


CS-GY 6313 / CUSP-GX 6006: Data Visualization - Spring '24

Homework #4: Interactive Visualizations

In this homework assignment, you will be focusing on adding interactive components to 2D plots built with `Altair` (<https://altair-viz.github.io/gallery/index.html>). This should give you an idea on how to generate interactivity in your plots.

Interactive Weather Plots (15 points)

The code below is helper code to get you started. Let's import the necessary packages for Altair and the weather data that we'll be using in this assignment.

```
In [1]:  """ =====  
=== DO NOT MODIFY THIS CODE ===  
===== """  
  
# Installling the necessary packages  
!pip install "altair[all]"  
import altair as alt  
from vega_datasets import data
```

Requirement already satisfied: altair[all] in d:\annaconda\lib\site-packages (5.3.0)

Requirement already satisfied: jinja2 in d:\annaconda\lib\site-packages (from altair[all]) (3.1.2)

Requirement already satisfied: jsonschema>=3.0 in d:\annaconda\lib\site-packages (from altair[all]) (4.17.3)

Requirement already satisfied: numpy in d:\annaconda\lib\site-packages (from altair[all]) (1.24.3)

Requirement already satisfied: packaging in d:\annaconda\lib\site-packages (from altair[all]) (23.1)

Requirement already satisfied: pandas>=0.25 in d:\annaconda\lib\site-packages (from altair[all]) (2.0.3)

Requirement already satisfied: toolz in d:\annaconda\lib\site-packages (from altair[all]) (0.12.0)

Requirement already satisfied: altair-tiles>=0.3.0 in d:\annaconda\lib\site-packages (from altair[all]) (0.3.0)

Requirement already satisfied: anywidget>=0.9.0 in d:\annaconda\lib\site-packages (from altair[all]) (0.9.10)

Requirement already satisfied: pyarrow>=11 in d:\annaconda\lib\site-packages (from altair[all]) (11.0.0)

Requirement already satisfied: vega-datasets>=0.9.0 in d:\annaconda\lib\site-packages (from altair[all]) (0.9.0)

Requirement already satisfied: vegafusion[embed]>=1.6.6 in d:\annaconda\lib\site-packages (from altair[all]) (1.6.9)

Requirement already satisfied: vl-convert-python>=1.3.0 in d:\annaconda\lib\site-packages (from altair[all]) (1.4.0)

Requirement already satisfied: mercantile in d:\annaconda\lib\site-packages (from altair-tiles>=0.3.0->altair[all]) (1.2.1)

Requirement already satisfied: xyzservices in d:\annaconda\lib\site-packages (from altair-tiles>=0.3.0->altair[all]) (2022.9.0)

Requirement already satisfied: ipywidgets>=7.6.0 in d:\annaconda\lib\site-packages (from anywidget>=0.9.0->altair[all]) (8.0.4)

Requirement already satisfied: psygnal>=0.8.1 in d:\annaconda\lib\site-packages (from anywidget>=0.9.0->altair[all]) (0.11.1)

Requirement already satisfied: typing-extensions>=4.2.0 in d:\annaconda\lib\site-packages (from anywidget>=0.9.0->altair[all]) (4.11.0)

Requirement already satisfied: attrs>=17.4.0 in d:\annaconda\lib\site-packages (from jsonschema>=3.0->altair[all]) (22.1.0)

Requirement already satisfied: pyparsing!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in d:\annaconda\lib\site-packages (from jsonschema>=3.0->altair[all]) (0.18.0)

Requirement already satisfied: python-dateutil>=2.8.2 in d:\annaconda\lib\site-packages (from pandas>=0.25->altair[all]) (2.8.2)

Requirement already satisfied: pytz>=2020.1 in d:\annaconda\lib\site-packages (from pandas>=0.25->altair[all]) (2023.3.post1)

Requirement already satisfied: tzdata>=2022.1 in d:\annaconda\lib\site-packages (from pandas>=0.25->altair[all]) (2023.3)

Requirement already satisfied: psutil in d:\annaconda\lib\site-packages (from vegafusion[embed]>=1.6.6->altair[all]) (5.9.0)

Requirement already satisfied: protobuf in d:\annaconda\lib\site-packages (from vegafusion[embed]>=1.6.6->altair[all]) (4.25.3)

Requirement already satisfied: vegafusion-python-embed==1.6.9 in d:\annaconda\lib\site-packages (from vegafusion[embed]>=1.6.6->altair[all]) (1.6.9)

Requirement already satisfied: MarkupSafe>=2.0 in d:\annaconda\lib\site-packages (from jinja2->altair[all]) (2.1.1)

Requirement already satisfied: ipykernel>=4.5.1 in d:\annaconda\lib\site-packages (from ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (6.25.0)

Requirement already satisfied: ipython>=6.1.0 in d:\annaconda\lib\site-packages (from ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (8.15.0)

Requirement already satisfied: traitlets>=4.3.1 in d:\annaconda\lib\site-packages (from ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (5.7.1)

Requirement already satisfied: widgetsnbextension~=4.0 in d:\anaconda\lib\site-packages (from ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (4.0.5)

Requirement already satisfied: jupyterlab-widgets~=3.0 in d:\anaconda\lib\site-packages (from ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (3.0.5)

Requirement already satisfied: six>=1.5 in d:\anaconda\lib\site-packages (from python-dateutil>=2.8.2->pandas>=0.25->altair[all]) (1.16.0)

Requirement already satisfied: click>=3.0 in d:\anaconda\lib\site-packages (from mercantile->altair-tiles>=0.3.0->altair[all]) (8.0.4)

Requirement already satisfied: colorama in d:\anaconda\lib\site-packages (from click>=3.0->mercantile->altair-tiles>=0.3.0->altair[all]) (0.4.6)

Requirement already satisfied: comm>=0.1.1 in d:\anaconda\lib\site-packages (from ipykernel>=4.5.1->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (0.1.2)

Requirement already satisfied: debugpy>=1.6.5 in d:\anaconda\lib\site-packages (from ipykernel>=4.5.1->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (1.6.7)

Requirement already satisfied: jupyter-client>=6.1.12 in d:\anaconda\lib\site-packages (from ipykernel>=4.5.1->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (7.4.9)

Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in d:\anaconda\lib\site-packages (from ipykernel>=4.5.1->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (5.3.0)

Requirement already satisfied: matplotlib-inline>=0.1 in d:\anaconda\lib\site-packages (from ipykernel>=4.5.1->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (0.1.6)

Requirement already satisfied: nest-asyncio in d:\anaconda\lib\site-packages (from ipykernel>=4.5.1->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (1.5.6)

Requirement already satisfied: pyzmq>=20 in d:\anaconda\lib\site-packages (from ipykernel>=4.5.1->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (23.2.0)

Requirement already satisfied: tornado>=6.1 in d:\anaconda\lib\site-packages (from ipykernel>=4.5.1->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (6.3.2)

Requirement already satisfied: backcall in d:\anaconda\lib\site-packages (from ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (0.2.0)

Requirement already satisfied: decorator in d:\anaconda\lib\site-packages (from ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (5.1.1)

Requirement already satisfied: jedi>=0.16 in d:\anaconda\lib\site-packages (from ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (0.18.1)

Requirement already satisfied: pickleshare in d:\anaconda\lib\site-packages (from ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (0.7.5)

Requirement already satisfied: prompt-toolkit!=3.0.37,<3.1.0,>=3.0.30 in d:\anaconda\lib\site-packages (from ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (3.0.36)

Requirement already satisfied: pygments>=2.4.0 in d:\anaconda\lib\site-packages (from ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (2.15.1)

Requirement already satisfied: stack-data in d:\anaconda\lib\site-packages (from ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (0.2.0)

Requirement already satisfied: parso<0.9.0,>=0.8.0 in d:\anaconda\lib\site-packages (from jedi>=0.16->ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (0.8.3)

Requirement already satisfied: entrypoints in d:\anaconda\lib\site-packages (from jupyter-client>=6.1.12->ipykernel>=4.5.1->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[all]) (0.4)

Requirement already satisfied: platformdirs>=2.5 in d:\anaconda\lib\site-packages (from jupyter-core!=5.0.*,>=4.12->ipykernel>=4.5.1->ipywidgets>=7.6.0->

```

anywidget>=0.9.0->altair[all]) (3.10.0)
Requirement already satisfied: pywin32>=300 in d:\annaconda\lib\site-packages
(from jupyter-core!=5.0.*,>=4.12->ipykernel>=4.5.1->ipywidgets>=7.6.0->anywid
get>=0.9.0->altair[all]) (305.1)
Requirement already satisfied: wcwidth in d:\annaconda\lib\site-packages (fro
m prompt-toolkit!=3.0.37,<3.1.0,>=3.0.30->ipython>=6.1.0->ipywidgets>=7.6.0->
anywidget>=0.9.0->altair[all]) (0.2.5)
Requirement already satisfied: executing in d:\annaconda\lib\site-packages (f
rom stack-data->ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[a
ll]) (0.8.3)
Requirement already satisfied: asttokens in d:\annaconda\lib\site-packages (f
rom stack-data->ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[a
ll]) (2.0.5)
Requirement already satisfied: pure-eval in d:\annaconda\lib\site-packages (f
rom stack-data->ipython>=6.1.0->ipywidgets>=7.6.0->anywidget>=0.9.0->altair[a
ll]) (0.2.2)

```

The next code block handles the data import for the Seattle weather data between 2012 to 2015. Meanwhile, we define the color scale that maps particular weather types to colors. We will use this color mapping when producing visualizations of each weather type.

```

In [2]: ▶ """ =====
=== DO NOT MODIFY THIS CODE ===
===== """

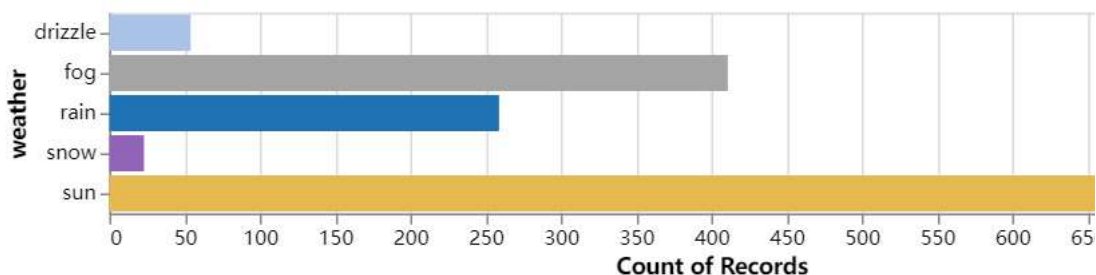
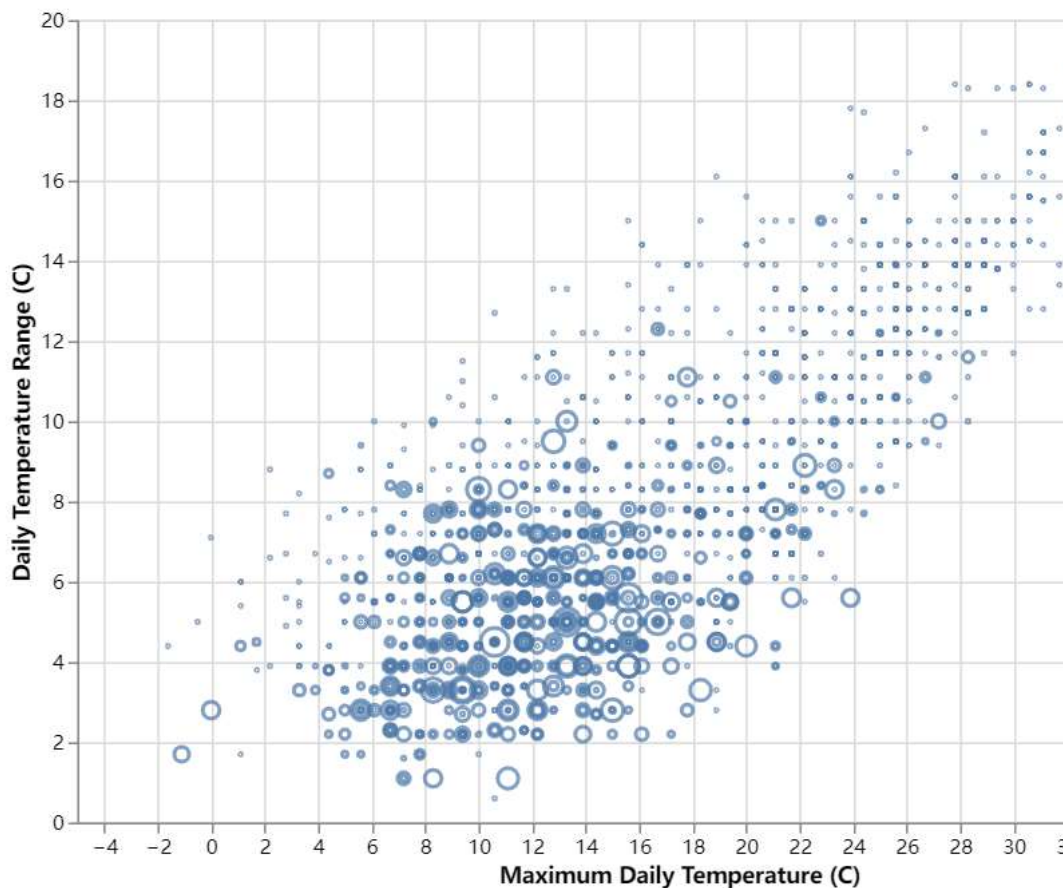
# Import the seattle weather data
source = data.seattle_weather()

# Initialize some parameters for the data
scale = alt.Scale(domain=['sun', 'fog', 'drizzle', 'rain', 'snow'],
                    range=['#e7ba52', '#a7a7a7', '#aec7e8', '#1f77b4', '#9467bd']
color = alt.Color('weather:N', scale=scale)

```

```
In [3]: """ =====  
=== You can modify the code starting from here ===  
===== """  
  
points = alt.Chart().mark_point().encode(  
    alt.X('temp_max:Q', title='Maximum Daily Temperature (C)'),  
    alt.Y('temp_range:Q', title='Daily Temperature Range (C)'),  
    size=alt.Size('precipitation:Q', scale=alt.Scale(range=[1, 200]))  
)  
.transform_calculate(  
    "temp_range", "datum.temp_max - datum.temp_min"  
)  
.properties(  
    width=600,  
    height=400  
)  
  
bars = alt.Chart().mark_bar().encode(  
    x='count()',  
    y='weather:N',  
    color=alt.Color('weather:N', scale=scale),  
)  
.transform_calculate(  
    "temp_range", "datum.temp_max - datum.temp_min"  
)  
.properties(  
    width=600  
)  
  
alt.vconcat(points, bars, data=source)
```

Out[3]:



Part 1. Changing Plots (5 points)


Change the bottom chart such that, instead of showing the number of days that correspond to each weather type, the bottom chart is a horizontal bar chart that shows the amount of precipitation for each day. The bar charts must be a stacked bar chart where the x-axis represents the sum of the amount of precipitation that day, the y-axis represents the date (in month-date format), and the stacking must indicate the percentage each weather type contributed to the total sum. The stacking must also be color-coded to reflect the colors in the scale provided.

In [4]: `import pandas as pd`

In [5]:  source.head()

Out[5]:

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

In [6]: 

```
# Convert the 'date' column to datetime format
source['date'] = pd.to_datetime(source['date'])

# Extract day of year from the date
source['day_of_year'] = source['date'].dt.dayofyear

source.head()
```

Out[6]:

	date	precipitation	temp_max	temp_min	wind	weather	day_of_year
0	2012-01-01	0.0	12.8	5.0	4.7	drizzle	1
1	2012-01-02	10.9	10.6	2.8	4.5	rain	2
2	2012-01-03	0.8	11.7	7.2	2.3	rain	3
3	2012-01-04	20.3	12.2	5.6	4.7	rain	4
4	2012-01-05	1.3	8.9	2.8	6.1	rain	5


```

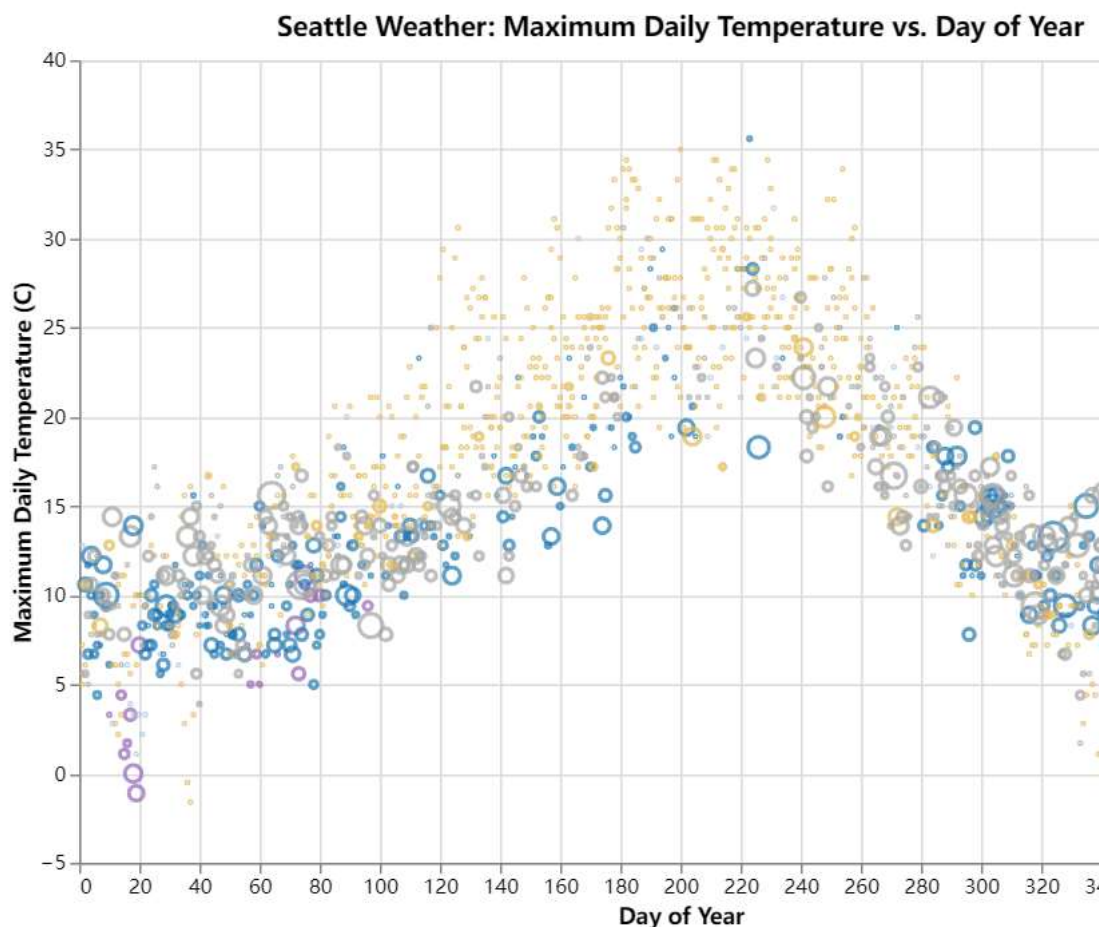
In [7]: ▶ # Create the scatter plot
points = alt.Chart(source).mark_point().encode(
    alt.X('day_of_year:Q', title='Day of Year'),
    alt.Y('temp_max:Q', title='Maximum Daily Temperature (C)'),
    size=alt.Size('precipitation:Q', scale=alt.Scale(range=[1, 200])),
    color=alt.Color('weather:N', scale=scale, title='Weather'),
    tooltip=['date:T', 'temp_max:Q', 'precipitation:Q']
).properties(
    title='Seattle Weather: Maximum Daily Temperature vs. Day of Year',
    width=600,
    height=400
)

# Create the stacked horizontal bar chart
bars = alt.Chart(source).mark_bar().encode(
    alt.X('sum(precipitation):Q', title='Sum of Precipitation'),
    alt.Y('monthdate(date):O', title='Date'),
    color=alt.Color('weather:N', scale=scale, title='Weather'),
    order=alt.Order('weather', sort='ascending'),
    tooltip=['date:T', 'precipitation:Q', 'weather:N']
).properties(
    title='Total Precipitation by Date',
    width=600,
    height=400
)

# Show the scatter plot and the stacked bar chart side by side
alt.hconcat(points, bars)

```

Out[7]:



In []: ▶

In []: ▶

Part 2: Weather Type Selection (5 points)

Modify the top graph so that if you click the weather name in the legend, all data points with different weather types are grayed out in the top chart. If I click anywhere else, the legend is reset and all colors are returned to the top chart.

You will likely have to make use of some Altair functions such as `alt.selection_point(...)` and `alt.condition(...)` to add the required functionality

```
In [20]: # Create a selection that chooses the nearest point & selects based on weather
nearest = alt.selection_single(
    on='mouseover', nearest=True, empty='none',
    fields=['day_of_year'], clear='click'
)

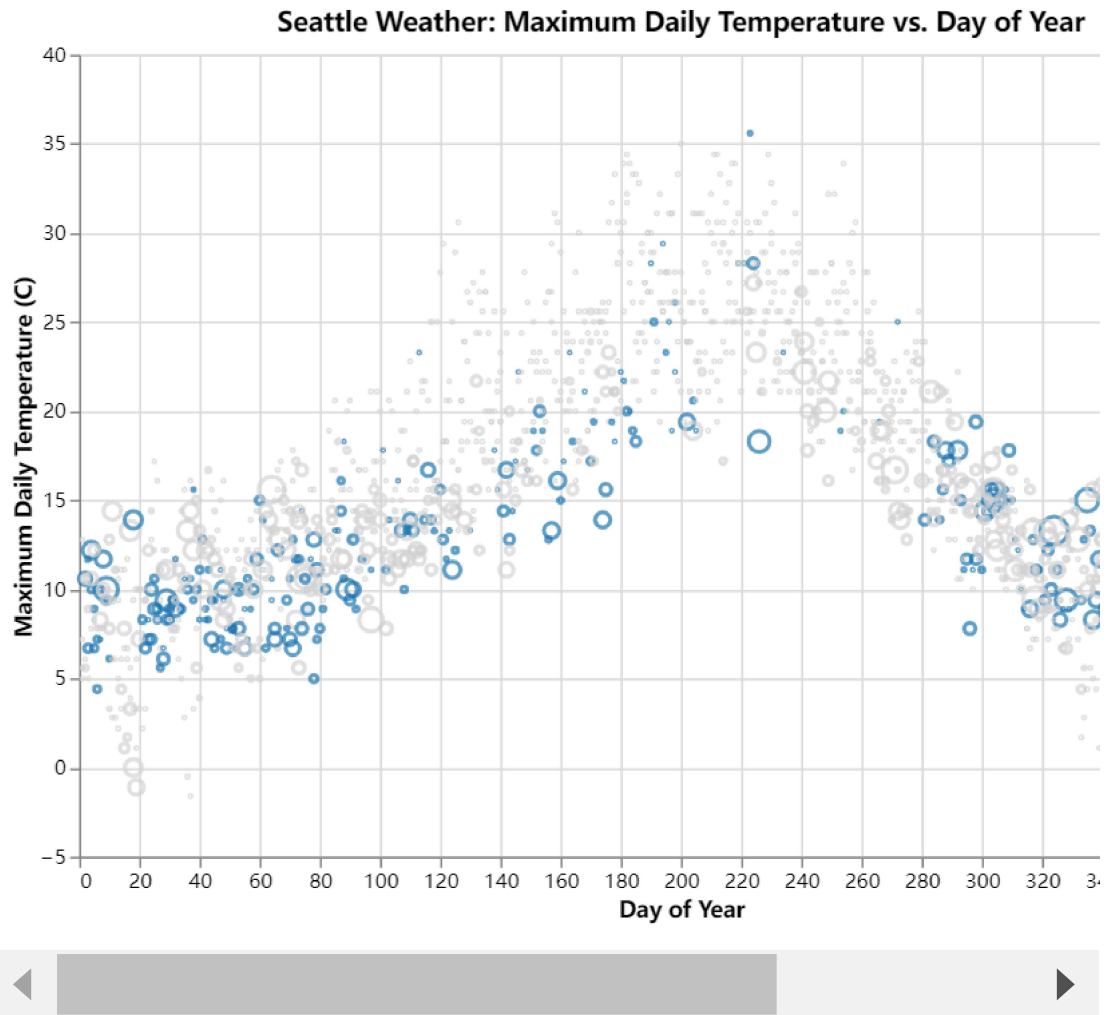
# Create a selection for weather legend
legend_selection = alt.selection_multi(fields=['weather'], bind='legend')

# Chart with weather graying out functionality
points = alt.Chart(source).mark_point().encode(
    alt.X('day_of_year:Q', title='Day of Year'),
    alt.Y('temp_max:Q', title='Maximum Daily Temperature (C)'),
    size=alt.Size('precipitation:Q', scale=alt.Scale(range=[1, 200])),
    color=alt.condition(
        legend_selection,
        alt.Color('weather:N', scale=scale, title='Weather'),
        alt.value('lightgray')
    ),
    tooltip=['date:T', 'temp_max:Q', 'precipitation:Q']
).properties(
    title='Seattle Weather: Maximum Daily Temperature vs. Day of Year',
    width=600,
    height=400
).add_selection(
    nearest
).add_selection(
    legend_selection
)

# Display the chart
points
```

```
D:\annaconda\Lib\site-packages\altair\utils\deprecation.py:65: AltairDeprecat
ionWarning: 'selection_single' is deprecated. Use 'selection_point'
  warnings.warn(message, AltairDeprecationWarning, stacklevel=1)
D:\annaconda\Lib\site-packages\altair\vegalite\v5\api.py:398: AltairDeprecati
onWarning: The value of 'empty' should be True or False.
  warnings.warn(
D:\annaconda\Lib\site-packages\altair\utils\deprecation.py:65: AltairDeprecat
ionWarning: 'selection_multi' is deprecated. Use 'selection_point'
  warnings.warn(message, AltairDeprecationWarning, stacklevel=1)
D:\annaconda\Lib\site-packages\altair\utils\deprecation.py:65: AltairDeprecat
ionWarning: 'add_selection' is deprecated. Use 'add_params' instead.
  warnings.warn(message, AltairDeprecationWarning, stacklevel=1)
```

Out[20]:



Part 3: Interval Selection (5 points)

Modify the top chart so that if you create a selection of data points by clicking and dragging your mouse, you will see the bottom horizontal bar chart reflect the data points in the selection only.

Let's see if we can add another layer of interactivity to the top chart. This time, I want to be able to select an interval of data points on the top chart by clicking-and-dragging to make a selection. Similar to how one might select multiple files on a computer screen. Your task for this part is to modify the top chart such that if you create a selection of data points by clicking and dragging your mouse, you will see: . The bottom horizontal bar chart reflect the data points in the selection only. For example, if I collect a sample of points such that the data from February 10 to February 20 is selected, I should only see Feb 10 to Feb 20 on the y-axis in the bottom chart. . All points outside the selection will be grayed out, similar to Part 2. We should also make it so that if I click on a weather type in the legend, only the points in my selection will have their colors adjusted; all other points outside the selection are still grayed out.

```

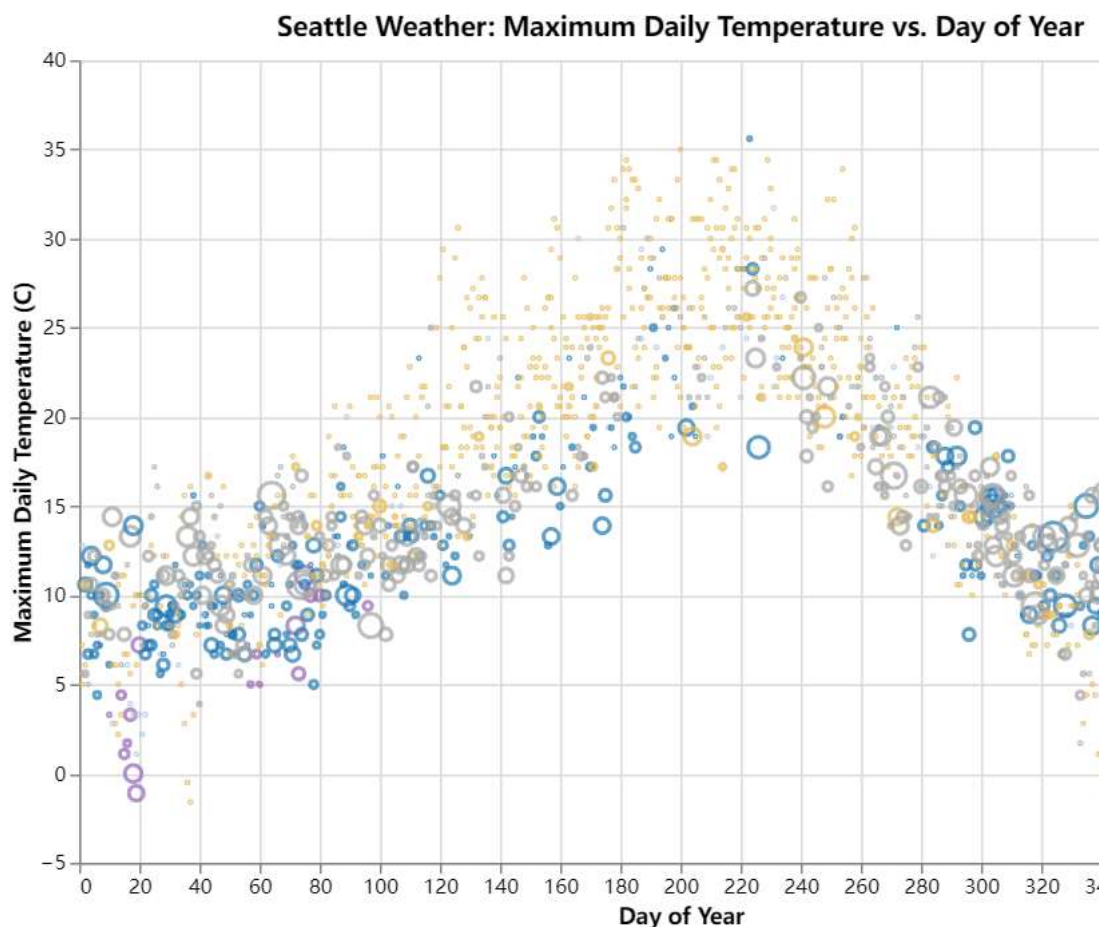
In [21]: ▶ # Create the scatter plot
points = alt.Chart(source).mark_point().encode(
    alt.X('day_of_year:Q', title='Day of Year'),
    alt.Y('temp_max:Q', title='Maximum Daily Temperature (C)'),
    size=alt.Size('precipitation:Q', scale=alt.Scale(range=[1, 200])),
    color=alt.Color('weather:N', scale=scale, title='Weather'),
    tooltip=['date:T', 'temp_max:Q', 'precipitation:Q']
).properties(
    title='Seattle Weather: Maximum Daily Temperature vs. Day of Year',
    width=600,
    height=400
)

# Create the stacked horizontal bar chart
bars = alt.Chart(source).mark_bar().encode(
    alt.X('sum(precipitation):Q', title='Sum of Precipitation'),
    alt.Y('monthdate(date):O', title='Date'),
    color=alt.Color('weather:N', scale=scale, title='Weather'),
    order=alt.Order('weather', sort='ascending'),
    tooltip=['date:T', 'precipitation:Q', 'weather:N']
).properties(
    title='Total Precipitation by Date',
    width=600,
    height=400
)

# Show the scatter plot and the stacked bar chart side by side
alt.hconcat(points, bars)

```

Out[21]:



```

In [22]: ▶ # Create a brush selection for the top chart
brush = alt.selection_interval(encodings=['x'])

# Create a selection for weather legend
legend_selection = alt.selection_multi(fields=['weather'], bind='legend')

# Chart with interval selection and weather graying out functionality
points = alt.Chart(source).mark_point().encode(
    alt.X('day_of_year:Q', title='Day of Year'),
    alt.Y('temp_max:Q', title='Maximum Daily Temperature (C)'),
    size=alt.Size('precipitation:Q', scale=alt.Scale(range=[1, 200])),
    color=alt.condition(
        legend_selection & brush,
        alt.Color('weather:N', scale=scale, title='Weather'),
        alt.value('lightgray')
    ),
    tooltip=['date:T', 'temp_max:Q', 'precipitation:Q']
).properties(
    title='Seattle Weather: Maximum Daily Temperature vs. Day of Year',
    width=600,
    height=400
).add_selection(
    brush
).add_selection(
    legend_selection
)

# Create the stacked horizontal bar chart with a filter based on brush selection
bars = alt.Chart(source).mark_bar().encode(
    alt.X('sum(precipitation):Q', title='Sum of Precipitation'),
    alt.Y('monthdate(date):O', title='Date'),
    color=alt.Color('weather:N', scale=scale, title='Weather'),
    order=alt.Order('weather', sort='ascending'),
    tooltip=['date:T', 'precipitation:Q', 'weather:N']
).transform_filter(
    brush
).properties(
    title='Total Precipitation by Date',
    width=600,
    height=400
)

# Show the scatter plot and the filtered stacked bar chart side by side
alt.hconcat(points, bars)

```

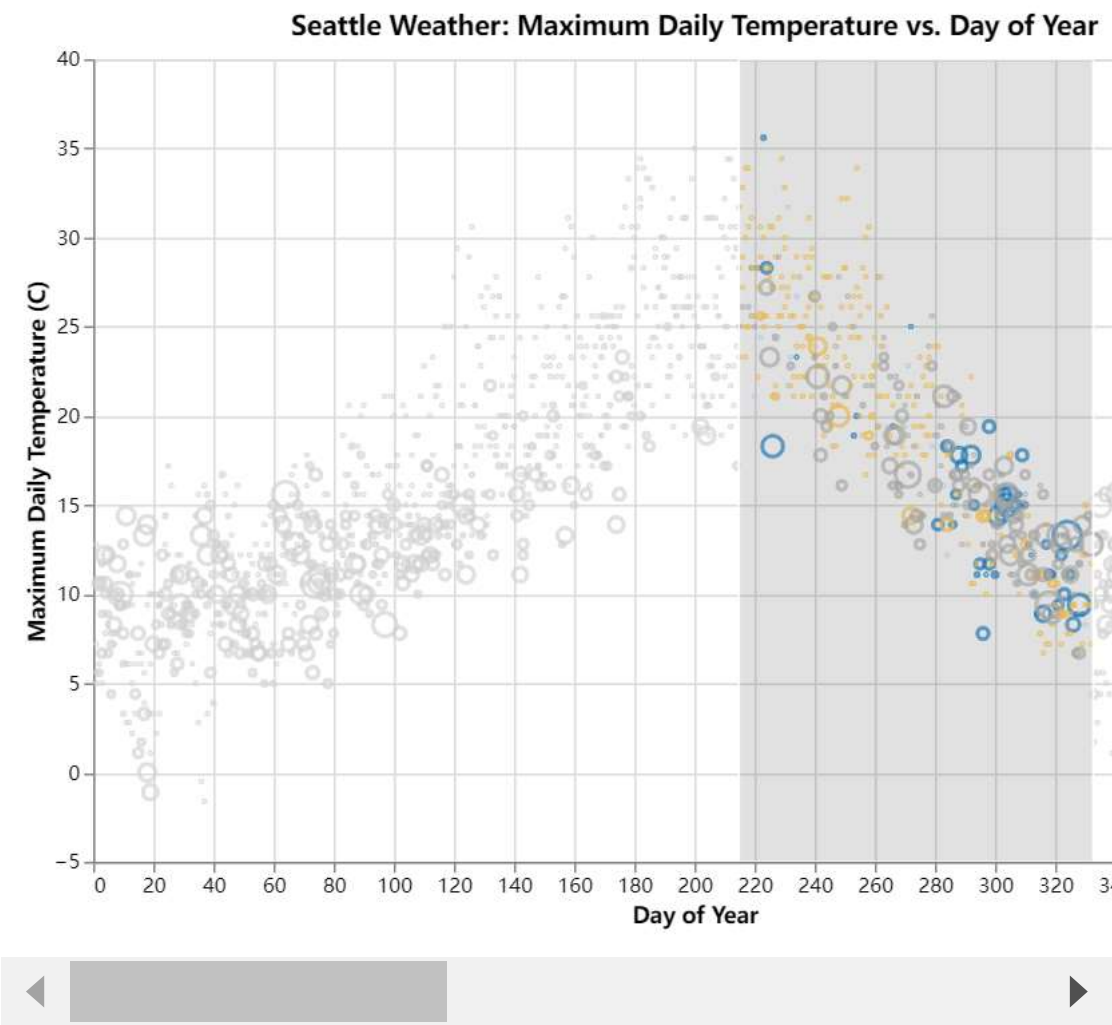
D:\annaconda\Lib\site-packages\altair\utils\deprecation.py:65: AltairDeprecationWarning: 'selection_multi' is deprecated. Use 'selection_point'

warnings.warn(message, AltairDeprecationWarning, stacklevel=1)

D:\annaconda\Lib\site-packages\altair\utils\deprecation.py:65: AltairDeprecationWarning: 'add_selection' is deprecated. Use 'add_params' instead.

warnings.warn(message, AltairDeprecationWarning, stacklevel=1)

Out[22]:



In []:

▶

In []:

▶

In []:

▶

In []:

▶