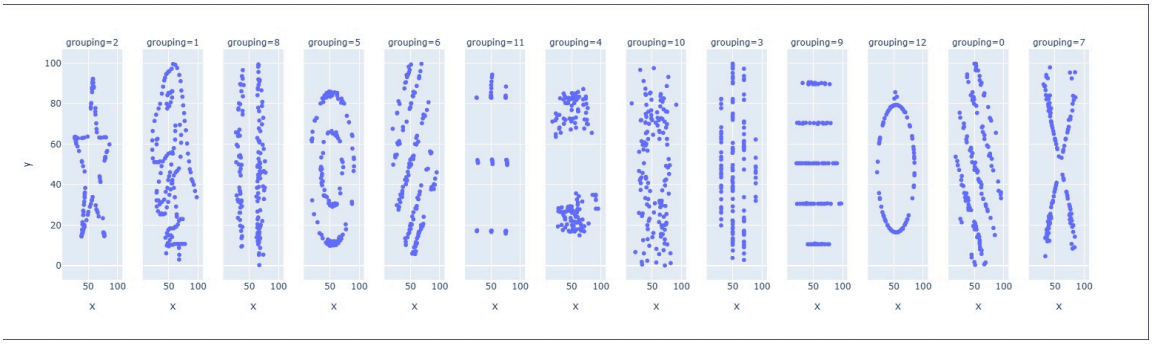

20
0
grouping = 10
100
80
60
40
20
0
y
grouping = 11
100
80
60
40
20
0
y
grouping = 12
100
80
60
40
20
0
y



Using Plotly Express

- Make the same facet chart using Plotly Express. Reference (To get it to look similar to the Altair and Seaborn versions, I added the keyword argument `height=4000` to the `px.scatter` function.)

In [10]:
```
import plotly.express as px
px.scatter(df, "x", "y", facet_col="grouping")
```



The plot isn't displaying when I export the notebook, so I've included a screenshot

Submission

- Using the `Share` button at the top right, enable public sharing, and enable Comment privileges. Then submit the created link on Canvas.