

삼성전자 AI전문가 교육과정 사전교육 - 데이터 분석 및 시각화

Altair 실습 4 - composition

요약

- 하나의 시각화에 대한 인코딩, 축, 마크, 데이터 변환, 인터랙션을 적용하기
- 여러 개의 시각화를 동시에 볼 수 있는?
- Tableau와 Spotfire의 “대시보드”
- https://altair-viz.github.io/user_guide/compound_charts.html

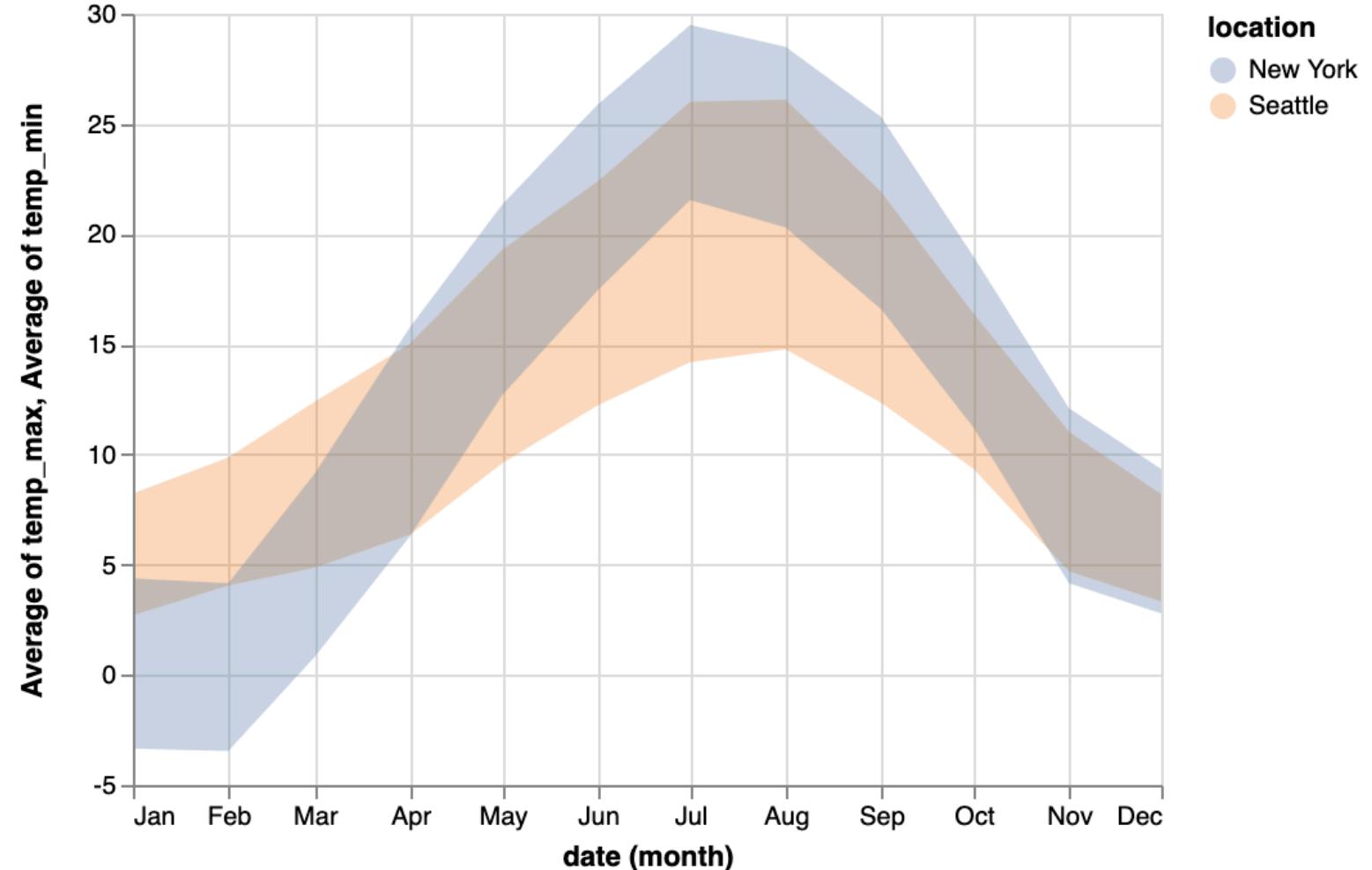
Visualization Composition Grammar

- Layer (겹치기)
- Facet
- Repeat (반복하기)
- Concat (병치하기, juxtaposition)

Visualization Composition Grammar

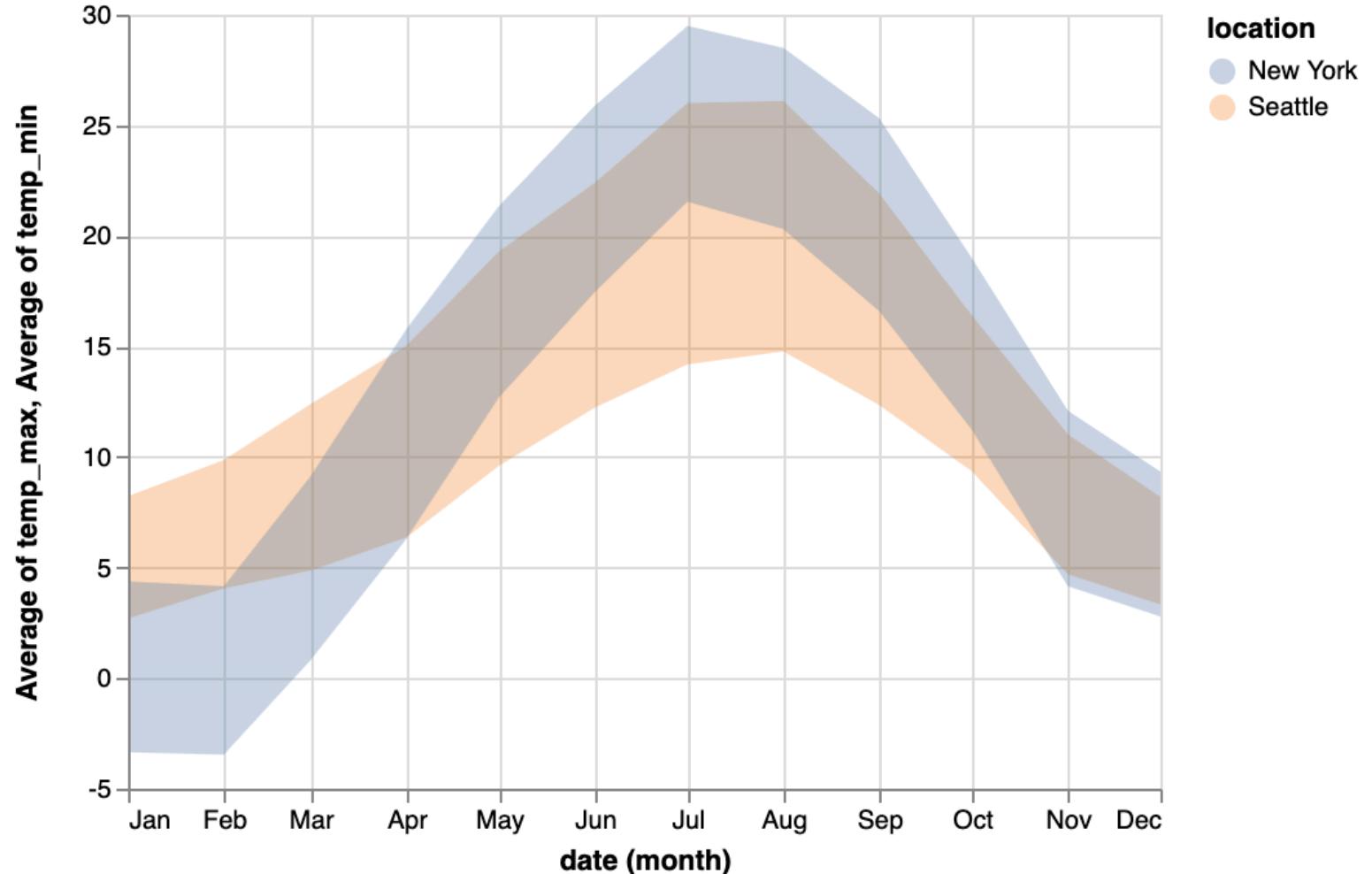
- Layer: 두 시각화를 겹친다.
- Facet: 데이터를 특정 필드 값으로 나누고 시각화를 여러 개 그린다.
- Repeat: 인코딩을 바꿔가면서 시각화를 여러 개 그린다.
- Concat: 시각화 두개를 가로/세로로 나란히 놓는다.

Layering?



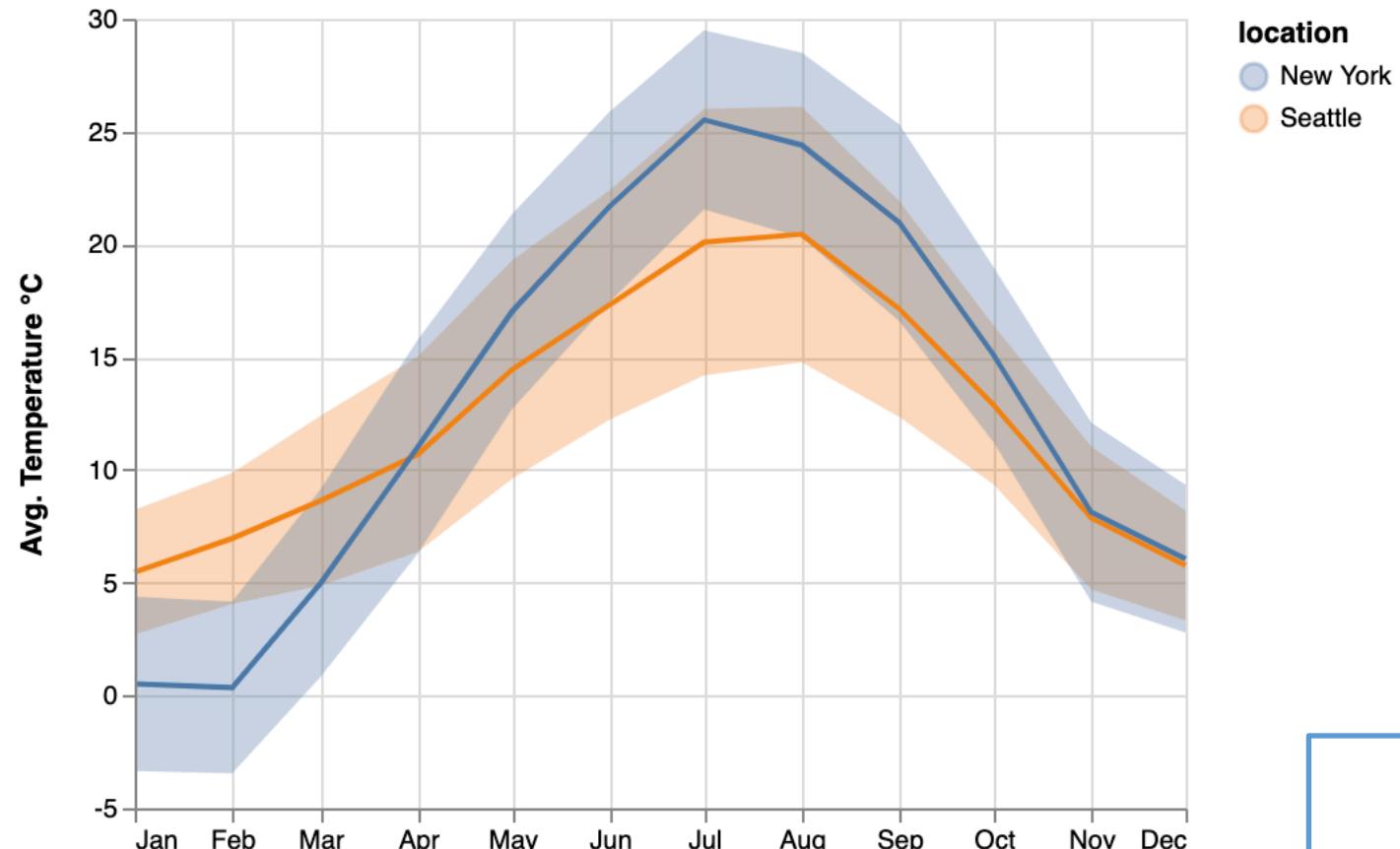
Layering? 아님

- 그냥 색상 채널에 location을 매핑한 것



Layering? 아님

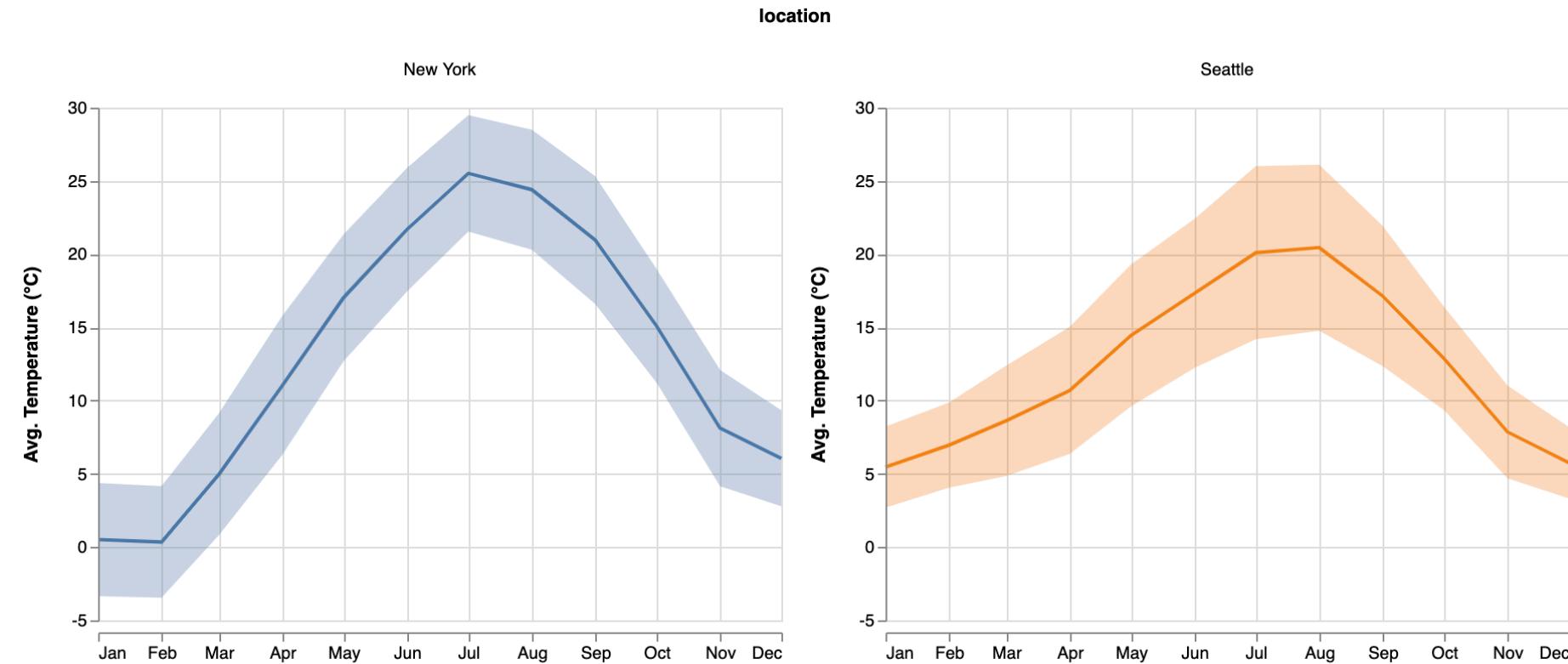
- 주황 차트와 파란 차트가 겹쳐진 것이 아닙니다.
- 영역 차트와 라인 차트가 겹쳐진 것입니다 (= 다른 인코딩).



요약

- Layer: 두 시각화를 겹친다.
 - 인코딩: 적어도 하나의 축은 공유가 되어야!
 - 데이터: 모든 데이터를 보여준다.
- Facet: 데이터를 특정 필드 값으로 나누고 시각화를 여러 개 그린다.
 - 인코딩: 인코딩이 같은 시각화가 여러 개가 생긴다.
 - 데이터: 특정 필드로 데이터를 나누기 때문에 각 시각화는 데이터 중 일부를 중복없이 보여준다.
- Repeat: 인코딩을 바꿔가면서 시각화를 여러 개 그린다.
 - 인코딩: 인코딩이 다른 시각화가 여러 개가 생긴다 (x , y 축이 달라짐).
 - 데이터: 각 시각화는 모든 데이터를 보여준다.
- Concat: 시각화 두개를 가로/세로로 나란히 놓는다.
 - 인코딩/데이터: 자유롭게 가능.

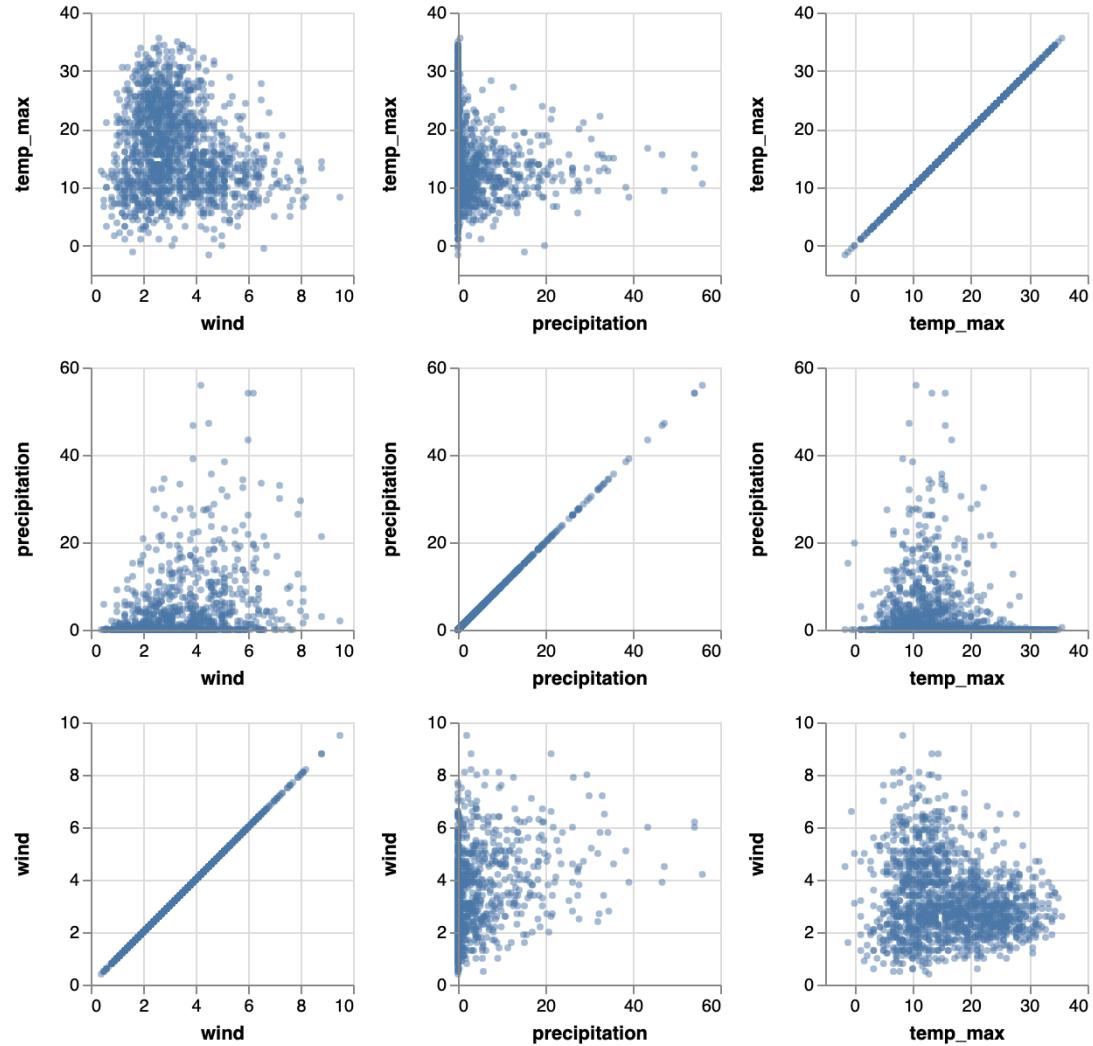
Facet



요약

- Layer: 두 시각화를 겹친다.
 - 인코딩: 적어도 하나의 축은 공유가 되어야!
 - 데이터: 모든 데이터를 보여준다.
- Facet: 데이터를 특정 필드 값으로 나누고 시각화를 여러 개 그린다.
 - 인코딩: 인코딩이 같은 시각화가 여러 개가 생긴다.
 - 데이터: 특정 필드로 데이터를 나누기 때문에 각 시각화는 데이터 중 일부를 중복없이 보여준다.
- Repeat: 인코딩을 바꿔가면서 시각화를 여러 개 그린다.
 - 인코딩: 인코딩이 다른 시각화가 여러 개가 생긴다 (x , y 축이 달라짐).
 - 데이터: 각 시각화는 모든 데이터를 보여준다.
- Concat: 시각화 두개를 가로/세로로 나란히 놓는다.
 - 인코딩/데이터: 자유롭게 가능.

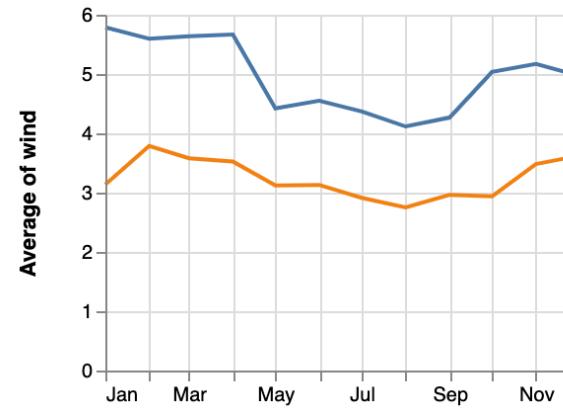
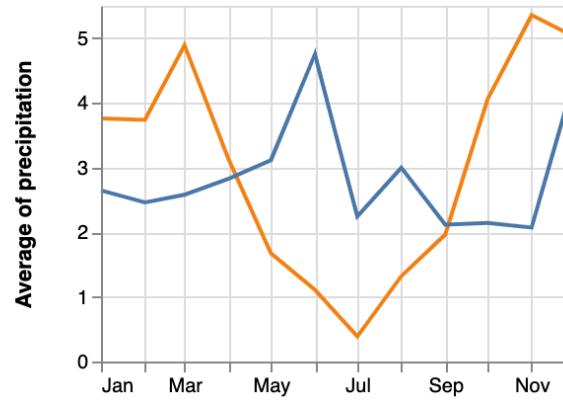
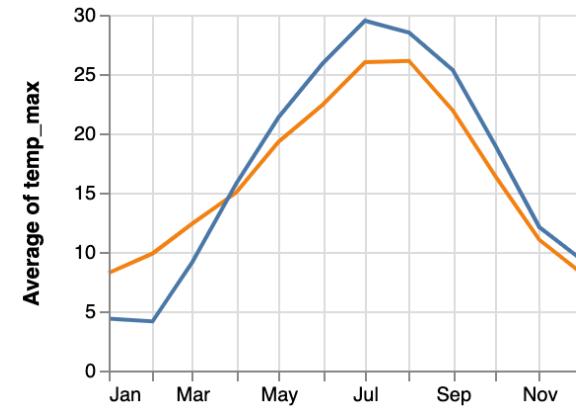
Repeat



요약

- Layer: 두 시각화를 겹친다.
 - 인코딩: 적어도 하나의 축은 공유가 되어야!
 - 데이터: 모든 데이터를 보여준다.
- Facet: 데이터를 특정 필드 값으로 나누고 시각화를 여러 개 그린다.
 - 인코딩: 인코딩이 같은 시각화가 여러 개가 생긴다.
 - 데이터: 특정 필드로 데이터를 나누기 때문에 각 시각화는 데이터 중 일부를 중복없이 보여준다.
- Repeat: 인코딩을 바꿔가면서 시각화를 여러 개 그린다.
 - 인코딩: 인코딩이 다른 시각화가 여러 개가 생긴다 (x , y 축이 달라짐).
 - 데이터: 각 시각화는 모든 데이터를 보여준다.
- Concat: 시각화 두개를 가로/세로로 나란히 놓는다.
 - 인코딩/데이터: 자유롭게 가능.

Concat



location
New York
Seattle

요약

- Layer: 두 시각화를 겹친다.
 - 인코딩: 적어도 하나의 축은 공유가 되어야!
 - 데이터: 모든 데이터를 보여준다.
- Facet: 데이터를 특정 필드 값으로 나누고 시각화를 여러 개 그린다.
 - 인코딩: 인코딩이 같은 시각화가 여러 개가 생긴다.
 - 데이터: 특정 필드로 데이터를 나누기 때문에 각 시각화는 데이터 중 일부를 중복없이 보여준다.
- Repeat: 인코딩을 바꿔가면서 시각화를 여러 개 그린다.
 - 인코딩: 인코딩이 다른 시각화가 여러 개가 생긴다 (x , y 축이 달라짐).
 - 데이터: 각 시각화는 모든 데이터를 보여준다.
- Concat: 시각화 두개를 가로/세로로 나란히 놓는다.
 - 인코딩/데이터: 자유롭게 가능.

데이터 불러오기

- 오늘은 날씨 데이터를 탐색해보도록 하겠습니다.

- 위치 (뉴욕 또는 시애틀)
- 날짜, 강수량
- 최대온도
- 최소온도
- 바람세기
- 날씨 (이슬비, 안개, 맑음, 눈 등)

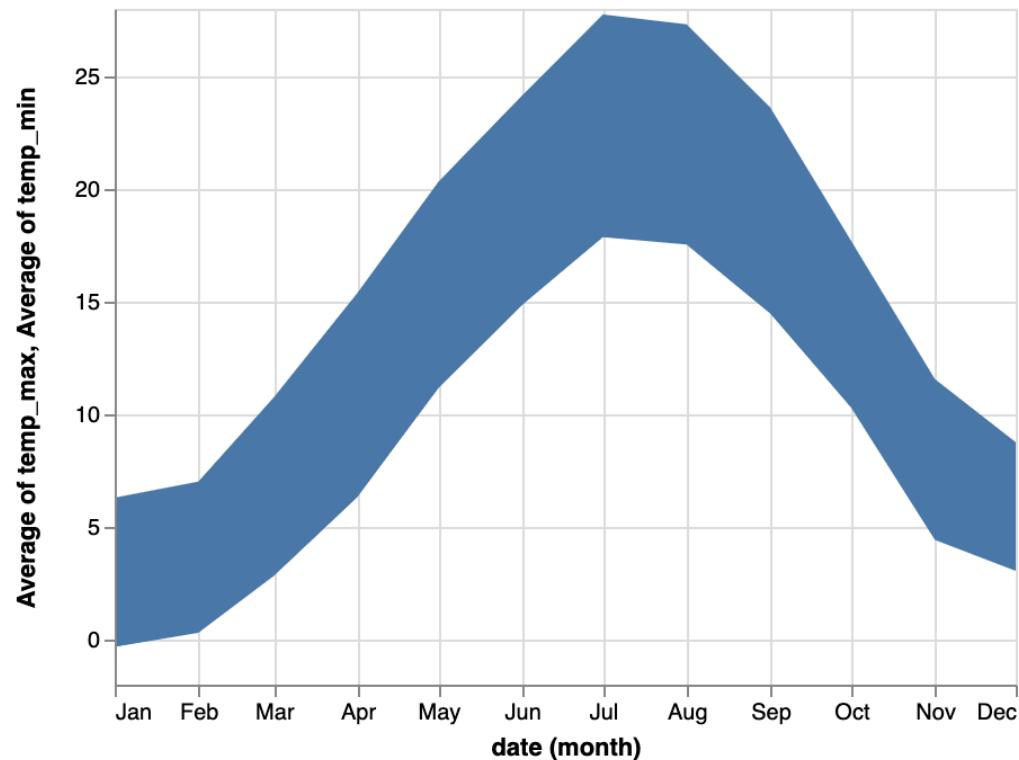
```
1 import altair as alt
2 import pandas as pd
3
4 weather_url = 'https://vega.github.io/vega-datasets/data/weather.csv'
5 weather = pd.read_csv(weather_url)
6 weather.head()
7
```

	location	date	precipitation	temp_max	temp_min	wind	weather
0	Seattle	2012-01-01		0.0	12.8	5.0	4.7
1	Seattle	2012-01-02		10.9	10.6	2.8	4.5
2	Seattle	2012-01-03		0.8	11.7	7.2	2.3
3	Seattle	2012-01-04		20.3	12.2	5.6	4.7
4	Seattle	2012-01-05		1.3	8.9	2.8	6.1

Layer

- 최소 / 최대온도
- 지역 구분 없이 합쳐졌습니다!

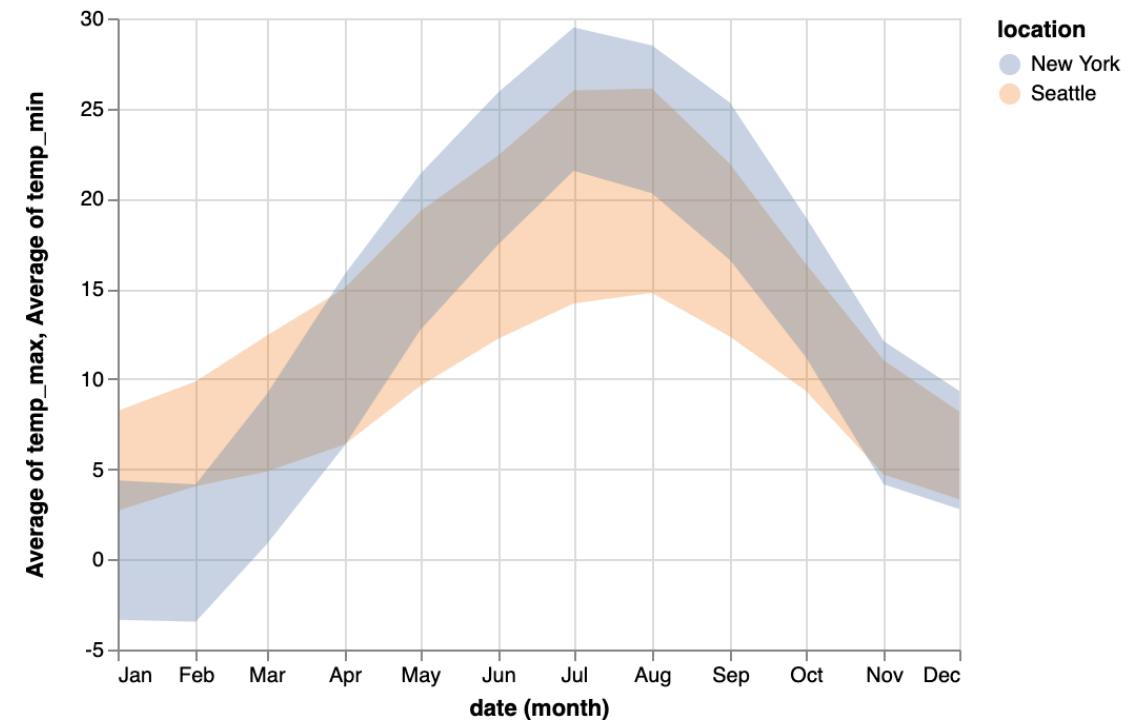
```
alt.Chart(weather).mark_area().encode(  
    alt.X('month(date):T'),  
    alt.Y('average(temp_max):Q'),  
    alt.Y2('average(temp_min):Q')  
)
```



Layer

- 최소/최대온도 지역 구분하여 시각화하기

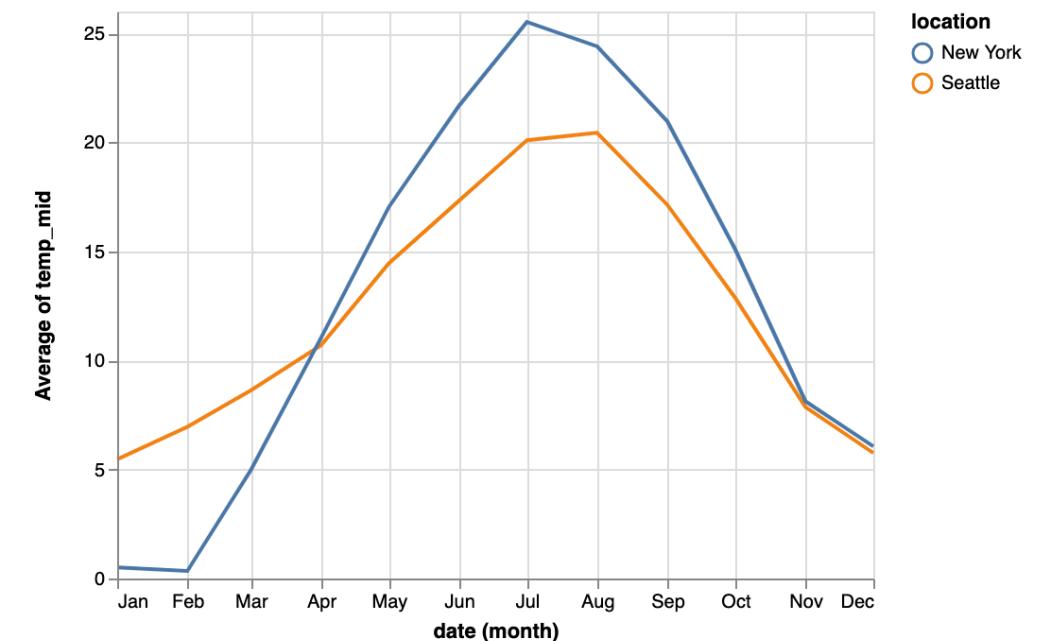
```
alt.Chart(weather).mark_area(opacity=0.3).encode(  
    alt.X('month(date):T'),  
    alt.Y('average(temp_max):Q'),  
    alt.Y2('average(temp_min):Q'),  
    alt.Color('location:N'))
```



Layer

- 중간 온도 시각화하기
- 이미 배운 transform_calculate를 이용하여 새로운 필드를 정의하고 시각화해 봅시다.

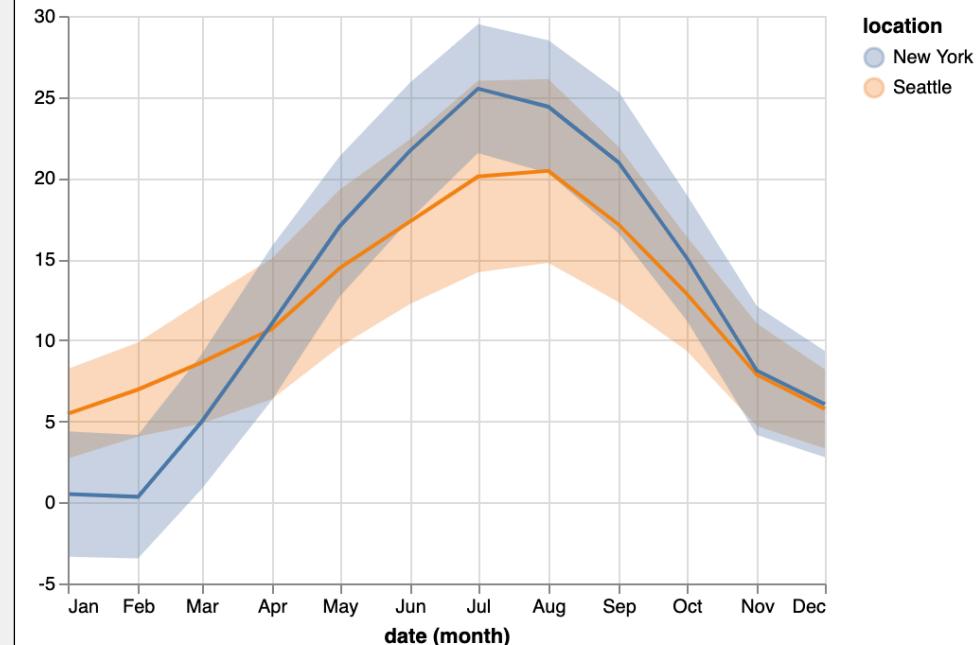
```
alt.Chart(weather).mark_line().transform_calculate(  
    temp_mid='(datum.temp_min + datum.temp_max) / 2'  
).encode(  
    alt.X('month(date):T'),  
    alt.Y('average(temp_mid):Q'),  
    alt.Color('location:N')  
)
```



Layer

- 앞서 만든 시각화를 변수에 담고, + 연산자를 활용하여 겹쳐 봅시다.

```
tempMinMax = alt.Chart(weather).mark_area(opacity=0.3).encode(  
    alt.X('month(date):T'),  
    alt.Y('average(temp_max):Q'),  
    alt.Y2('average(temp_min):Q'),  
    alt.Color('location:N'))  
  
tempMid = alt.Chart(weather).mark_line().transform_calculate(  
    temp_mid='(datum.temp_min + datum.temp_max) / 2').encode(  
    alt.X('month(date):T'),  
    alt.Y('average(temp_mid):Q'),  
    alt.Color('location:N'))  
  
tempMinMax + tempMid
```



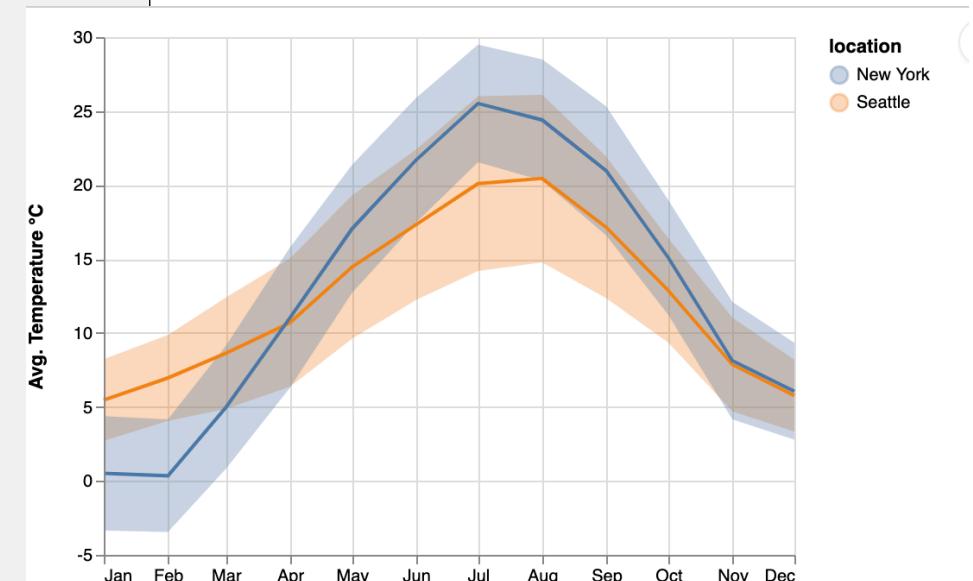
Layer

- X축에 월만 표시하고 제목을 제거합시다.
- Y 축의 이름을 하나로 통일합시다.

```
tempMinMax = alt.Chart(weather).mark_area(opacity=0.3).encode(  
    alt.X('month(date):T', axis=alt.Axis(title=None, format='%b')),  
    alt.Y('average(temp_max):Q', axis=alt.Axis(title='Avg. Temperature °C')),  
    alt.Y2('average(temp_min):Q'),  
    alt.Color('location:N')  
)
```

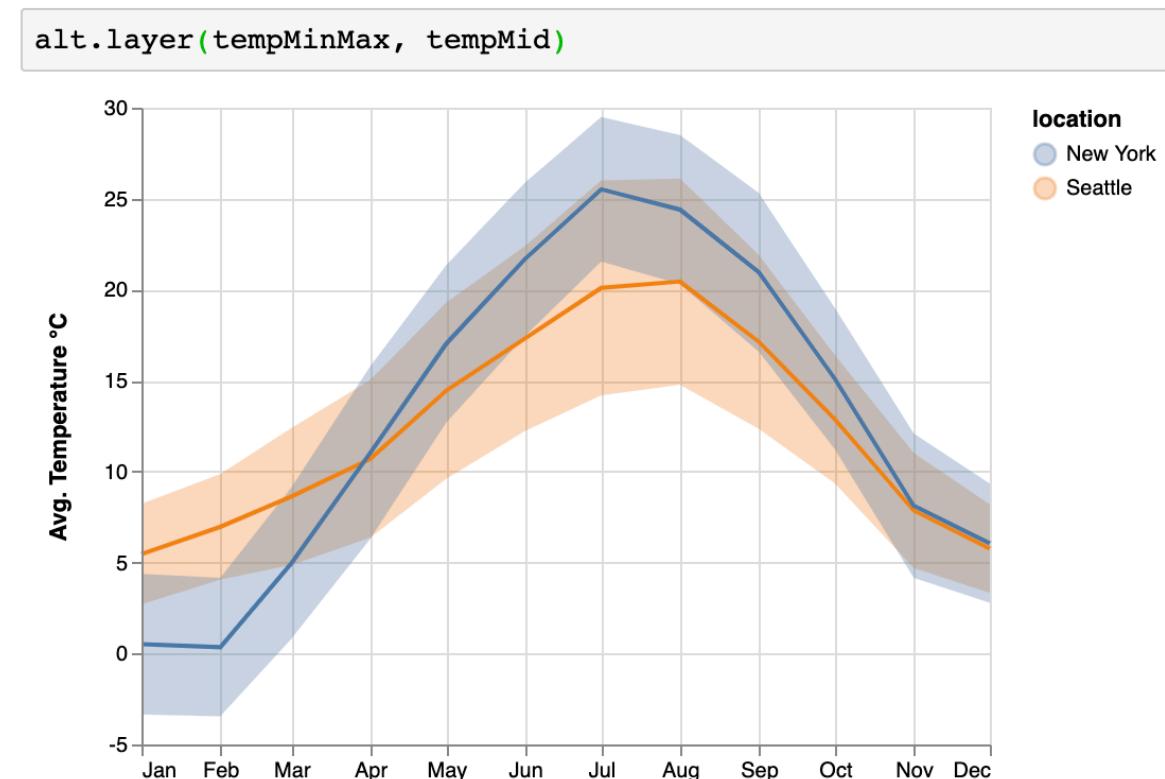
```
tempMid = alt.Chart(weather).mark_line().transform_calculate(  
    temp_mid='(datum.temp_min + datum.temp_max) / 2'  
)  
.encode(  
    alt.X('month(date):T'),  
    alt.Y('average(temp_mid):Q'),  
    alt.Color('location:N')  
)
```

```
tempMinMax + tempMid
```



Layer

- + 연산자는 실제 alt.layer라는 함수를 통해 구현됩니다.
- 연산자 overriding으로 만들어진 바로가기입니다.



Layer

- Altair를 속여봅시다.
- 이전까지는 두 시각화(Area chart, Line chart)의 x축이 같았습니다. 일부러 다르게 하면 어떻게 겹칠까요?

Layer

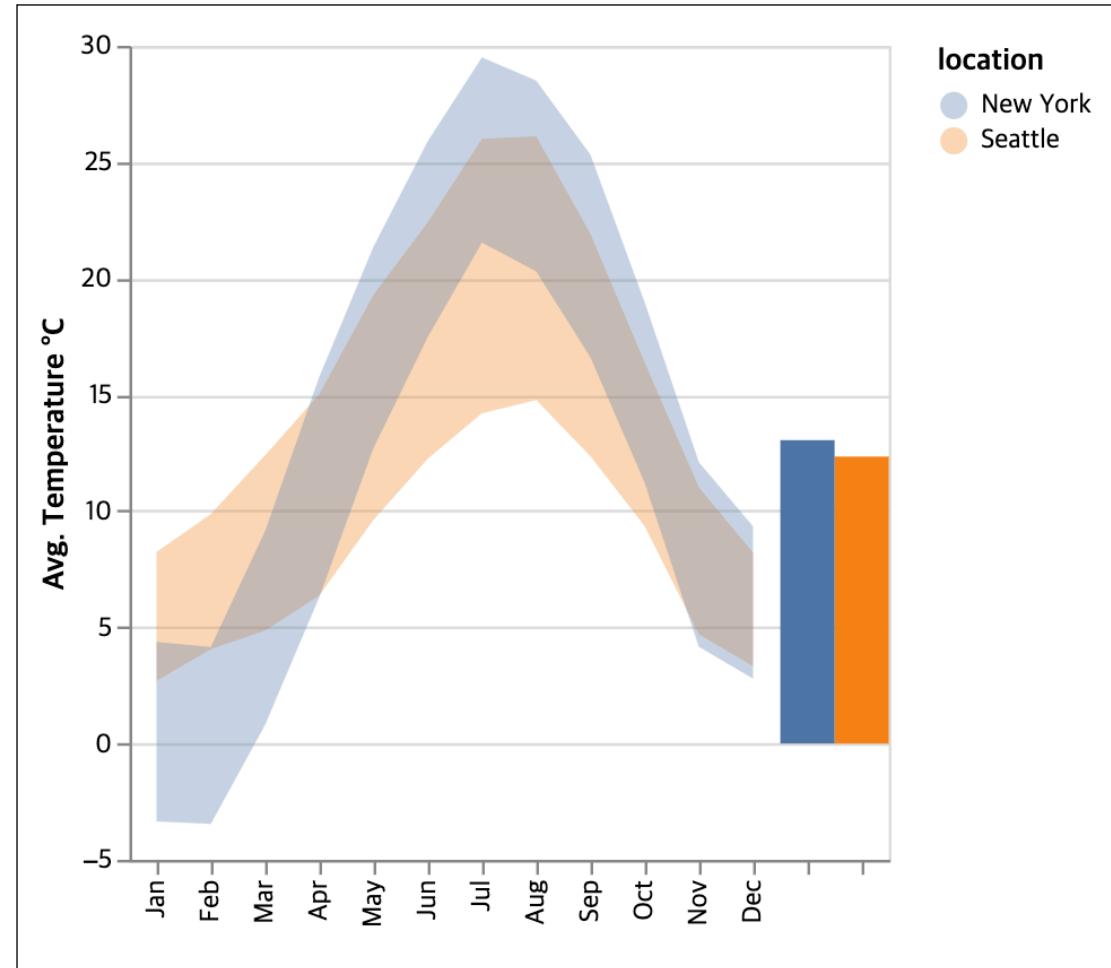
```
tempMinMax = alt.Chart(weather).mark_area(opacity=0.3).encode(
    alt.X('month(date):T', axis=alt.Axis(title=None, format='%b')),
    alt.Y('average(temp_max):Q', axis=alt.Axis(title='Avg. Temperature °C')),
    alt.Y2('average(temp_min):Q'),
    alt.Color('location:N')
)

tempMid = alt.Chart(weather).mark_bar().transform_calculate(
    temp_mid='(datum.temp_min + datum.temp_max) / 2'
).encode(
    alt.X('location:N'),
    alt.Y('average(temp_mid):Q'),
    alt.Color('location:N')
)

tempMinMax + tempMid
```

Layer

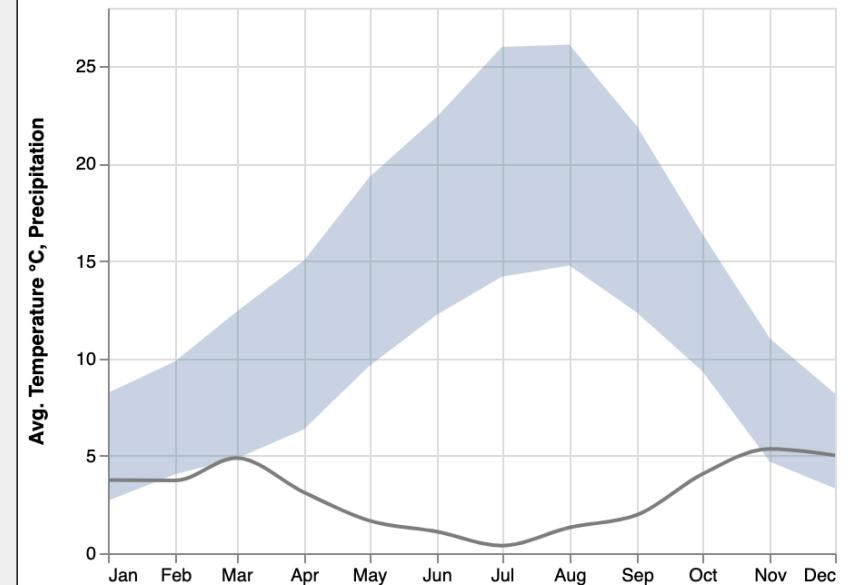
- Altair는 최선을 다한 것 같습니다.



Layer

- 두 개의 시각화를 겹치면 기본적으로 축이 공통으로 사용됩니다.

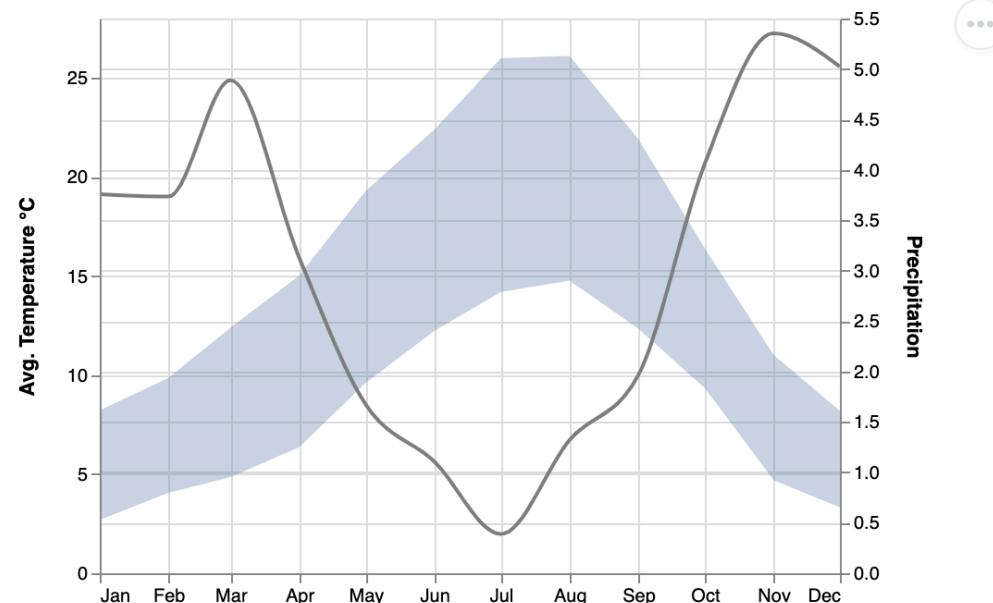
```
tempMinMax = alt.Chart(weather).transform_filter(  
    'datum.location == "Seattle"'  
).mark_area(opacity=0.3).encode(  
    alt.X('month(date):T', axis=alt.Axis(title=None, format='%b')),  
    alt.Y('average(temp_max):Q', axis=alt.Axis(title='Avg. Temperature °C')),  
    alt.Y2('average(temp_min):Q')  
)  
  
precip = alt.Chart(weather).transform_filter(  
    'datum.location == "Seattle"'  
).mark_line(  
    interpolate='monotone',  
    stroke='grey'  
).encode(  
    alt.X('month(date):T'),  
    alt.Y('average(precipitation):Q',  
        axis=alt.Axis(title='Precipitation'))  
)  
  
alt.layer(tempMinMax, precip)
```



Layer

- 그렇지만 단위가 다를 경우 이는 틀린 시각화입니다.
- 축을 따로 설정해봅시다.
- `resolve_scale` 메소드에 `y='independent'`로 설정하여 축을 분리합시다.

```
alt.layer(tempMinMax, precip).resolve_scale(y='independent')
```



Layer

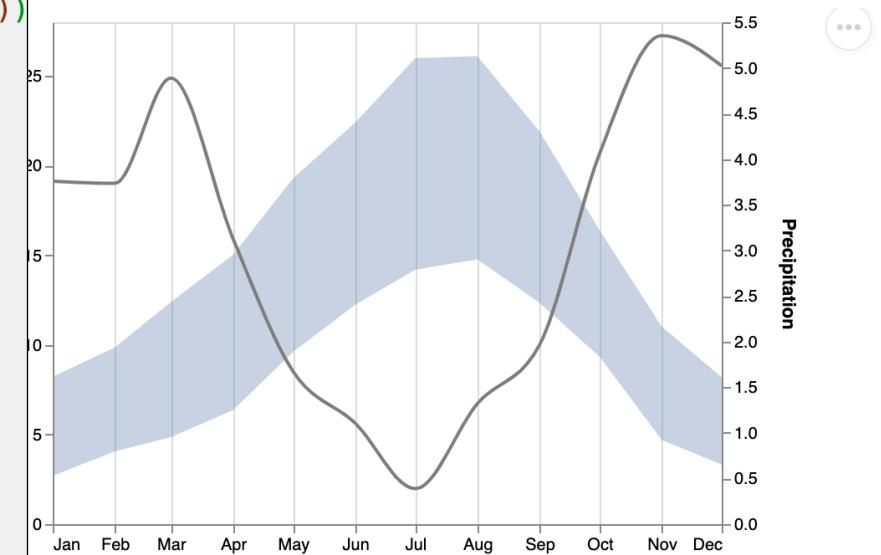
- 보다 간결하게 겹치기

- 축을 따로 설정해봅시다.
- 두 Y축의 두 눈금선을 제거합시다.
- 겹칠 경우 데이터를 입력하는 부분과 필터를 적용하는 부분은 중복입니다. 이를 하나로 합침시다.

```
tempMinMax = alt.Chart().mark_area(opacity=0.3).encode(
    alt.X('month(date):T', axis=alt.Axis(title=None, format='%b')),
    alt.Y('average(temp_max):Q', axis=alt.Axis(title='Avg. Temperature °C', grid=False)),
    alt.Y2('average(temp_min):Q')
)

precip = alt.Chart().mark_line(
    interpolate='monotone',
    stroke='grey'
).encode(
    alt.X('month(date):T'),
    alt.Y('average(precipitation):Q',
        axis=alt.Axis(title='Precipitation', grid=False))
)

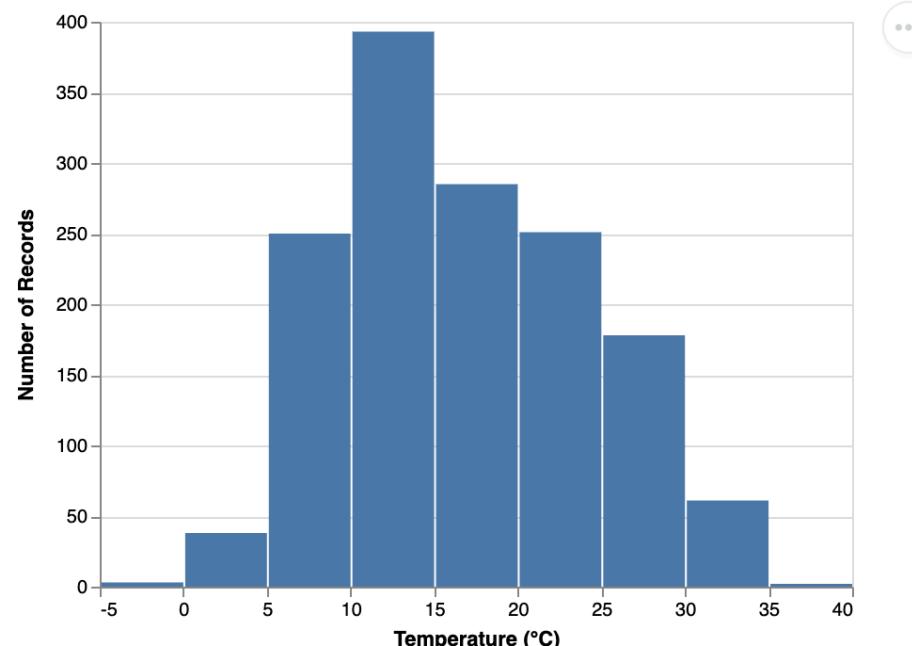
alt.layer(tempMinMax, precip, data=weather).transform_filter(
    'datum.location == "Seattle"'
).resolve_scale(y='independent')
```



Facet

- 같은 시각화를 데이터를 쪼개서 여러 개를 그린다!

```
alt.Chart(weather).mark_bar().transform_filter(  
    'datum.location == "Seattle"'  
).encode(  
    alt.X('temp_max:Q', bin=True, axis=alt.Axis(title='Temperature (°C)'),  
    alt.Y('count():Q')  
)
```



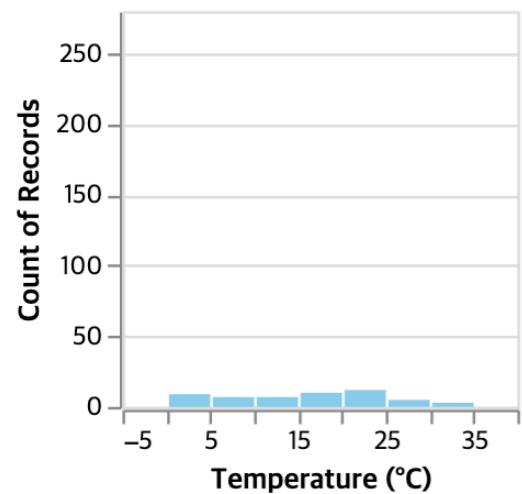
Facet

```
colors = alt.Scale(  
    domain=['drizzle', 'fog', 'rain', 'snow', 'sun'],  
    range=['skyblue', 'lightgrey', 'red', 'lightpink', 'green'])  
  
alt.Chart().mark_bar().encode(  
    alt.X('temp_max:Q', bin=True, axis=alt.Axis(title='Temperature (°C)'),  
    alt.Y('count():Q'),  
    alt.Color('weather:N', scale=colors))  
.properties(  
    width=150,  
    height=150  
.facet(  
    data=weather,  
    column='weather:N'  
.transform_filter(  
    'datum.location == "Seattle"'  
)
```

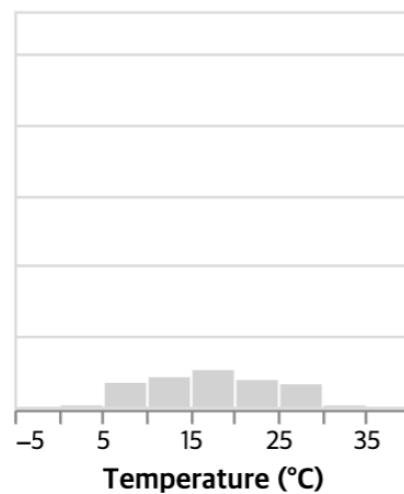
Facet

weather

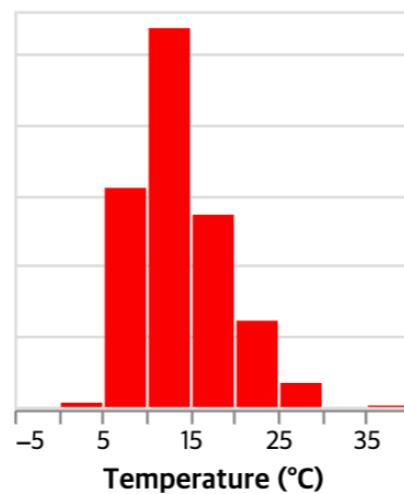
drizzle



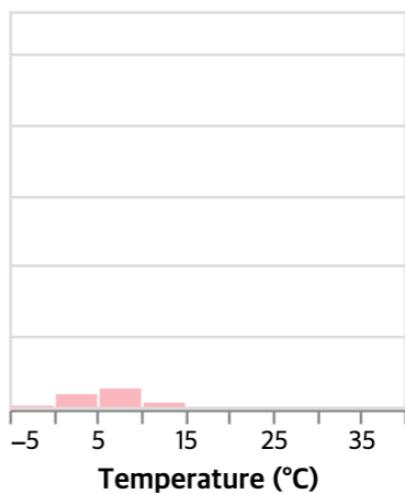
fog



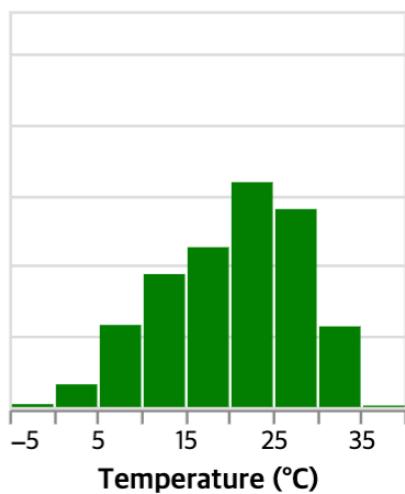
rain



snow



sun



weather

- drizzle
- fog
- rain
- snow
- sun

Facet

- Layer한 차트도 Facet할 수 있다.

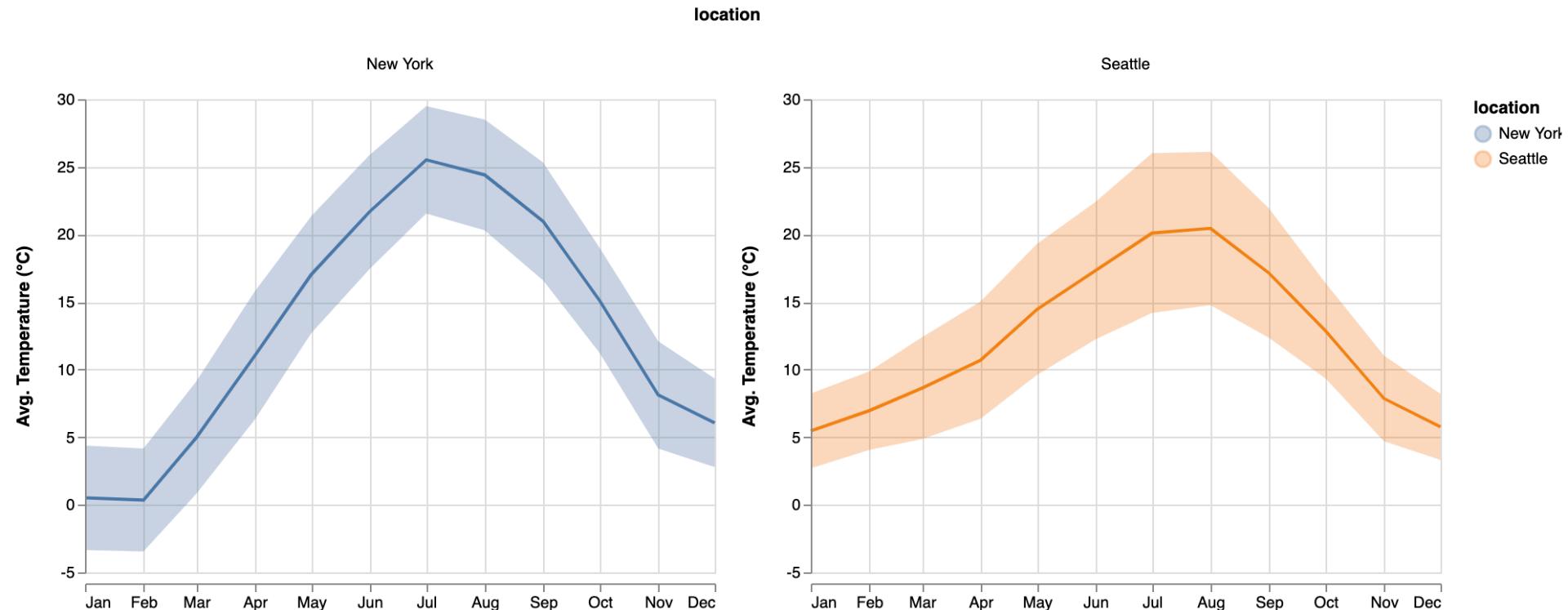
```
tempMinMax = alt.Chart().mark_area(opacity=0.3).encode(
    alt.X('month(date):T', axis=alt.Axis(title=None, format='%b')),
    alt.Y('average(temp_max):Q', axis=alt.Axis(title='Avg. Temperature (°C)')),
    alt.Y2('average(temp_min):Q'),
    alt.Color('location:N')
)

tempMid = alt.Chart().mark_line().transform_calculate(
    temp_mid='(datum.temp_min + datum.temp_max) / 2'
).encode(
    alt.X('month(date):T'),
    alt.Y('average(temp_mid):Q'),
    alt.Color('location:N')
)

alt.layer(tempMinMax, tempMid).facet(
    data=weather,
    column='location:N'
).resolve_axis(y='independent')
```

Facet

- Layer한 시각화를 Facet할 수 있습니다.
 - 이 경우 layer로 인해 겹쳐진 시각화가 facet에 의해 여러 개 그려집니다.



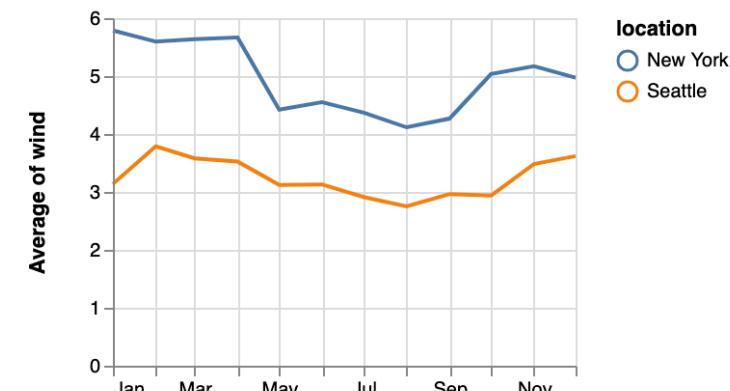
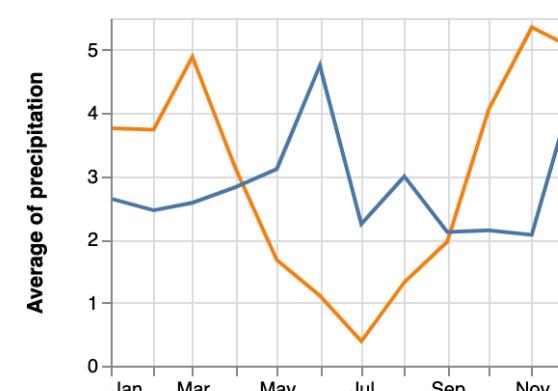
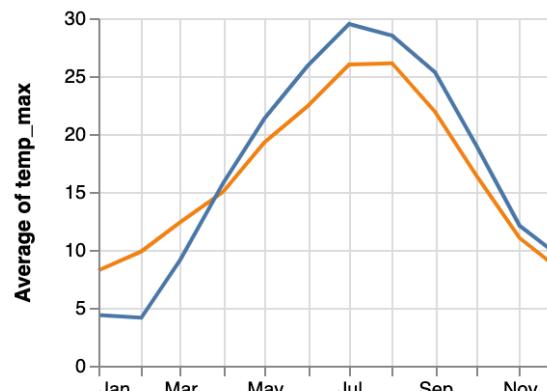
Concat

- | (alt.hconcat) 연산자를 이용하여 세로로 시각화를 병치하기
 - 이 때, 이는 단순히 시각화를 붙이는 것이므로 인코딩이나 데이터에 조건이 없다.

```
base = alt.Chart(weather).mark_line().encode(
    alt.X('month(date):T', axis=alt.Axis(title=None)),
    color='location:N'
).properties(
    width=240,
    height=180
)

temp = base.encode(alt.Y('average(temp_max):Q'))
precip = base.encode(alt.Y('average(precipitation):Q'))
wind = base.encode(alt.Y('average(wind):Q'))

temp | precip | wind
```



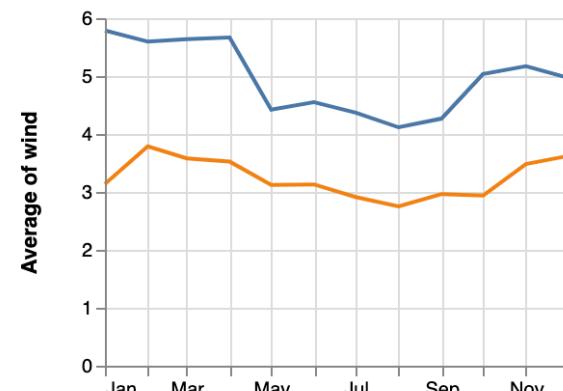
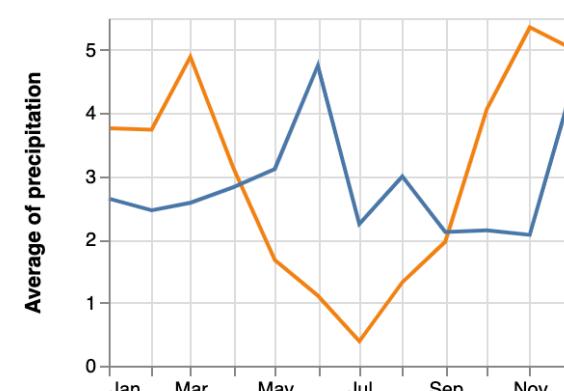
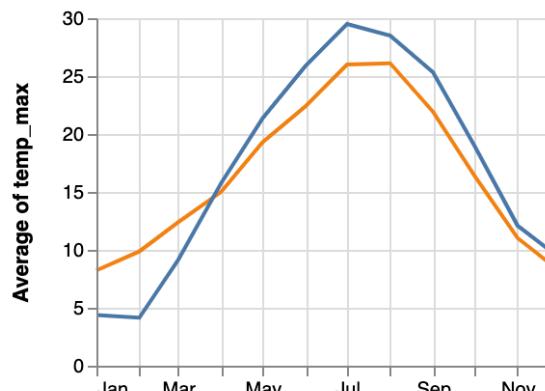
Concat

- 같은 시각화를 그릴 때 모든 코드를 반복하지 않고, 공통된 부분은 base에 인코딩을 기술하고, 달라지는 부분만 바꾼 것에 주목합시다.

```
base = alt.Chart(weather).mark_line().encode(
    alt.X('month(date):T', axis=alt.Axis(title=None)),
    color='location:N'
).properties(
    width=240,
    height=180
)

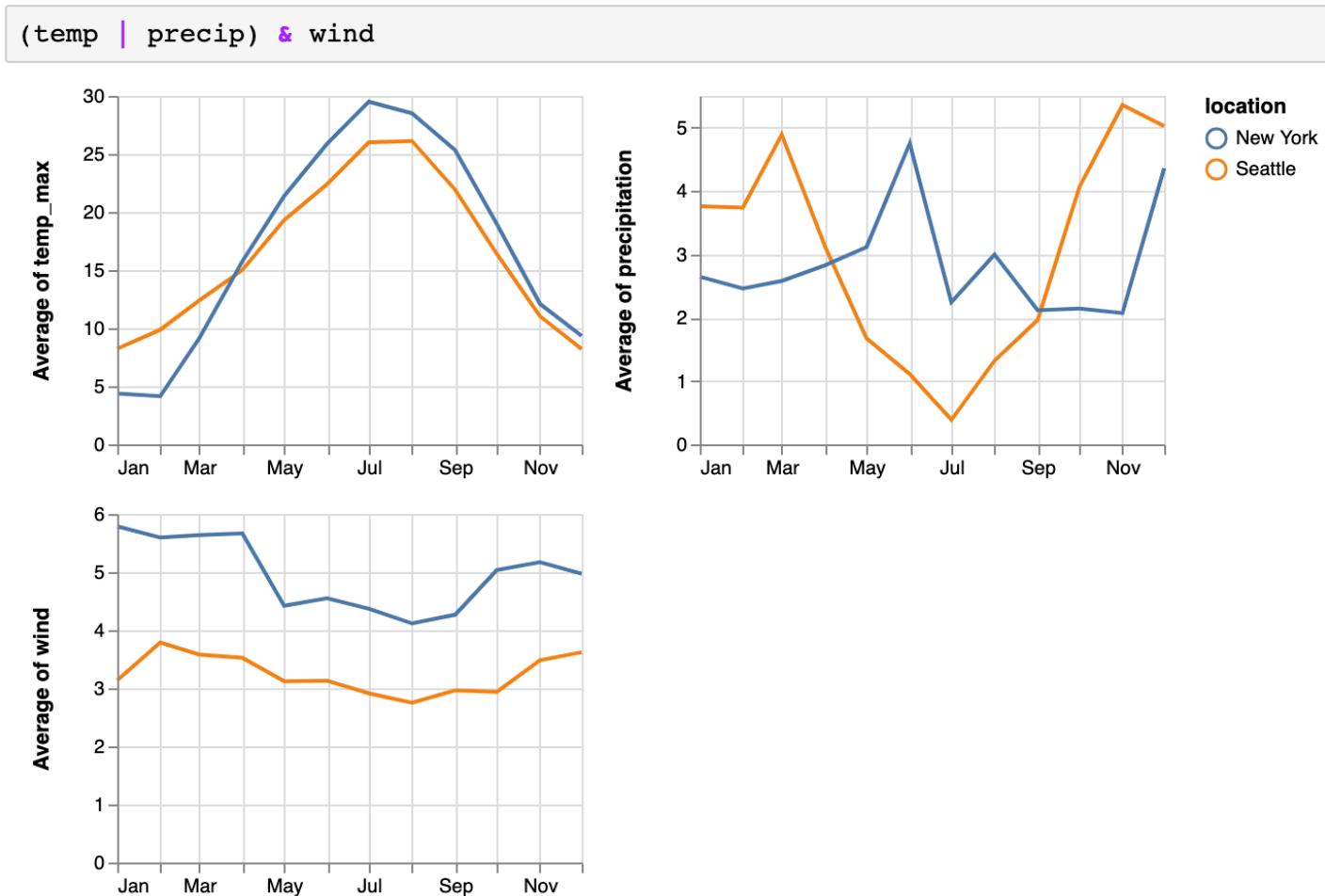
temp = base.encode(alt.Y('average(temp_max):Q'))
precip = base.encode(alt.Y('average(precipitation):Q'))
wind = base.encode(alt.Y('average(wind):Q'))

temp | precip | wind
```



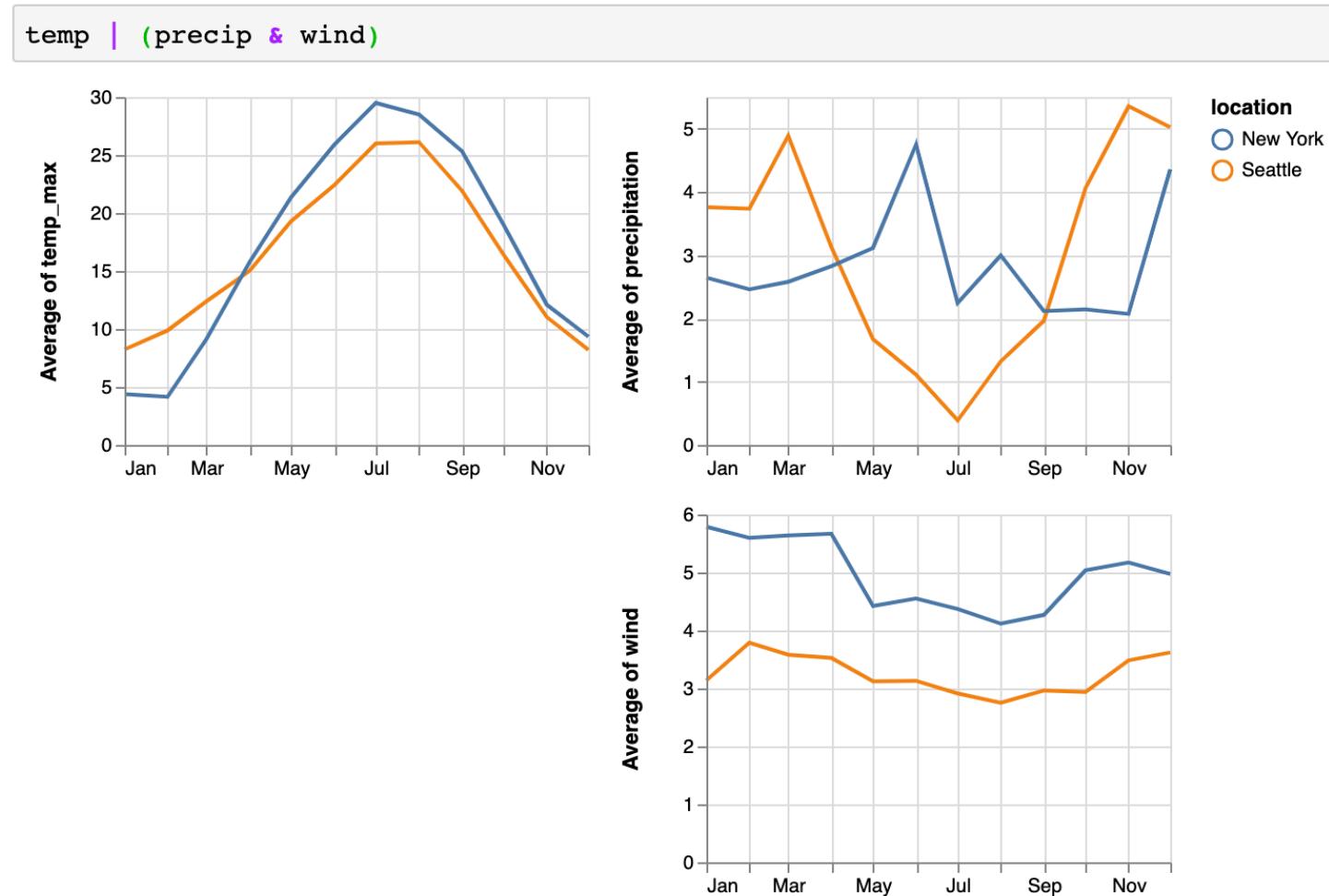
Concat

- & (alt.vconcat) 을 이용하여 세로로 병치할 수도 있습니다.



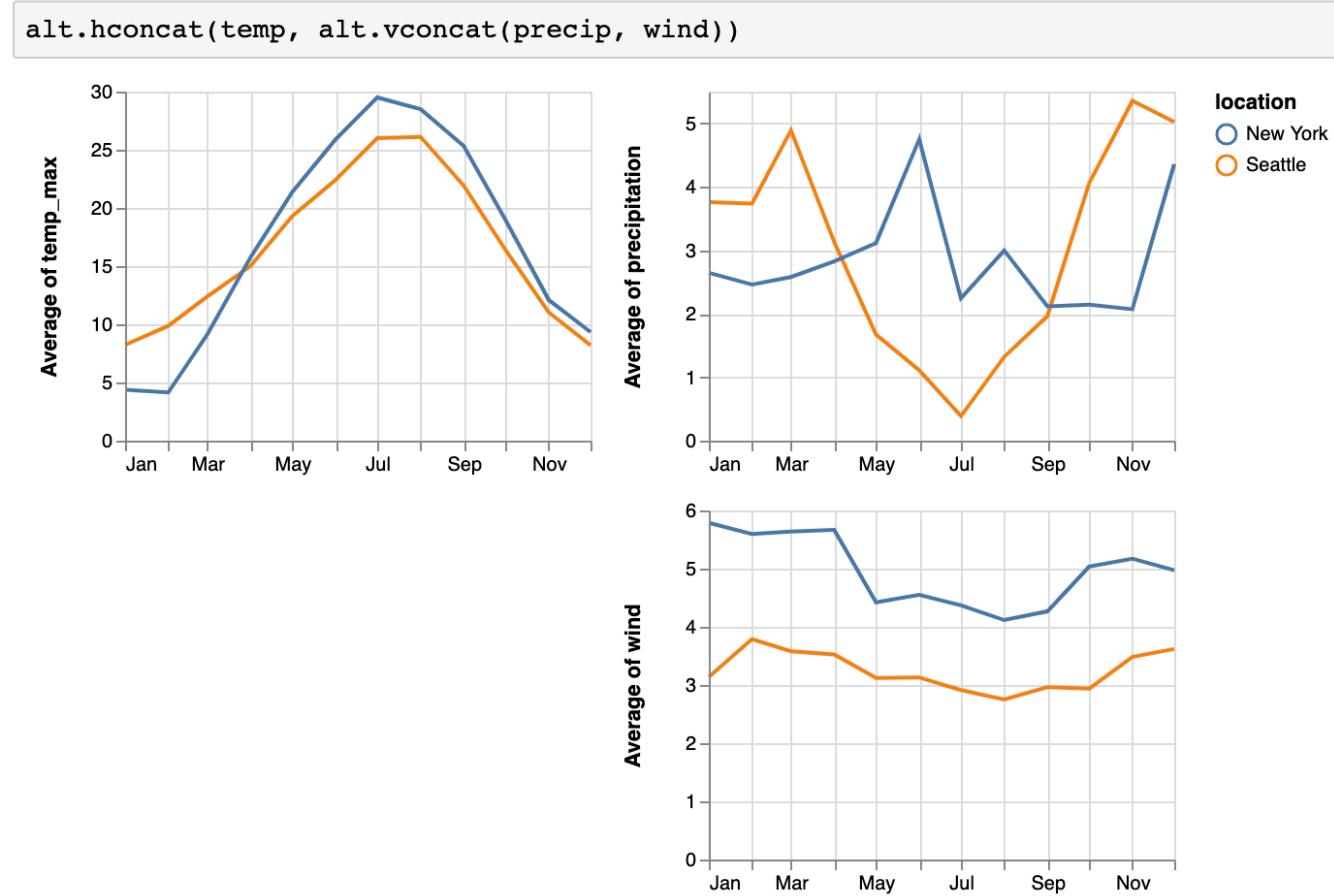
Concat

- 괄호를 통해 연산 순서가 병치 순서에 영향을 미칩니다.



Concat

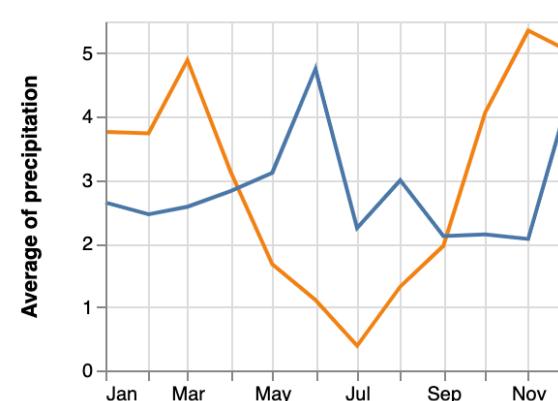
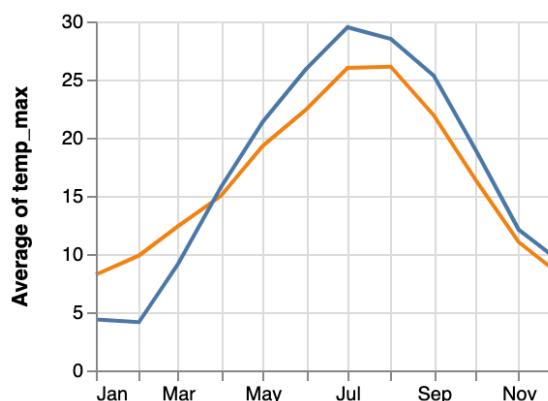
- + 처럼, 원래 & 및 | 연산자를 alt.vconcat, alt.hconcat 함수에 overriding한 것입니다.



Repeat

- Facet은 인코딩을 유지한 채 데이터를 쪼개서 여러 개의 시각화를 병치합니다.
- alt.repeat을 통해 데이터를 유지한 채 인코딩을 바꿀 수도 있습니다.

```
alt.Chart(weather).mark_line().encode(  
    alt.X('month(date):T', axis=alt.Axis(title=None)),  
    alt.Y(alt.repeat('column'), aggregate='average', type='quantitative'),  
    color='location:N'  
)  
.properties(  
    width=240,  
    height=180  
)  
.repeat(  
    column=['temp_max', 'precipitation', 'wind'])  
)
```



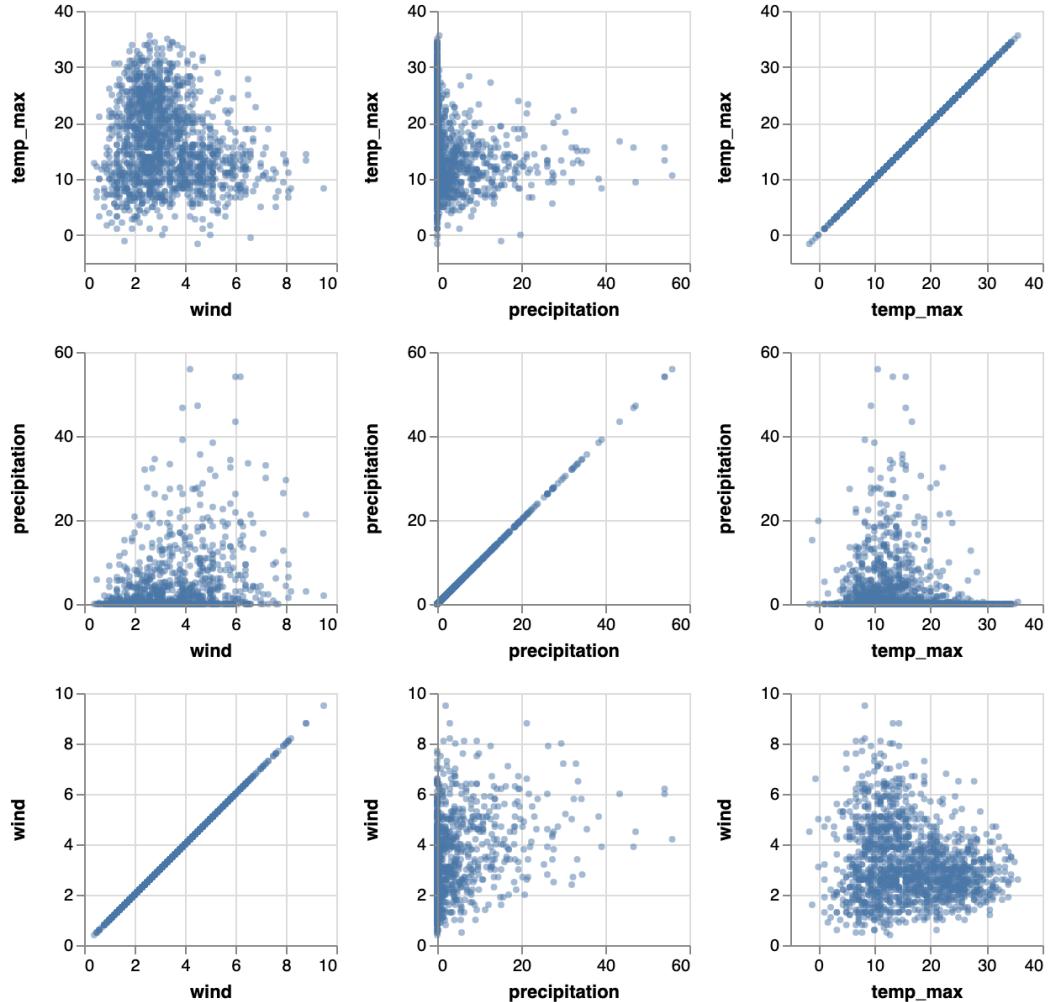
Repeat

- 아까는 base를 이용하여 세 개의 시각화를 만들고 이를 |로 병치했습니다.
- 이번에는 인코딩에 빈 칸을 남겨두고 여기에 들어갈 필드들의 리스트를 제공하여 간단하게 했습니다.
- 이는 특히, 필드의 수가 많거나 필드 조합을 인코딩 할 때 유용합니다.

```
alt.Chart().mark_point(filled=True, size=15, opacity=0.5).encode(  
    alt.X(alt.repeat("column"), type='quantitative'),  
    alt.Y(alt.repeat("row"), type='quantitative')  
).properties(  
    width=150,  
    height=150  
).repeat(  
    data=weather,  
    row=[ 'temp_max', 'precipitation', 'wind' ],  
    column=[ 'wind', 'precipitation', 'temp_max' ]  
).transform_filter(  
    'datum.location == "Seattle"  
)
```

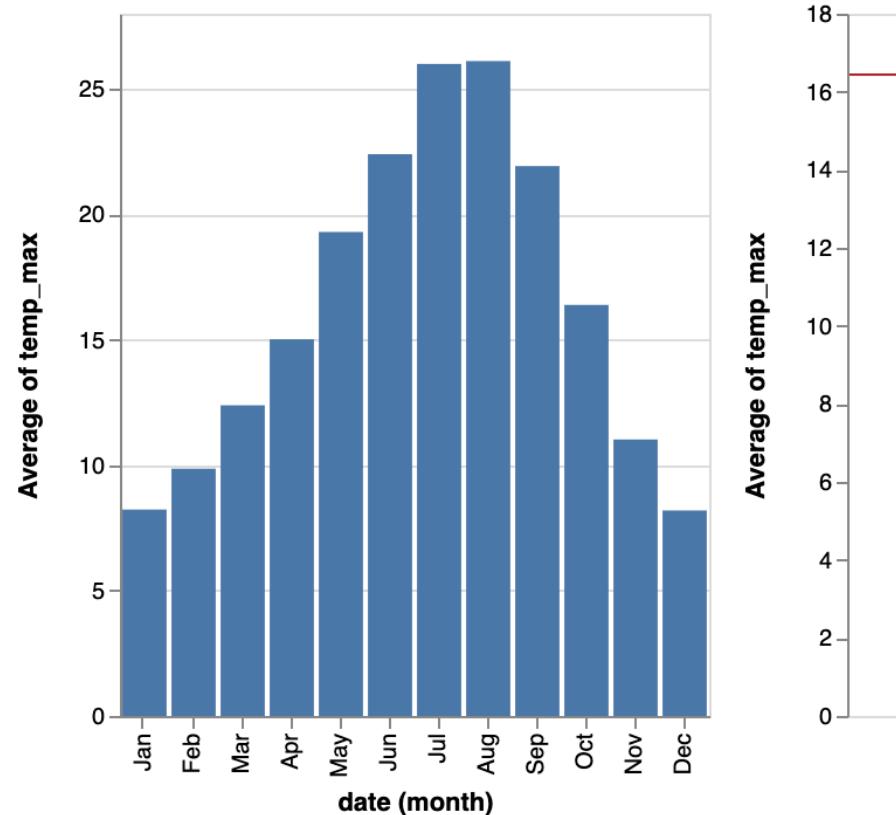
Repeat

- SPLOM (Scatterplot Matrix)



응용) Rule Mark와 Concatenation

```
basic1 = alt.Chart(weather).transform_filter(  
    'datum.location == "Seattle"'  
).mark_bar().encode(  
    alt.X('month(date):O'),  
    alt.Y('average(temp_max):Q')  
)  
  
basic2 = alt.Chart(weather).transform_filter(  
    'datum.location == "Seattle"'  
).mark_rule(stroke='firebrick').encode(  
    alt.Y('average(temp_max):Q')  
)  
  
basic1 | basic2
```



응용) Rule Mark와 Concatenation

```
alt.layer(
    alt.Chart().mark_bar().encode(
        alt.X('month(date):0', axis=alt.Axis(title='Month')),
        alt.Y(alt.repeat('column'), aggregate='average', type='quantitative')
    ),
    alt.Chart().mark_rule(stroke='firebrick').encode(
        alt.Y(alt.repeat('column'), aggregate='average', type='quantitative')
    )
).properties(
    width=200,
    height=150
).repeat(
    data=weather,
    column=[ 'temp_max', 'precipitation', 'wind' ]
).transform_filter(
    'datum.location == "Seattle"'
)
```

