SI649-20w Lab 4 -> Altair III

School of Information, University of Michigan

Assignment Overview

1. Interaction
2. Review transform

We will replicate 3 visualizations (and a bonus visualization) created by the article posted on Five Thirty Eight available online (Hickey, Koeze, Dottle, Wezerek 2017).

For this lab, please write Altair code to answer the questions. It's fine if your visualization looks slightly different from the example (e.g., getting 1.1 instead of 1.0, use orange instead of red, have different titles, chart width/height,and mark size/opacity)

Also, it's possible that your chart does not show the desired interactivity because of altair/colab issues. Check the potential bug section for more detail.

Resources:

- Article by Five Thirty Eight available [online] (https://fivethirtyeight.com/features/the-mayweather-mcgregor-fight-as-told-through-emojis/) (Hickey, Koeze, Dottle, Wezerek 2017)
- the original can be found on [Five Thirty Eight Mayweather vs McGregor] (https://github.com/fivethirtyeight/data/tree/master/mayweather-mcgregor)

General Hints:

- Yes, you can render emojis in colab and in your chart. You can consider them as text.
- We recommend that you finish all the static charts before adding interactions.
- If you see duplicated axes, use `axis=None` to get rid of unnecessary axes.
- Don't forget to set `empty="none"` . The default behavior is that when nothing is selected, *everything* is selected. When set to none, empty selections contain no data values.
- `resolve_scale` ensures charts share axes and scales.

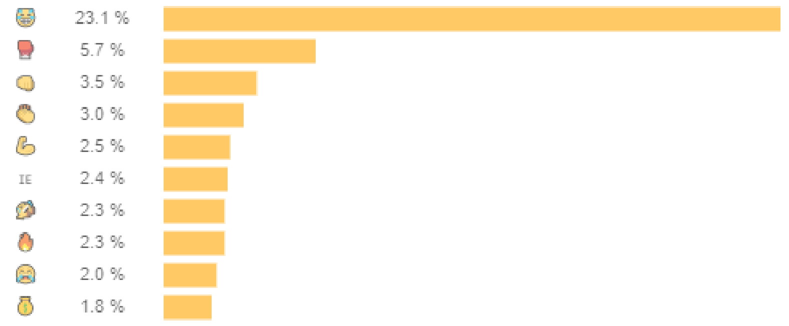
```
# start with the setup
import pandas as pd
import altair as alt
import numpy as np
```

```
#load data
```

```
df1=pd.read_csv("https://raw.githubusercontent.com/LiciaHe/SI649/master/week4/data/df1.csv")
df2=pd.read_csv("https://raw.githubusercontent.com/LiciaHe/SI649/master/week4/data/df2_count.csv")
df3=pd.read_csv("https://raw.githubusercontent.com/LiciaHe/SI649/master/week4/data/df3.csv")
df4=pd.read_csv("https://raw.githubusercontent.com/LiciaHe/SI649/master/week4/data/df4.csv")
```

Visualization 1: Emoji and percentage of usage

We will replicate the following visualization

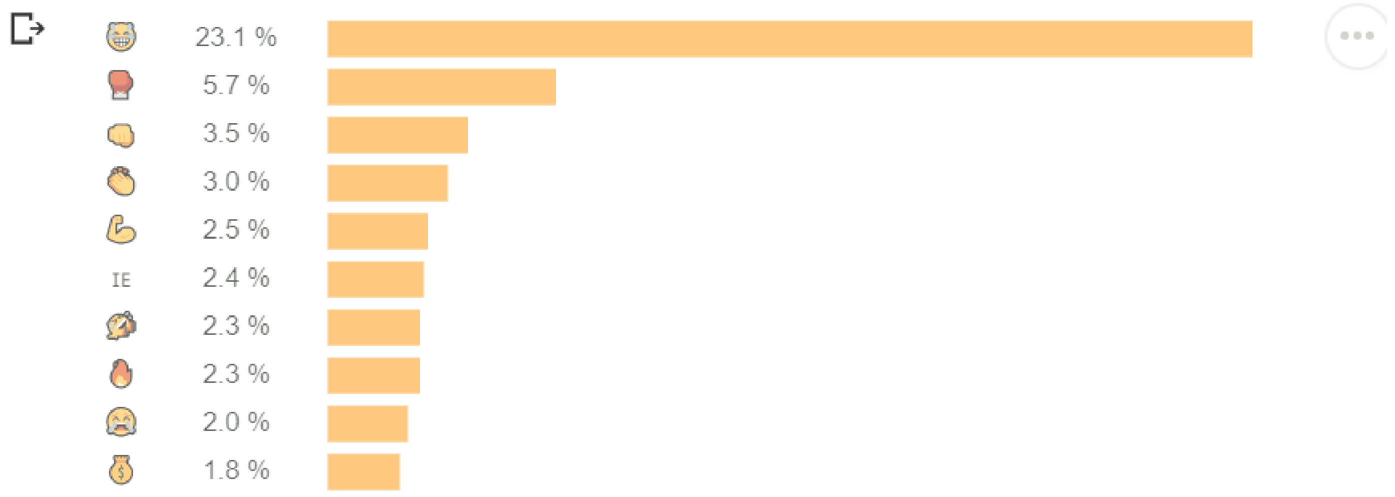


Description of the visualization (static):

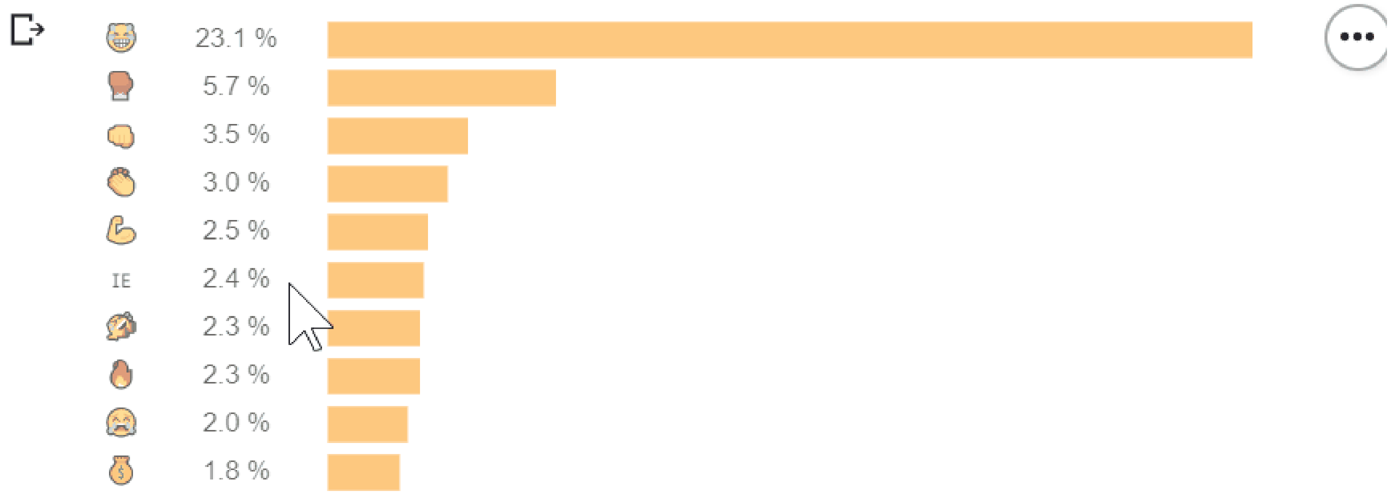
- Use *df1* for this exercise
- This visualization has 3 components: **emoji**, **percentage text**, and **bars**
- All 3 components share the same y axis, which display the *rank* of percentage from highest to lowest.
- The width of the bar(along x axis) encodes *PERCENT*
- All 3 components have a low opacity because we want to add interactions (see the next cell).

Description of the visualization (interactivity):

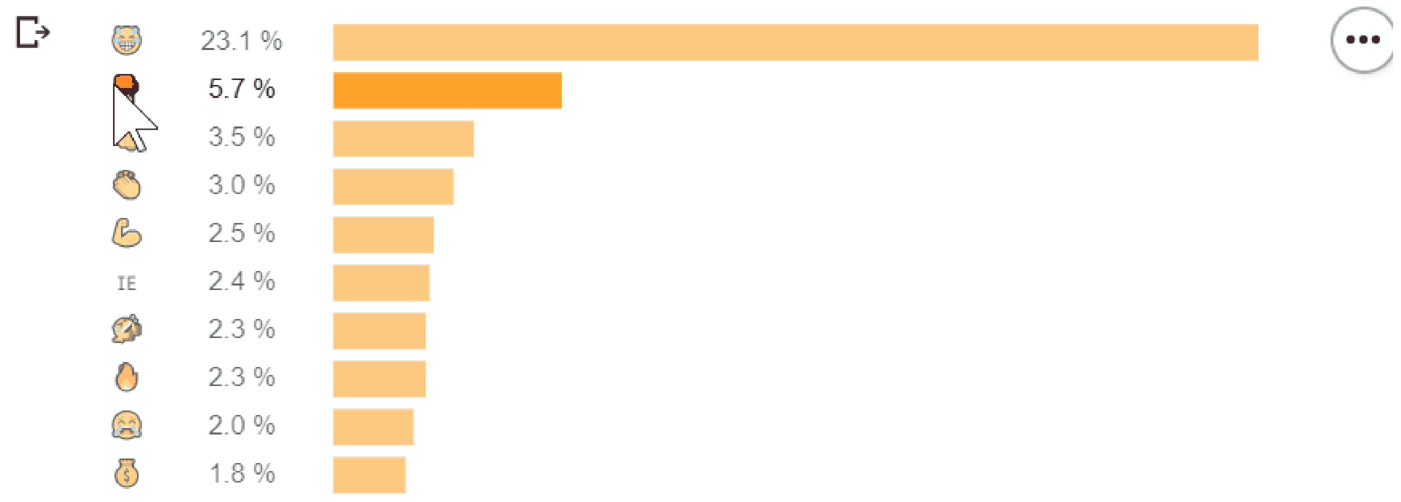
1. When hovering over bars, the associated emoji will show up as tooltip



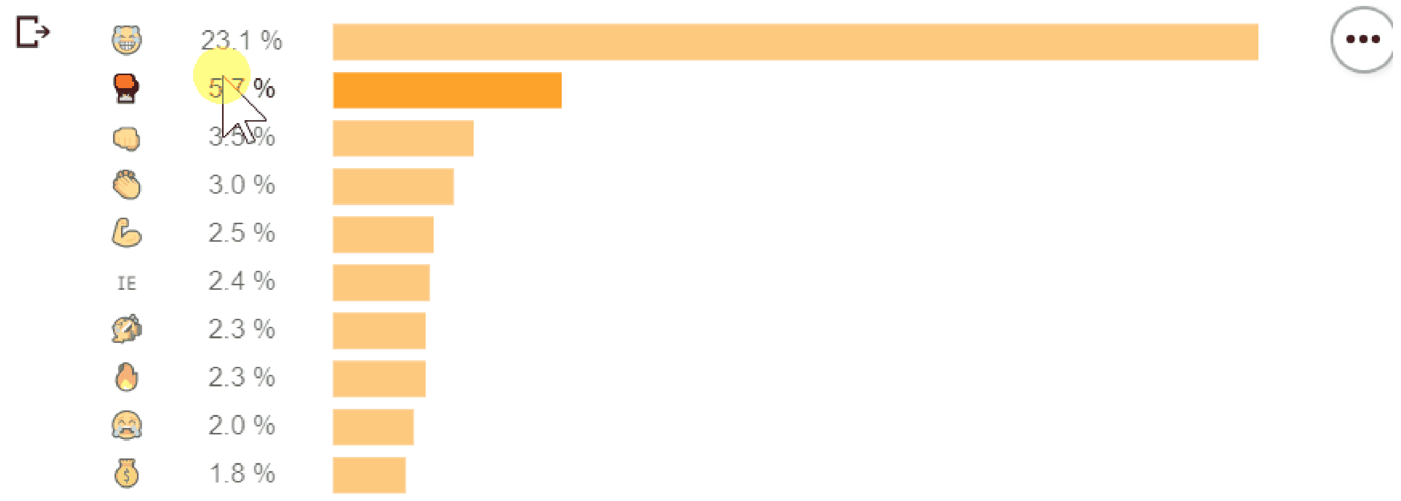
2. When hovering over emojis, percentage texts, or bars, the opacity of the selected row will change to 1.



3. Brushing over the emojis will filter bars.



4. Brushing over percentage text will filter bars.



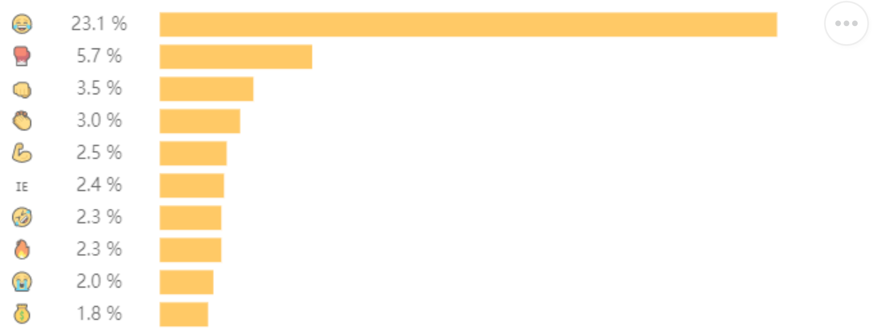
Sample style settings (optional): Here's a list of default style settings we used to generate the graph.

- Original opacity for all 3 components: 0.6.
- Hovered opacity: 1
- bar height = 15, color = orange
- text chart and emoji width are both 20
- after building the compound chart, use the following line to disable border : `.configure_view(strokeWidth=0)`

Hint:

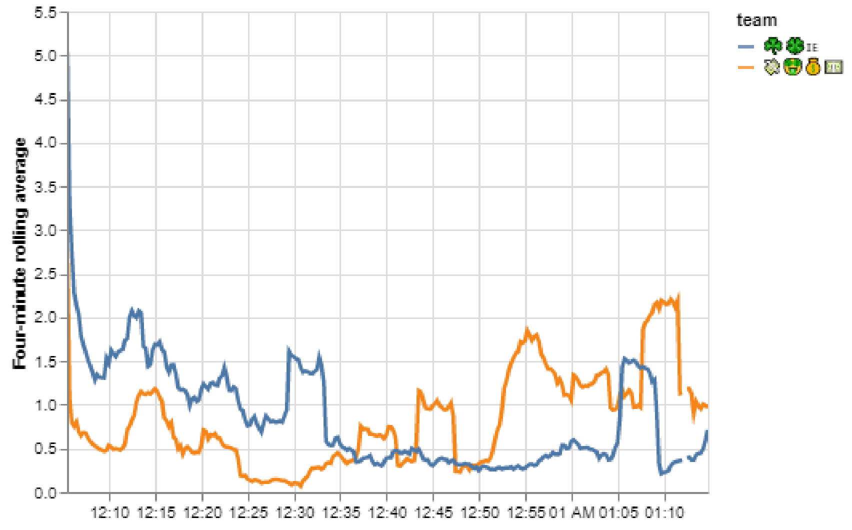
- We recommend getting all static components working before writing any interactivity.
- Add one interaction at a time and test whether or not it works.
- To add an interaction that's not tooltip and zooming, you need four steps (review in-class demo).
- Selection is used in two scenarios: 1) to add to a *condition*, which is used in `encode`. 2) to add in `transform_filter`. In this visualization, you will implement both. Think through which you will use where before trying to build this.

```
##TODO: replicate vis 1
selection=alt.selection_single(empty="none",on="mouseover")
opacityCondition = alt.condition(selection,alt.value(1.0),alt.value(0.6))
selection2=alt.selection_interval(
        encodings=['y'],
)

df1 = df1.sort_values('rank')

emoji = alt.Chart(df1).add_selection(
        selection
).add_selection(
        selection2
).mark_text().transform_calculate(
        zero = "0*datum.PERCENT"
).encode(
        x = alt.X("zero:Q",axis=None),
        y = alt.Y("rank:O",sort=alt.EncodingSortField(field="rank"),title='',axis=None),
        text = alt.Text("EMOJI"),
        opacity = opacityCondition,
).properties(
        width=20
)

text = emoji.encode(
        text = alt.Text("PERCENT_TEXT"),
        opacity = opacityCondition
)


bar = alt.Chart(df1).add_selection(
        selection
).add_selection(
        selection2
).mark_text().transform_calculate(
        zero = "0*datum.PERCENT"
).mark_bar(height=15, color="orange", opacity=0.6).encode(
        x = alt.X("PERCENT:Q",axis=None),
        y  =  alt.Y("rank:O",sort=alt.EncodingSortField(field="rank"),title='',axis=None),
        tooltip  =  alt.Tooltip('EMOJI'),
        opacity  =  opacityCondition
).transform_filter(
        selection2
)


(emoji  |  text  |  bar).resolve_scale(y="shared").configure_view(strokeWidth=0)
```



Visualization 2: Irish Pride vs. Money Team

We will replicate the following visualization

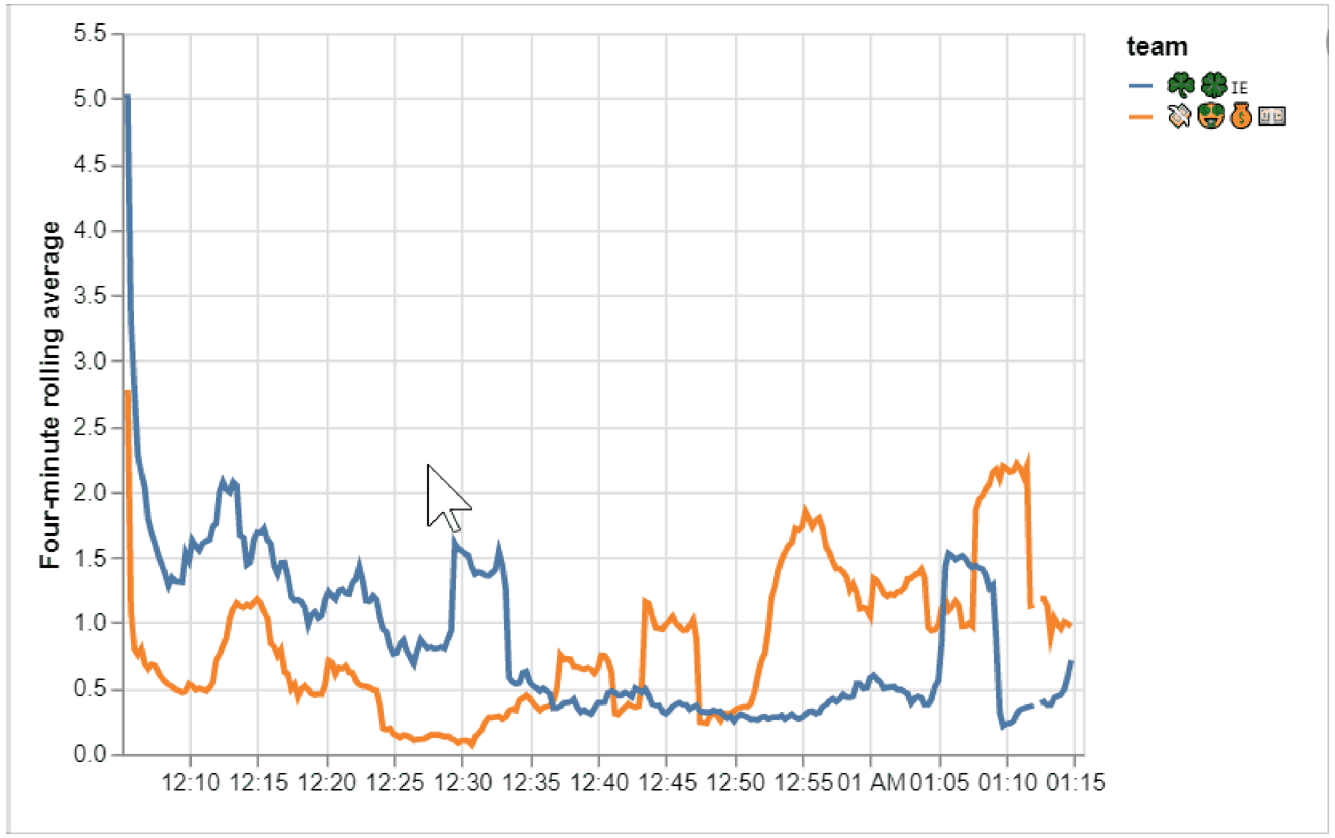


Description of the visualization (static):

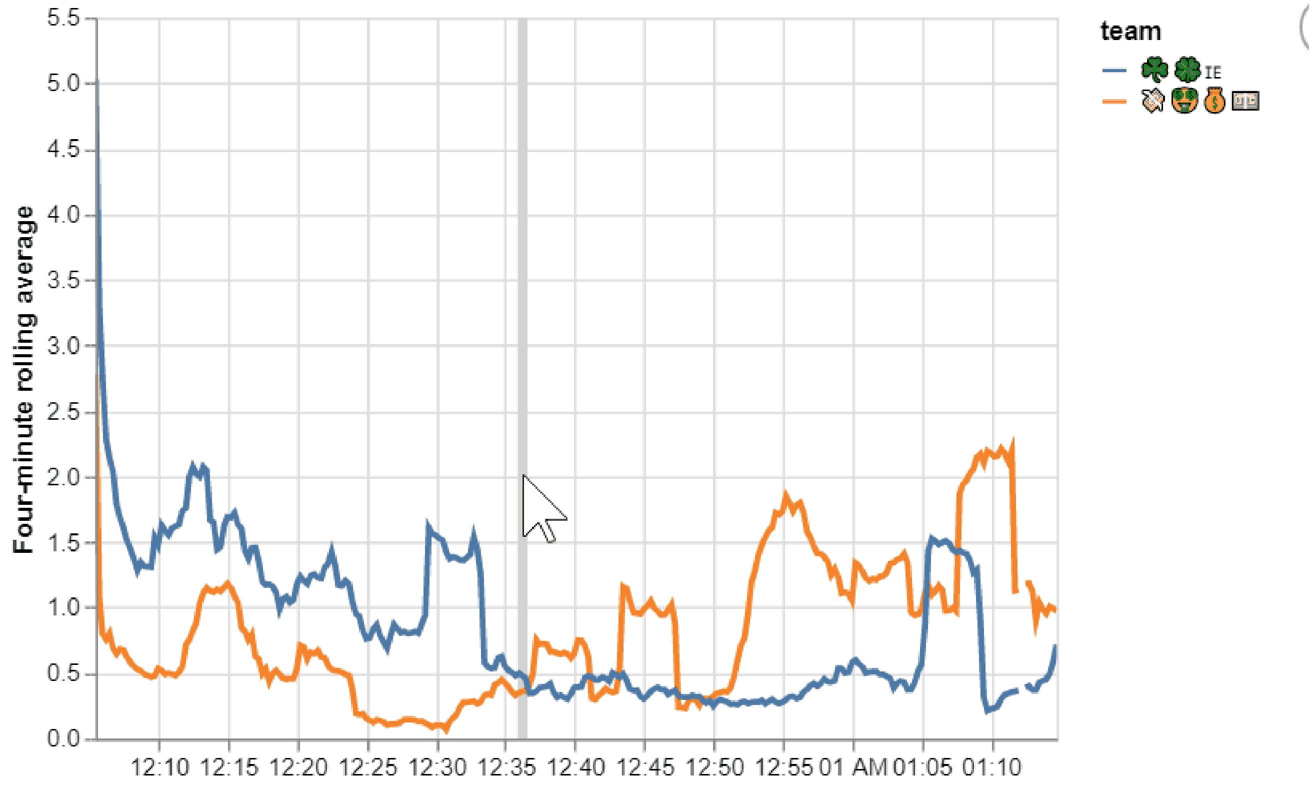
- Use *df2*
- The visualization has 1 "static" component: **line chart**. It displays the relationship between *datetime* and *count*

Description of the visualization (interactivity):

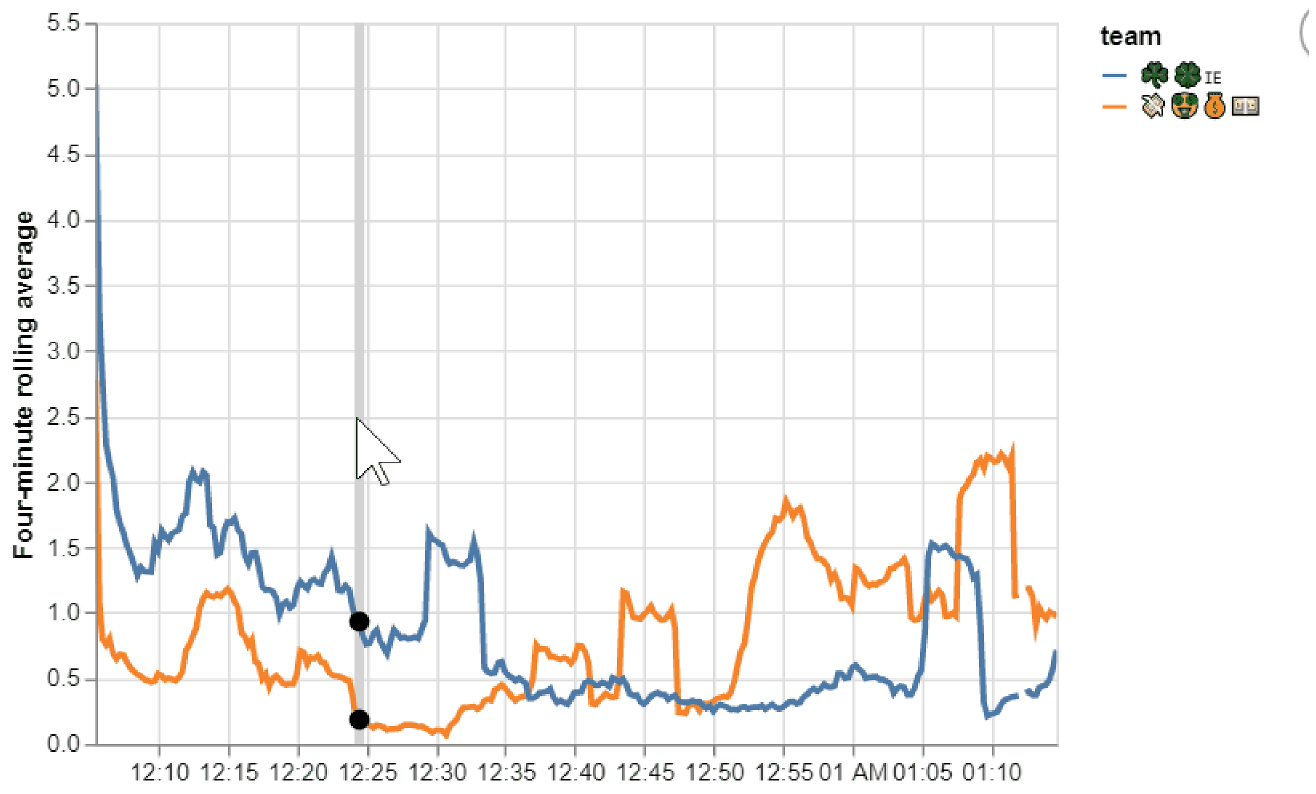
1. Enable zooming and panning along the x-axis. (The gif below only displays the line chart.)



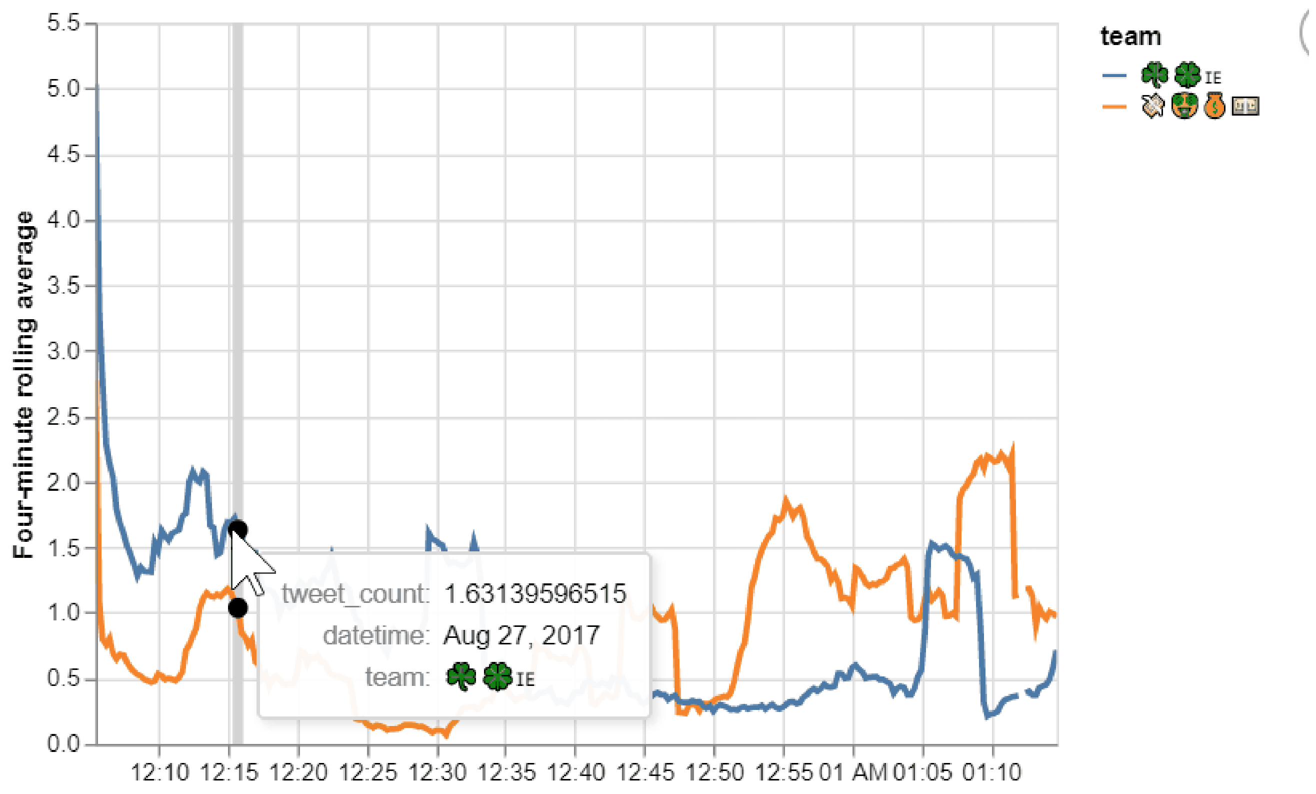
2. Display a vertical line that moves with the mouse. This will require you to add additional chart component(let's call it **vLine**).



3. Display the intersection of the **vLine** with the **line chart** as 2 circles (let's call these two circles **intersection dots**).



4. When hovering over these **intersection dots**, display the *tweet_count*, *datetime*, and *team* in tooltip.



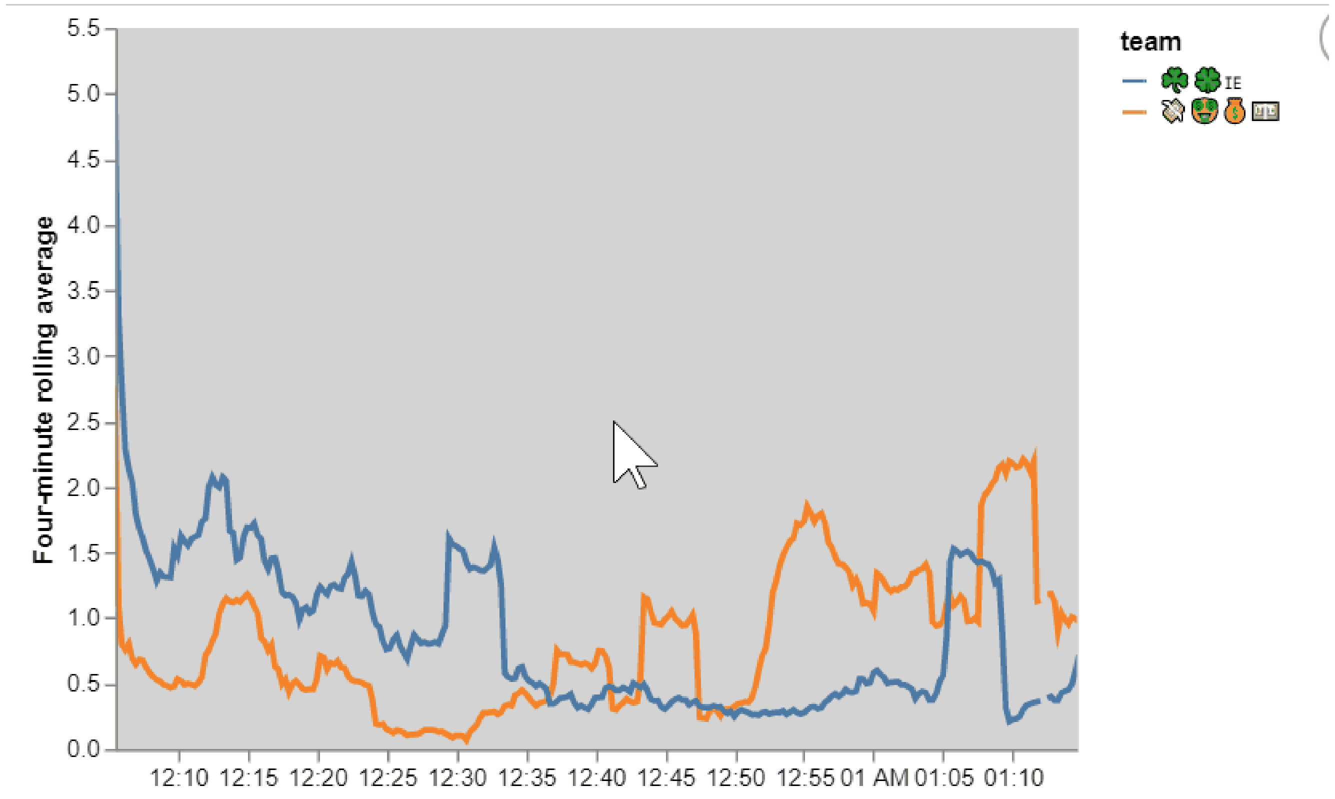
Sample style settings (optional): Here's a list of default style settings we used to generate the graph.

- line chart size = 2.5,
- vLine: size=4, color="lightgray", initial opacity = 0
- indicator dot: color="black" size=70

Bugs

If your interaction look similar to this, don't worry about it. It's likely a altair/colab issue.

1. Blinking lines: You can fix by making your **vLine** thicker, or view it in vegalite editor.



- If your tooltip does not appear, make sure you have a selection bond to the chart that has the tooltip encoding.
- If your are getting an error: "avascript Error: Duplicate signal name", it means you added your selections more than once. Remember, when layering charts together, they will share their selections. Therefore, don't add the same selection to both charts.

Hint

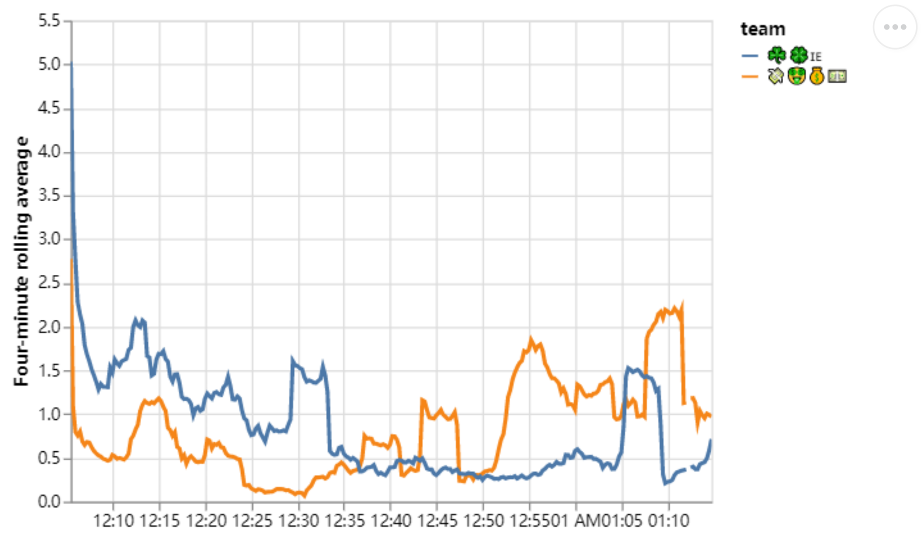
- We only want to enable zooming and panning along the x-axis.
- There are multiple ways of implementing the **vLine**. Here is one of them:

  > 1) use mark_rule to generate a line for every single data point and set these line's opacity to be 0.

  > 2)when mouse hovering over a line, display it by changing its opacity.

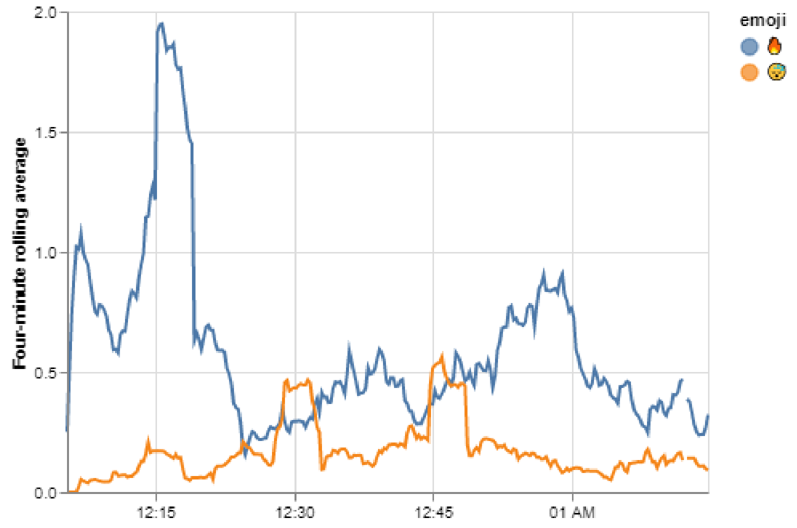
- The implementation of the **intersection dots** is similar to that of the **vLine**. Do you need a new selection/condition for the **intersection dots**?

```
#TODO: replicate vis2
selectionScale=alt.selection_interval(
        bind = "scales", # alter scales when zooming
        encodings = ["x"]
)

selectionHover=alt.selection_single(
        empty="none",
        on='mouseover',
        nearest=True,
        encodings = ["x"]
)

opacityCondition = alt.condition(selectionHover,alt.value(1.0),alt.value(0))


lineChart = alt.Chart(df2).add_selection(
        selectionScale
).mark_line(size=2.5).encode(
    x = alt.X("datetime:T",title=''),
    y = alt.Y("tweet_count:Q",title='Four-minute rolling average'),
    color = "team:N",
)

vline = alt.Chart(df2).add_selection(
        selectionHover
).mark_rule(size=4, color="lightgray").encode(
    x = alt.X("datetime:T",title=''),
    #y = alt.Y("tweet_count:Q",title='Four-minute rolling average'),
    opacity = opacityCondition
)

intersectDots = alt.Chart(df2).mark_circle(color="black",size=70).encode(
    x = alt.X("datetime:T",title=''),
    y = alt.Y("tweet_count:Q"),
    tooltip = ["tweet_count",alt.Tooltip('datetime',timeUnit='yearmonthdate',title="datetime"),"team"],
    opacity = opacityCondition,
).interactive()

lineChart + vline + intersectDots
```



Visualization 3: Much hype, more boredom

We will replicate the following visualization



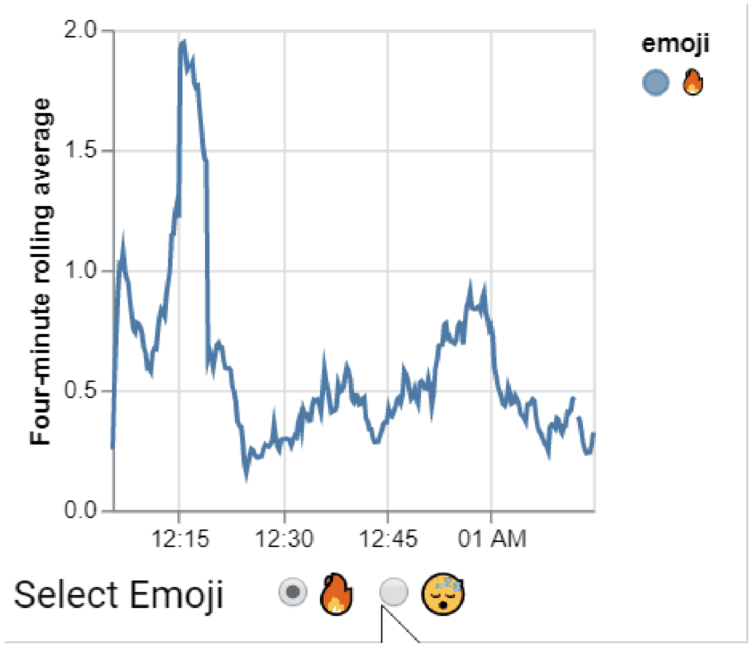
Description of the visualization (static):

- Use *df3*

- The visualization has 1 component: **line chart**. It displays the relationship between *datetime* and *tweet_count*. Each line represents one

Description of the visualization (interactivity):

1. Build radio selections for emojis. Theoretically, only one line will be shown at any given time. See the "bug" section for more detail.



2. Brushing over line chart will display individual data points as circles. This will require another chart component, let's call this component **circles**.

Sample style settings (optional): Here's a list of default style settings we used to generate the graph.

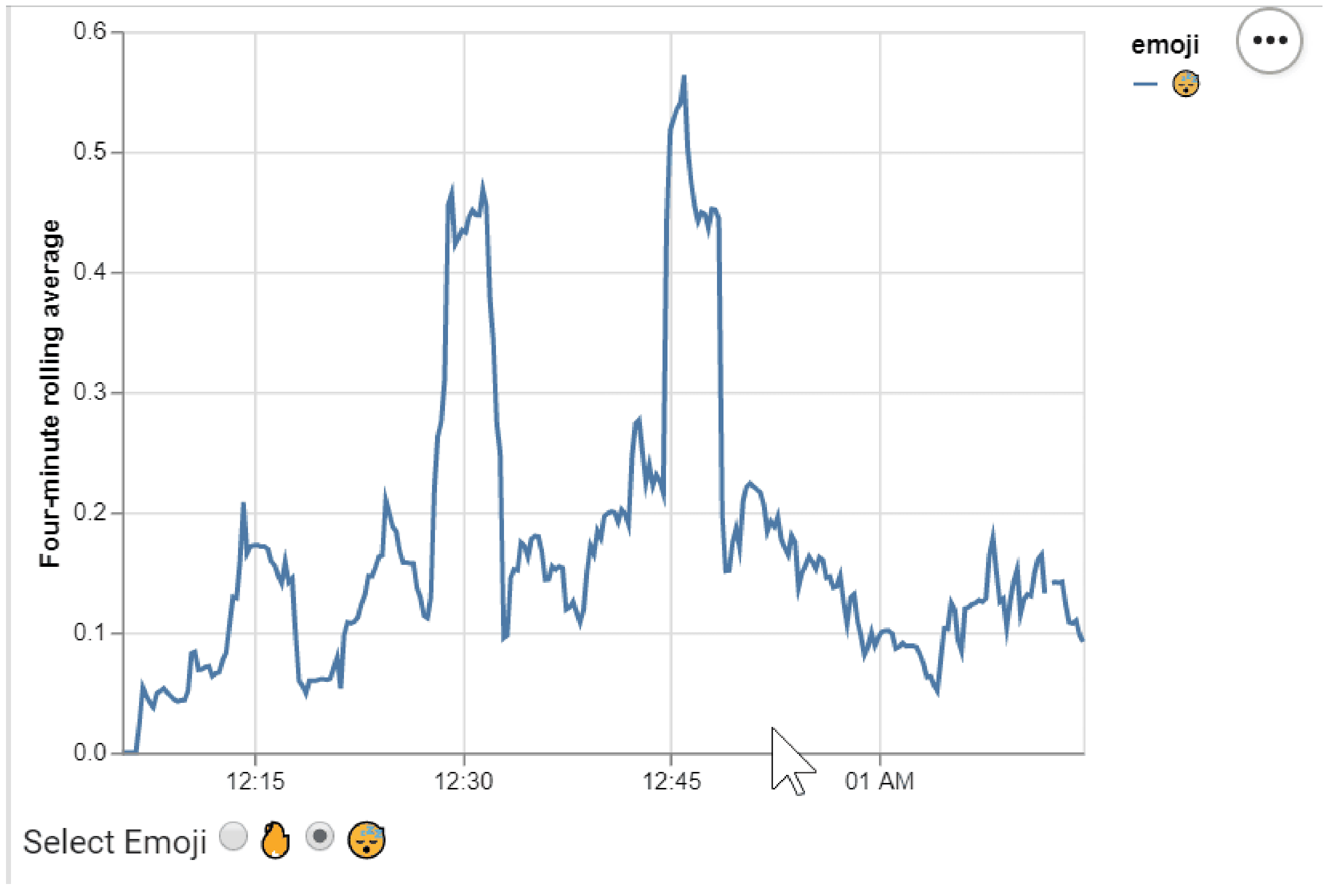
- **circles**: color="black",opacity=0.7
- For both x and y axis, we have the following tickCount specified

> axis=alt.Axis(tickCount=5,...)

Bugs

If your interaction look similar to this, don't worry about it. It's likely a altair/colab issue.

1. single click reset selection even when *clear=False*. This is likely a colab issue, because the chart will behave correctly in vega-editor. You can ignore this bug or view it in vegalite online editor.

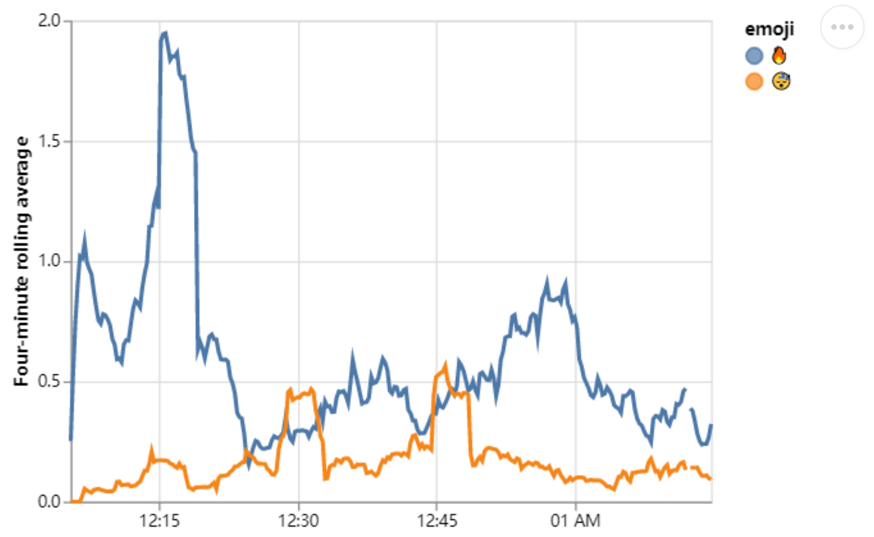


Hint

- You will have 2 selections. One for the emoji selection and one for the time selection. Ensure that these two interactions work independently before merging them together.
- You can use multiple transform_filter, or use logical operand to chain multiple selections.
- There are only 2 emojis in this chart, you can set a variable and refer to it in your radio selection.

> domain = ['🔥','😴']

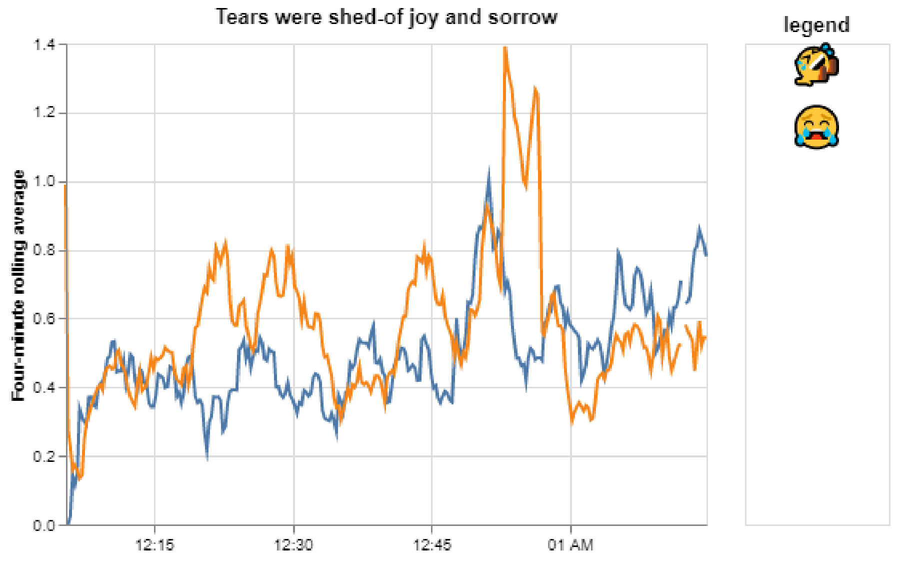
```
##TODO: replicate vis3
selectEmoji=alt.selection_single(
        fields=['emoji'],
        # notice the binding_radio
        bind=alt.binding_radio(options=['🔥','😴'],name="Select Emoji"),
)

brush = alt.selection_interval(empty="none")

circleLegend = alt.Chart(df3).mark_circle(size=0).encode(
        x = alt.X("datetime:T",title='',axis=alt.Axis(tickCount=5)),
        y = alt.Y("tweet_count:Q",title='Four-minute rolling average',axis=alt.Axis(tickCount=5)),
        color = alt.Color("emoji:N"),
)

lineChart = circleLegend.add_selection(
        selectEmoji
).mark_line(size=2.5).encode(
        x = alt.X("datetime:T",title='',axis=alt.Axis(tickCount=5)),
        y = alt.Y("tweet_count:Q",title='Four-minute rolling average',axis=alt.Axis(tickCount=5)),
        color = alt.Color("emoji:N"),
).transform_filter(
        selectEmoji
)

circles = alt.Chart(df3).add_selection(brush).mark_circle(color="black",opacity=0.7).encode(
        x = alt.X("datetime:T",title='',axis=alt.Axis(tickCount=5)),
        y = alt.Y("tweet_count:Q",title='Four-minute rolling average',axis=alt.Axis(tickCount=5)),
).transform_filter(brush).transform_filter(selectEmoji)

circleLegend + lineChart + circles
```



Select Emoji ○ 🔥 ○ 😵

BONUS: Visualization 4: Tears were shed-of joy and sorrow

OPTIONAL: We will replicate the following visualization



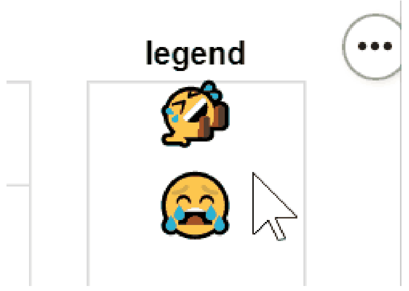
Description of the visualization (static):

- Use *df4*

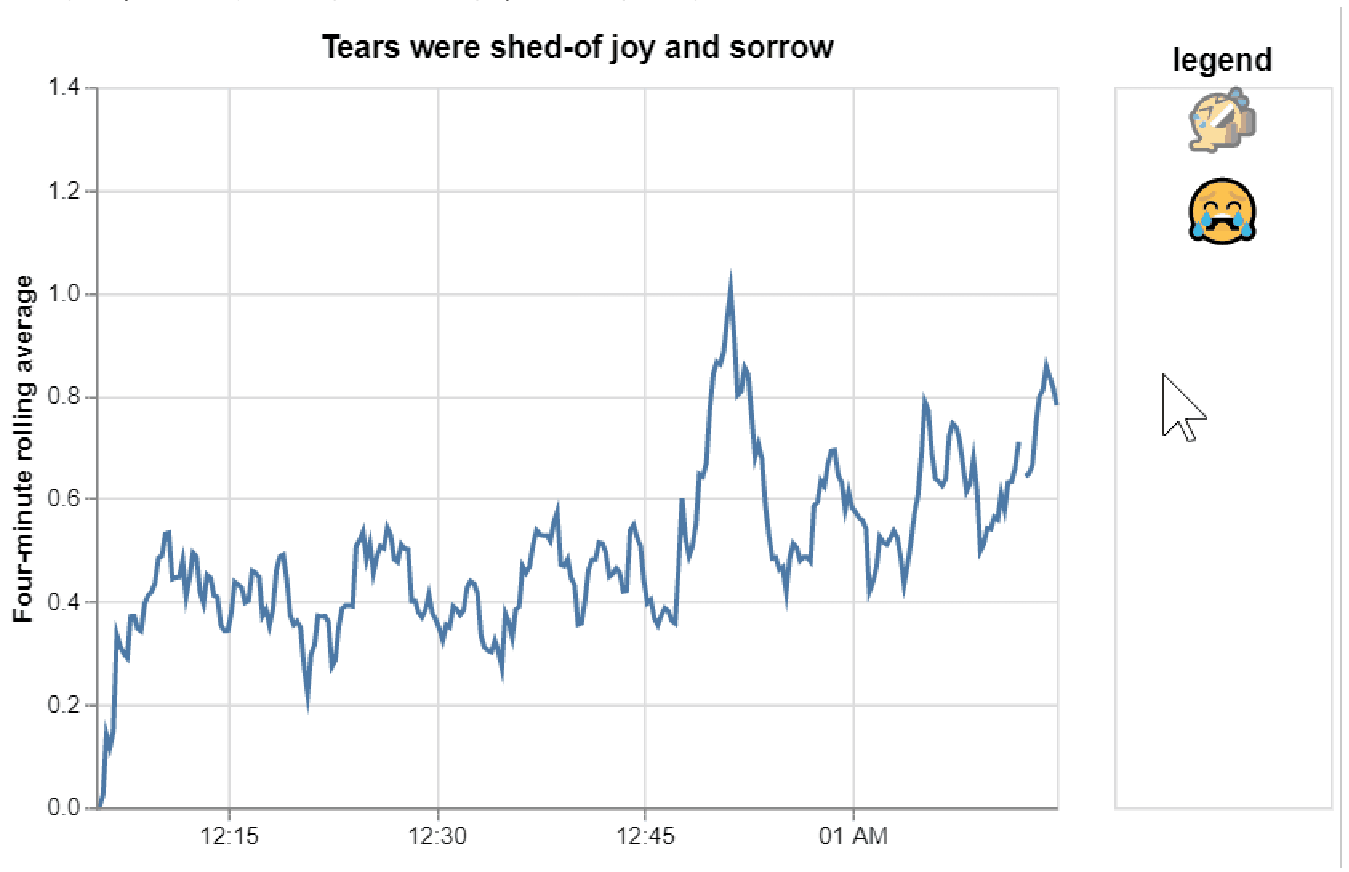
- The visualization has 2 components: **line chart** that displays relationship between *datetime* and *tweet_count*, and **legend** that displays the two emojis in the line chart.
- **legend** is a chart. It is not the automatically generated legend. (i.e., In the line chart's color encoding, set *legend=None*)
- Two components are displayed side by side

Description of the visualization (interactivity):

1. In the legend component, if one of the emoji is clicked, the selected emoji will have full opacity while the other emoji becomes transparent.



2. Clicking emojis in the **legend** component will display the corresponding line and hide the other line .



Sample style settings (optional): Here's a list of default style settings we used to generate the graph.

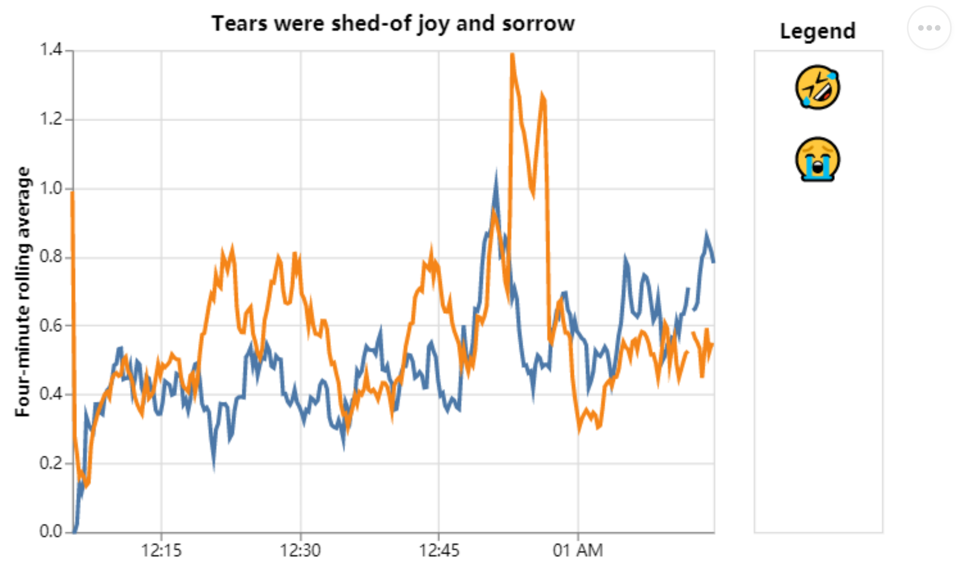
- For the line chart, we used tickCount=5 in x and y axis.
- For the legend, we used size=25,strokeWidth=0

Hint

- You can have one selection and multiple conditions that use the same selection.
- To hide a line, you have 2 options: changing opacity and adding transform_filter. The sample answer is generated with the first option. If you use the second option, you will get a slightly different view (because axes will be adjusted) and it's totally ok.

```
#TODO: BONUS: Replicate vis 4
selectEmoji=alt.selection_multi(fields=['emoji'])
opacityCondition = alt.condition(selectEmoji,alt.value(1.0),alt.value(0.5))
```

```
lineChart = alt.Chart(df4, title='Tears were shed-of joy and sorrow').mark_line(size=2.5).encode(
        x = alt.X("datetime:T",title='',axis=alt.Axis(tickCount=5)),
        y = alt.Y("tweet_count:Q",title='Four-minute rolling average',axis=alt.Axis(tickCount=5),scale=alt.Scale(domain=[0, 1.4])),
        color = alt.Color("emoji:N",legend=None),
).transform_filter(selectEmoji)



legend = alt.Chart(df4, title='Legend').add_selection(selectEmoji).mark_text(size=25,strokeWidth=0).encode(
        y = alt.Y("max(tweet_count)",axis=None, scale=alt.Scale(domain=[-1, 1.5])),
        text = 'emoji',
        color = alt.Color("emoji:N",legend=None),
        opacity= opacityCondition
).properties(width=80)

(lineChart | legend)
```



This is the end of lab 4.

Please run all cells (Runtime->Restart and run all), and

1. save to PDF (File->Print->Save PDF -> landscape, shrink to 80%)
2. save to ipynb (File -> Download .ipynb)

Rename both files with your uniqname: e.g. uniqname.pdf/ uniqname.ipynb

Upload both files to canvas.