Information Visualization I

School of Information, University of Michigan

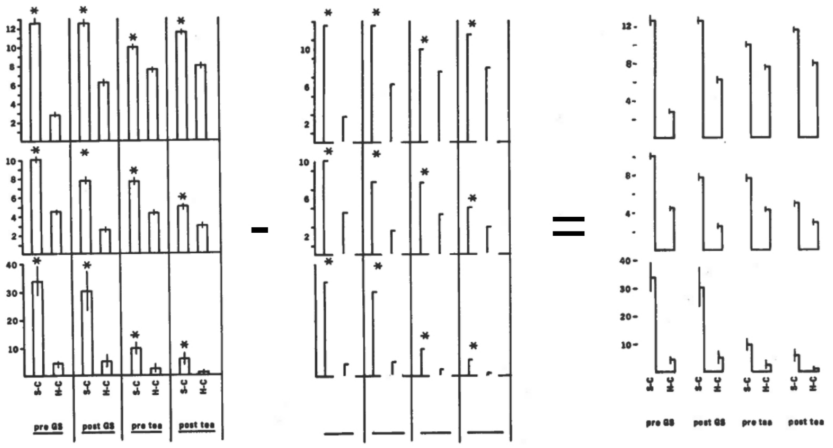
Week 4:

- Data Types
- Design

Assignment Overview

This assignment's objectives include:

- Review, reflect on, and apply the concepts of encoding different data types. Given a visualization, justify different encodings for certain datatypes.
- Review, reflect on, and apply good design decisions in a visualization. Given a visualization critique design decisions according to principles like Tufte's data-ink ratio, graphical integrity, chart junk, Munzner's rules of thumb, etc.



Same information, less ink

- Recreate, propose new and alternative visualizations using Altair

The total score of this assignment will be 100 points consisting of:

- Case study reflection: The Mayweather-McGregor Fight, As Told Through Emojis (30 points)
- Altair programming exercise (70 points)

Resources:

- Article by Five Thirty Eight available online (Hickey, Koeze, Dottle, Wezerek 2017)
- Datasets from Five Thirty Eight, we have download a subset of these datasets to ./assets but the original can be found on Five Thirty Eight Mayweather vs McGregor

Important notes:

1) Grading for this assignment is entirely done by manual inspection. Because there are many ways to generate the correct visualization, we will not be using the autograder for this assignment. Compare your result to the provided samples. Try to get as close to our provided solution as possible.

2) Depending on your operating system and browser combination your emojis may not exactly match ours or the ones from 538. That's fine.

3) When turning in your PDF, please use the File -> Print -> Save as PDF option ***from your browser***. Do ***not*** use the File->Download as->PDF option. Complete instructions for this are under Resources in the Coursera page for this class.

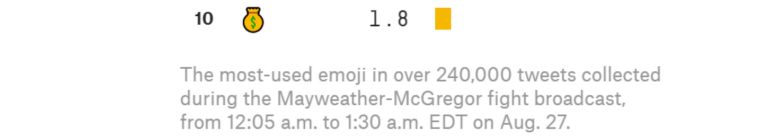
Part 1. Data Types & Design (30 points)

Read the article published in Five Thirty Eight The Mayweather-McGregor Fight, As Told Through Emojis and answer the following questions:

1.1 List the different data types in the following visualizations and their encodings (10 points)

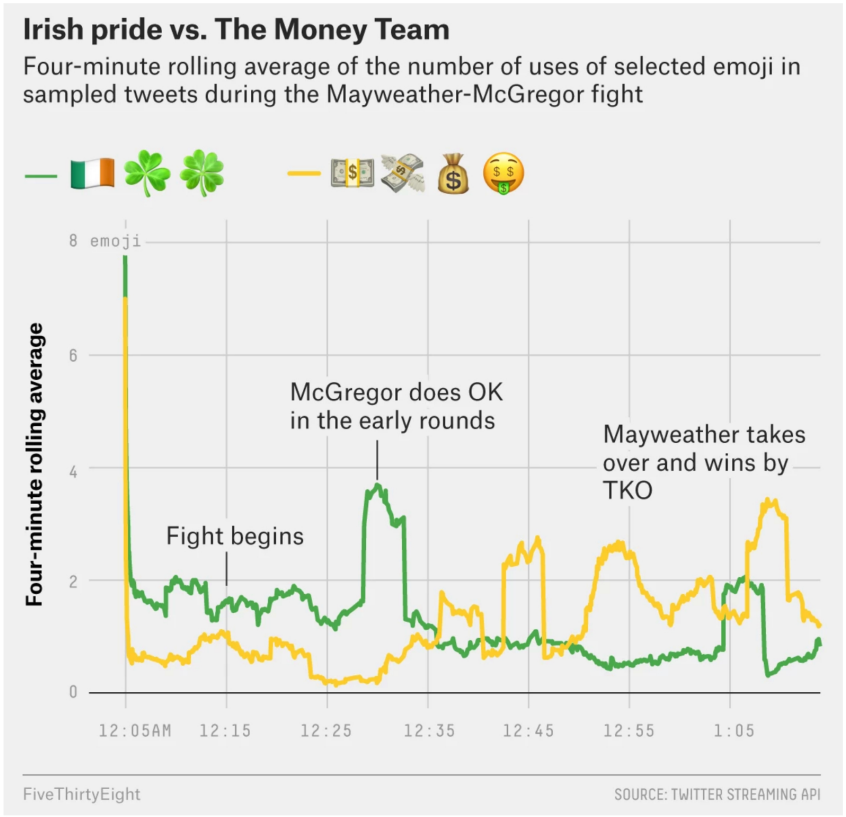
For each chart, describe the variable name, type, and encoding (e.g., weight, quantitative, bar length)

	EMOJI	PERCENT
1	😂	25.2%
2	🥊	5.6
3	👊	3.4
4	👏	3.3
5	🤣	2.5
6	💪	2.5
7	IE	2.4
8	🔥	2.3
9	😭	2.1



Encoding Annotations (Chart Above)

- Emoji, Ordinal, Position(y-axis)
- Emoji Ranking (descending), Ordinal, Position (y-axis)
- Percent, Quantitative, bar length and text

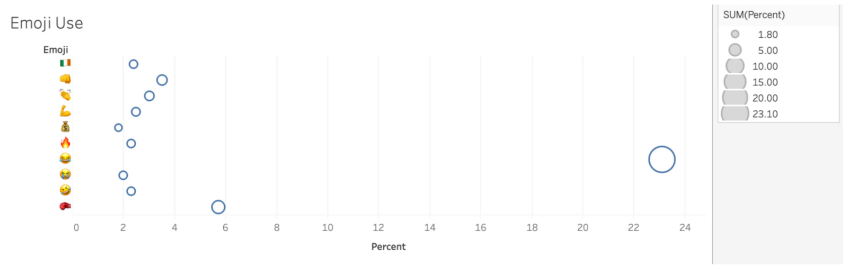


Encoding Annotations (chart above)

- Team, Nominal, Color
- Datetime, Time, Position (X-axis)
- 4-min rolling average, Quantitative, Position (y-axis)

1.2 Sketch a visualization with an alternative encoding for one of the charts above. Compare your solution to the original. (10 points)

You can hand sketch or create the solution digitally. Please upload an image or screenshot. Your data doesn't need to match perfectly. Reflect on the differences in terms of perception/cognition and design principles as appropriate.



Encodings

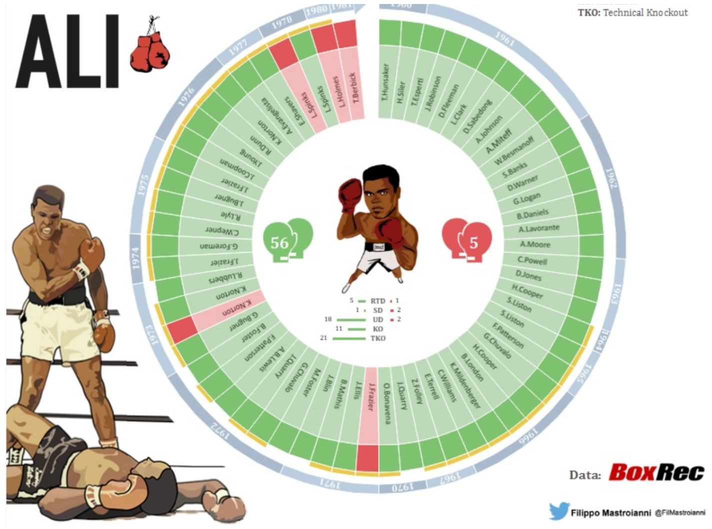
- Emoji,Ordinal,Position (Y)
- Percent, Quantitative, Size (circle) and Position (X)

My visualization requires greater cognitive processing to read compared to the visualization of "Five-Thirty-Eight." I believe this because the goal of the visualization is to show which emojis were most highly used during the match. Because in my visualization, the emojis are not ordered by rank one must scan the entire image and make several comparisons before finding the information they need. For example, to find the third most used emoji, I would have to scan the y-axis and either compare the circle sizes to one another (which humans are not good at) or look for the third rightmost circle on the X-axis (which nevertheless, takes some time to do).

Additionally, my visualization has redundant information. When it comes to displaying the percents per emoji, I encode this information twice: in position along the X-axis and with the size of the circle. Not only is it not necessary to have two encodings in this situation, doing so adds clutter to the visualization and makes it harder to read. There is also a lot of whitespace in my visualization. While that white space does highlight the difference between the laughing emoji and all other emojis, it is hard to compare across the other emojis. The bar chart representation has less white space since the bars are filled; this makes it easier for the eye to make comparisons.

1.3 Use one of the design principles reviewed in class (Tufte's data-ink ratio, graphical integrity, chart junk, etc.) to critique the following visualization (10 points)

Describe pros and cons of the visualization relative to these principles. If the image violates the principle, reflect on why this is might be ok or not.



http://vizzingdata.blogspot.com/2017/01/muhammad-ali-career-dataviz.html

I am going to critique the visualization on its **chart junk**. In my opinion, this visualization violates the principle of containing minimal chart junk as there are a lot of visual elements in the chart that are not necessary to comprehend the information represented on the graph.

Cons:

- This information visualization contains lots of cartoons, which don't convey much information. The cartoons are also distracting and take up valuable space that the author could use to enhance or enlarge other visualization elements to make it easier for the viewer to read.

Pros:

- I appreciate that the chart junk is on theme, making the visualization more attention-grabbing. Perhaps the author could remove most of the chart junk and keep only a few elements that grab the viewers' attention. Mainly, I think the author can remove the boxing character inside of the circle visualization.

Part 2. Altair programming exercise (70 points)

We have provided you with some code and parts of the article The Mayweather-McGregor Fight, As Told Through Emojis. This article is based on the dataset:

1. tweets Created by FiveThirtyEight with the Twitter Streaming API containing a sample of all the tweets that matched the search terms: #MayMac, #MayweatherMcGregor, #MayweatherVMcGregor, #MayweatherVsMcGregor, #McGregor and #Mayweather collected between 12:05 a.m. and 1:15 a.m. EDT, 12,118 that had emojis. Available on github

To earn points for this assignment, you must:

- Recreate the visualizations in the article (replace the images in the article with a code cell that creates a visualization). There are four visualizations to make. For the 2nd, 3rd, and 4th, we provide some example code to get the data into the right structure. The points for each visualization are distributed: (30 points: 9 (1st problem) + 7 (each of 2nd, 3rd & 4th)).
  - *Partial credit can be granted for each visualization (up to 5 points) if you provide the grammar of graphics description of the visualization without a fully implemented Altair solution*
- Propose one alternative visualization for one of the 4 article visualizations. Add a short paragraph describing why your visualization is better in terms of Design / Variable types encoding. (20 points/ 15 points plot + 5 justification)
- Propose a new visualization to complement a part of the article. Add a short paragraph justifying your decisions in terms of Design / Variable types encoding. (20 points/ 15 points plot + 5 justification)

Before you begin

IMPORTANT BROWSER ISSUE: For some non-ES6 Browsers there are problems with date/time conversions (see this). If things aren't working try something like Chrome for this assignment.

IMPORTANT DATA ISSUE: There are some differences in the data that 538 used and what we have. This will cause some issues to how things are normalized. Things should look very similar, but you may find, for example, that the scale of tweets when you normalize is different than the original figure. This is fine.

IMPORTANT STYLING/ANNOTATION NOTE: Part of this assignment is to get styling and annotation to be as close to 538 as possible. Altair and Vega-Lite aren't super helpful for annotation purposes. You will find that you need to do things like layering text and the arrows (hint: see here). What I usually do in practice (when I need the visualization to look good and have annotation) is get things as close to how I want them to look, export the image as an SVG and then load it into Figma, InkScape, or Adobe Illustrator. You can see "reasonably close approximations" in the examples we provide. Those have been generated using nothing but Altair.

In [1]:
```
# start with the setup
import pandas as pd
import altair as alt
import numpy as np
```

In [2]:
```
# enable correct rendering
alt.renderers.enable('default')

# enable fivethirtyeight theme
# alt.themes.enable('fivethirtyeight')
```

Out[2]:
```
RendererRegistry.enable('default')
```

In [3]:
```
# uses intermediate json files to speed things up
alt.data_transformers.enable('json')
```

Out[3]:
```
DataTransformerRegistry.enable('json')
```

In [4]:
```
# we're going to do some setup here in anticipation of needing the data in
# a specific format. We moved it all up here so everything is in one place.

# load the tweets
tweets = pd.read_csv('assets/tweets.csv')

# we're going to process the data in a couple of ways
# first, we want to know how many emojis are in each tweet so we'll create a new column
# that counts them
tweets['emojis'] = tweets['text'].str.findall(r'[^\w\s.,"@\'?/#!$%\^&\*;:{}=\-_`~()\U0001F1E6-\U0001F1FF]').str.len()

# next, there are a few specific emojis that we care about, we're going to create
# a column for each one and indicate how many times it showed up in the tweet
boxer_emojis = ['☘️','🇮🇪','🍀','💸','🤑','💰','💵','😴','😂','🤣','🥊','👊','👏','🇮🇪','💪','🔥','😭','💰']
for emoji in boxer_emojis:
    # here's a different way to get the counts
    tweets[emoji] = tweets.text.str.count(emoji)

# For the irish pride vs the money team we want the numer
# of either ☘️, 🇮🇪 or 🍀 and 💸, 🤑, 💰 or 💵 for each
tweets['irish_pride'] = tweets['☘️'] + tweets['🇮🇪'] + tweets['🍀']
tweets['money_team'] = tweets['💸'] + tweets['🤑'] + tweets['💰'] +  tweets['💵']
```

In [5]:
```
# uncomment to see what's inside
tweets.head()
```

Out[5]:

Out[5]:

	created_at	emojis	id	link	retweeted	screen_name	text	☘	🇮🇪	🍀	...	😂
0	2017-08-27 00:05:34	1	901656910939770881	https://twitter.com/statuses/901656910939770881	False	aaLiysr	Ringe çıkmadan ateş etmeye başladı 😀 #McGregor ...	0	0	0	...	0
1	2017-08-27 00:05:35	5	901656917281574912	https://twitter.com/statuses/901656917281574912	False	zulmafrancozaf	😂😂😂😂😂 @lalylourbet2 https://t.co/ERUGHhQINE	0	0	0	...	0
2	2017-08-27 00:05:35	2	901656917105369088	https://twitter.com/statuses/901656917105369088	False	Adriana11D	🇮🇪🇮🇪💪💪 #MayweathervMcgregor	0	3	0	...	0
3	2017-08-27 00:05:35	2	901656917747142657	https://twitter.com/statuses/901656917747142657	False	Nathan_Caro_	Cest partit #MayweatherMcGregor 💪🏿	0	0	0	...	0
4	2017-08-27 00:05:35	2	901656916828594177	https://twitter.com/statuses/901656916828594177	False	sahouraxox	Low key feeling bad for ppl who payed to watch...	0	0	0	...	0

5 rows × 25 columns

The Mayweather-McGregor Fight, As Told Through Emojis

We laughed, cried and cried some more.

Original article available at FiveThirtyEight

By Dhrumil Mehta, Oliver Roeder and Rachael Dottle

Filed under Mayweather vs. McGregor

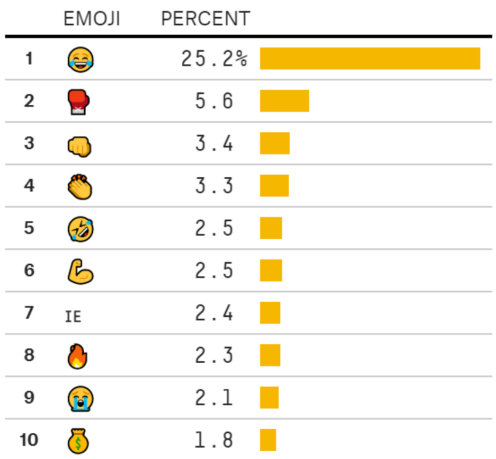
Get the data on GitHub

For the nearly 15,000 people in Las Vegas's T-Mobile Arena on Saturday night, and the millions more huddled around TVs across the world, the Floyd Mayweather–Conor McGregor fight was a roller coaster of emotions. They were anxious as pay-per-view technical problems pushed back the fight's start. They were full of anticipation when the combatants finally emerged after months of hype. They were surprised when McGregor held his own, or seemed to hold his own, for a couple of rounds. They were thrilled when Mayweather finally started fighting. And they were exhausted by the end.

How do we know all this? Emojis.

We were monitoring Twitter on fight night, pulling tweets that contained fight-related hashtags — those that included #MayweatherVsMcgregor, for example. In the end, we collected about 200,000 fight-related tweets, of which more than 12,000 contained emojis. (To be clear, that's a small enough sample that this emojinalysis might not make it through peer review.)[1]

> 1. We used the Twitter Streaming API which provides a sample of all the tweets that matched our search terms: #MayMac, #MayweatherMcGregor, #MayweatherVMcGregor, #MayweatherVsMcGregor, #McGregor and #Mayweather. Of the 197,989 tweets we collected between 12:05 a.m. and 1:15 a.m. EDT, 12,118 had emojis.



The most-used emoji in over 240,000 tweets collected during the Mayweather-McGregor fight broadcast, from 12:05 a.m. to 1:30 a.m. EDT on Aug. 27.

** Homework note, construct your solution to this chart in the cell below. Click here to see a sample output from Altair.

In [6]:
```
# We'll help you out with a table that has the percentages for each emoji

# dictionary that will map emoji to percentage
percentages = {}

# find total emojies
total = tweets['emojis'].sum()

# for each emoji, figure out how prevalent it is
emojis = ['😂','😭','😍','👊','😘','🇮🇪','💪','🔥','😎','💰']
for emoji in emojis:
    percentages[emoji] = [round(tweets[emoji].sum() / total * 100,1)]

# create a data frame to hold this from the dictionary
percentages_df = pd.DataFrame.from_dict(percentages).T

# sort the dictionary
percentages_df = percentages_df.sort_values(by=[0], ascending = False).reset_index()

# rename the columns
percentages_df = percentages_df.rename(columns={'index':'EMOJI', 0: 'PERCENT'})

# create a rank column based on position in the ordered list
percentages_df['rank'] = pd.Index(list(range(1,11)))

# modify the text
percentages_df['PERCENT_TEXT'] = percentages_df['PERCENT'].astype('str') + ' %'
```

In [7]:
```
# uncomment to see what's inside
percentages_df
```

Out[7]:

	EMOJI	PERCENT	rank	PERCENT_TEXT
0	😂	23.1	1	23.1 %
1	😭	5.7	2	5.7 %

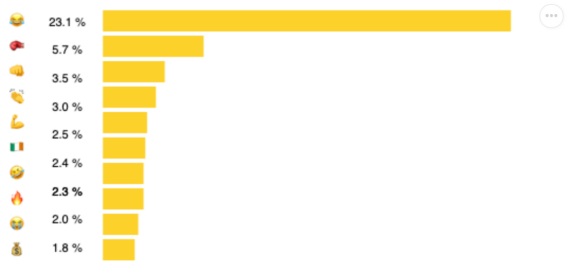
2	👊	3.5	3	3.5 %
3	👏	3.0	4	3.0 %
4	💪	2.5	5	2.5 %
5	🇮🇪	2.4	6	2.4 %
6	😏	2.3	7	2.3 %
7	🔥	2.3	8	2.3 %
8	😻	2.0	9	2.0 %
9	💰	1.8	10	1.8 %

** Homework note, construct your solution to this chart in the cell below. Click here to see a sample output from Altair.

2.1 use percentages_df to recreate the visualization above

In [8]:
```
# ---- bars -----
bars = alt.Chart(percentages_df).mark_bar(size=25,color='#FCD12A').encode(
    # encode x as the percent, and hide the axis
    x=alt.X(
        'PERCENT:Q',
        axis=None),
    y=alt.Y(
        # encode y using the name, use the movie name to label the axis, sort
        'EMOJI:O',
         axis=None,
         # we give the sorting order to avoid alphabetical order
         sort=list(percentages_df['EMOJI'])
    )
).properties(
    width=500,
    height=300)

# ---- % text ----
percents = alt.Chart(percentages_df).mark_text(fontSize=14).encode(
    y=alt.Y(
        'PERCENT:O',
        axis=None,
        sort=list(percentages_df['EMOJI'])
    ),
    text=alt.Text(
        'PERCENT_TEXT:O'
    )
).properties(
    width=40,
    height=300)

# ---- emojis ----
emojis = alt.Chart(percentages_df).mark_text(fontSize=14).encode(
    y=alt.Y(
        'EMOJI:O',
        axis=None,
        sort=list(percentages_df['EMOJI'])
    ),
    text=alt.Text(
        'EMOJI:O'
    )
).properties(
    width=40,
    height=300,
    )

# ---- layer the charts ----
emoji_chart = (emojis | percents | bars).configure_axis(
    grid=False
).configure_view(
    strokeWidth=0
)

emoji_chart
```

Out[8]:

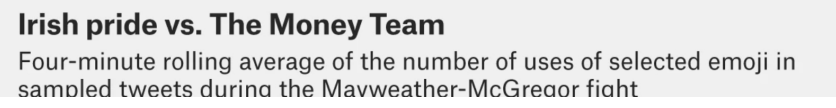


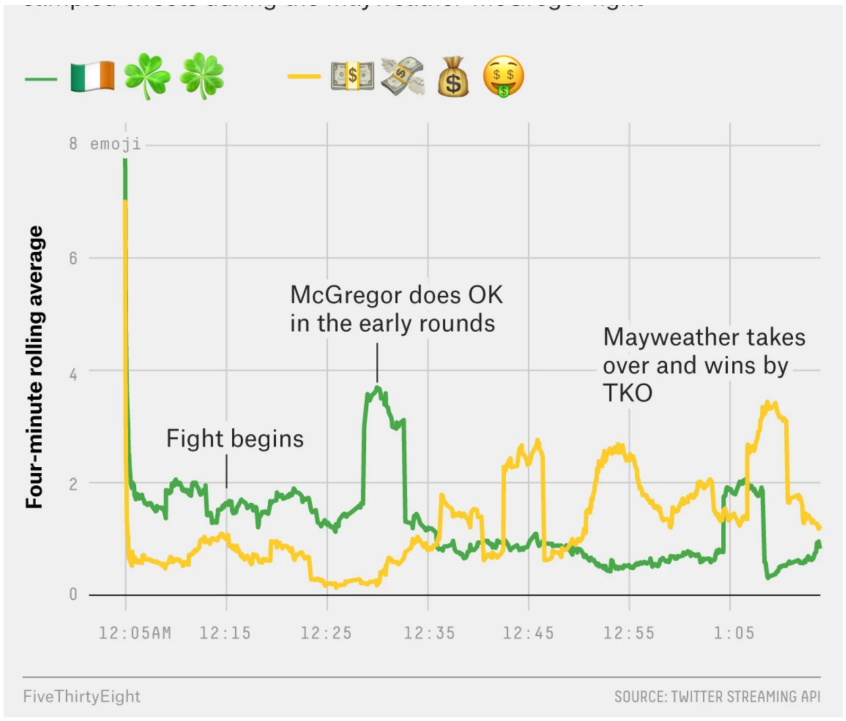
There were the likely frontrunners for most-used emoji: the 🇮🇪, the 👊, the 💪. But the emoji of the fight was far and away the 😂. ("Face with tears of joy.")[2]

> 1.2. That's certainly appropriate for this spectacle, but it should be noted that 😂 is also the most tweeted emoji generally.

Here's how the night unfolded, emoji-wise. (All of the charts below show them on a four-minute rolling average.)

For one thing, the fight was a sharply partisan affair. The majority of people in the arena appeared to be McGregor fans — he hails from Dublin and an Irish flag, worn cape-style, almost seemed like the evening's dress code. But other fans were members of TMT — The Money Team — and loyal to "Money" Mayweather. Twitter's loyalties came and went as the match progressed, with enthusiasm from either camp seemingly matching each fighter's success.





In [9]:
```
# Again, we're going to help you set up the data

# We're going to want to work with time objects so we need to make a datetime
# column (basically transforming the text in "created at"). It duplicates
# the data but it will make things easier


tweets['datetime'] = pd.to_datetime(tweets['created_at'])
tweets = tweets.set_index('datetime')


teams = tweets.copy()
teams['irish_pride']
teams = teams.resample('1s').sum()
teams  = teams[(teams['🐶']>0) | (teams['😀']>0) | (teams['💰']>0) | (teams['💸']>0) | (teams['☘']>0) | (teams['🍀']>0)

# next we're going to creat a rolling average
# first for the money team
mdf = teams['money_team'].rolling('4Min').mean().reset_index()
mdf['team'] = '🐶😀💰💸'
mdf = mdf.rename(columns={'money_team':'tweet_count'})

# next for the irish team
idf = teams['irish_pride'].rolling('4Min').mean().reset_index()
idf['team'] = '☘🍀🇮🇪'
idf = idf.rename(columns={'irish_pride':'tweet_count'})

# now we'll combine our datasets
ndf = pd.concat([mdf,idf])
```

In [10]:
```
# uncomment to see what's inside
ndf.head(5)
```

Out[10]:

	datetime	tweet_count	team
0	2017-08-27 00:05:35	7.000000	🐶😀💰💸
1	2017-08-27 00:05:38	3.500000	🐶😀💰💸
2	2017-08-27 00:05:40	2.333333	🐶😀💰💸
3	2017-08-27 00:05:43	2.000000	🐶😀💰💸
4	2017-08-27 00:05:44	1.600000	🐶😀💰💸

In [11]:
```
# we're also going to create an annotations data frame to help you
annotations = [['2017-08-27 00:15:00',4, 'Fight begins'],
               ['2017-08-27 00:22:00',5, 'McGregor does OK \nin the early rounds'],
               ['2017-08-27 00:53:00',4, 'Mayweather takes \nover and wins by \nTKO']]
a_df = pd.DataFrame(annotations, columns=['date','count','note'])
```

In [12]:
```
# uncomment to see what's inside
a_df
```

Out[12]:

	date	count	note
0	2017-08-27 00:15:00	4	Fight begins
1	2017-08-27 00:22:00	5	McGregor does OK \nin the early rounds
2	2017-08-27 00:53:00	4	Mayweather takes \nover and wins by \nTKO

In [13]:
```
lines = pd.DataFrame({
    'x': ['2017-08-27 00:15:00', '2017-08-27 00:15:00','2017-08-27 00:22:00','2017-08-27 00:30:00'],
    'y': [3.75, 2,4.4,3.9],
    'class': ['A', 'A','B','B']
})
```

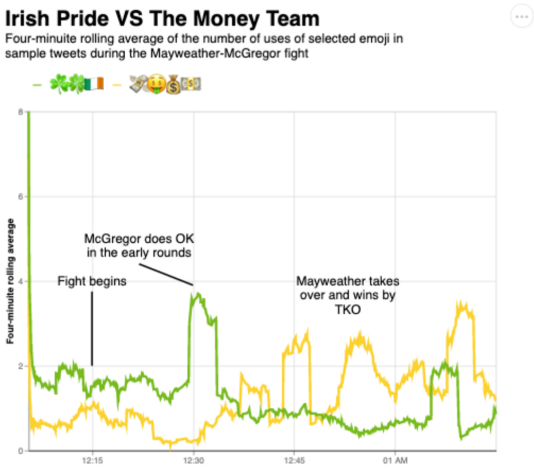
** Homework note, construct your solution to this chart in the cell below. Click here to see a sample output from Altair.

2.2 your turn, create your solution

In [14]:
```
# ---- '🐶😀💰💸' team ----
mayweather = alt.Chart(ndf).transform_filter(
    (alt.datum.team == '🐶😀💰💸')
).mark_line(strokeWidth=3).encode(
    x=alt.X(
        'datetime:T',
        axis=alt.Axis(
            title='',
            tickCount=4,
            domainColor='black'
        )
    ),
    y=alt.Y(
```

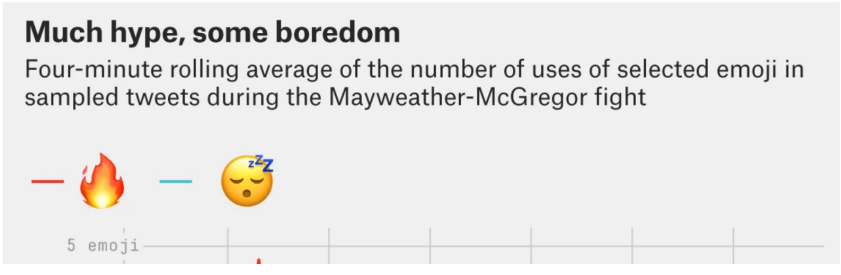
```
    y=alt.Y(
        'tweet_count:Q',
        axis=alt.Axis(
            title='Four-minuite rolling average',
            tickCount=5
        )
    ),
    color=alt.Color(
        'team:N',
        scale=alt.Scale(range=['#76BA1B','#FCD12A']),
        legend=alt.Legend(orient='top',title=' ')
    )
)

# ---- '☘️🇮🇪' team ----
mcgregor = alt.Chart(ndf).transform_filter(
    (alt.datum.team == '☘️🇮🇪')
).mark_line(strokeWidth=3).encode(
    x=alt.X('datetime:T'),
    y=alt.Y('tweet_count:Q'),
    color=alt.Color('team:N')
)

# ---- text annotation ----
annotation = alt.Chart(a_df).mark_text(
    align='center',
    fontSize=15,
    lineBreak='\n'
).encode(
    x=alt.X('date:T'),
    y=alt.Y('count:Q'),
    text=alt.Text('note')
)

# ---- line annotations ----
line_annotations = alt.Chart(lines).mark_line(color='black').encode(
    x=alt.X('x:T'),
    y=alt.Y('y:Q'),
    detail='class'
)

# ---- layering the figures and annotations ----
figure = (mayweather + mcgregor + annotation + line_annotations).properties(
   title=alt.TitleParams(
       text="Irish Pride VS The Money Team",
       subtitle=["Four-minuite rolling average of the number of uses of selected emoji in","sample tweets during the Ma
       fontSize=25,
       subtitleFontSize=15,
       anchor='start'
   ),
    width=550,
    height=400).configure_legend(labelFontSize=20
).configure_axis(labelColor='grey')

# ---- preview the figure ----
figure
```

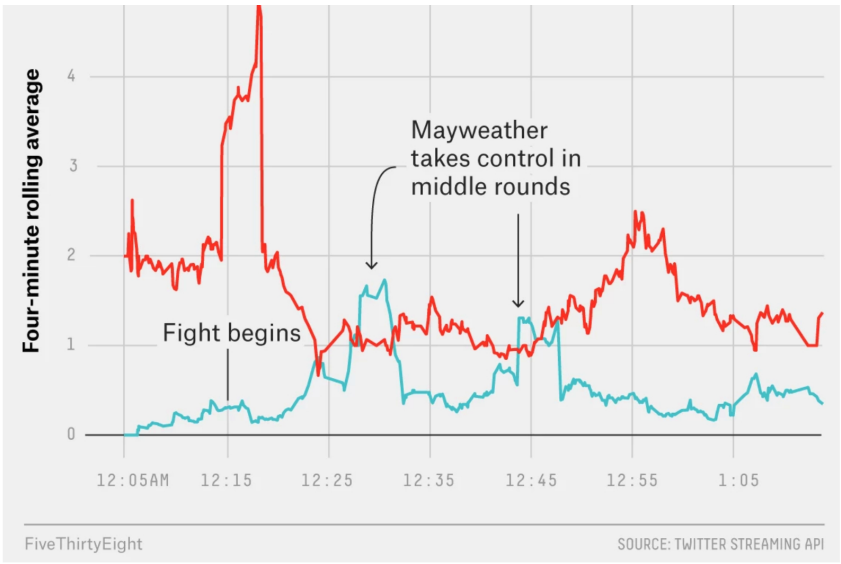
Out[14]:



To the surprise of many (of the neutral and pro-Mayweather viewers, anyway) McGregor won the first round. The next couple were washes, and a quarter of the way into the scheduled 12 rounds … the Irish underdog may have been winning! The Irish flags and shamrocks followed on Twitter. Things slowly (perhaps even 😴ly) turned around as one of the best pound-for-pound boxers in history took control of the man making his pro debut — an outcome which was predicted by precisely everyone. Out came the emoji money bags.

By the sixth round, it seemed like only a matter of time until the old pro dismantled the newcomer. By the ninth it was clear Mayweather was going for the knockout. It came soon thereafter. Mayweather unleashed a vicious flurry of punches in the 10th and the ref stepped in, declaring Mayweather the victor and saving McGregor, who was somehow still on his feet, from further damage.





** Homework note, construct your solution to this chart in the cell below. Click here to see a sample output from Altair.

2.3 your solution goes here, use the example above for the sampling and annotation

In [15]:
```
tweets['datetime'] = pd.to_datetime(tweets['created_at'])
tweets = tweets.set_index('datetime')

bored_hype = tweets.copy()[['created_at','🔥','😴']]
bored_hype = bored_hype.resample('1s').sum()
bored_hype  = bored_hype[(bored_hype['😴']>0) | (bored_hype['🔥']>0)]

# next we're going to creat a rolling average
# first for the '🔥' emoji
hype_df = bored_hype['🔥'].rolling('4Min').mean().reset_index()
hype_df['emoji'] = '🔥'
hype_df = hype_df.rename(columns={'🔥':'tweet_count'})

# next for the '😴' emoji
bored_df = bored_hype['😴'].rolling('4Min').mean().reset_index()
bored_df['emoji'] = '😴'
bored_df = bored_df.rename(columns={'😴':'tweet_count'})

# now we'll combine our datasets
ndf = pd.concat([hype_df,bored_df])

ndf.head(5)
```

Out[15]:

	datetime	tweet_count	emoji
0	2017-08-27 00:05:39	2.00	🔥
1	2017-08-27 00:05:52	2.00	🔥
2	2017-08-27 00:05:54	2.00	🔥
3	2017-08-27 00:05:59	2.25	🔥
4	2017-08-27 00:06:03	2.00	🔥

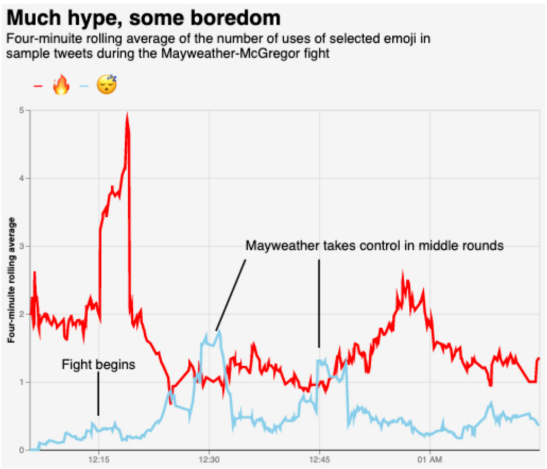
In [16]:
```
# we're also going to create an annotations data frame to help you
annotations = [['2017-08-27 00:10:00',1.25, 'Fight begins'],
               ['2017-08-27 00:35:00',3, 'Mayweather takes control in middle rounds']]
a_df = pd.DataFrame(annotations, columns=['date','count','note'])
```

In [17]:
```
lines = pd.DataFrame({
    'x': ['2017-08-27 00:15:00', '2017-08-27 00:15:00',
          '2017-08-27 00:31:00','2017-08-27 00:35:00',
          '2017-08-27 00:45:00','2017-08-27 00:45:00'
         ],
    'y': [1.15,0.5,
          1.75,2.8,
          2.8,1.5
         ],
    'class': ['A', 'A','B','B','C','C']
})
```

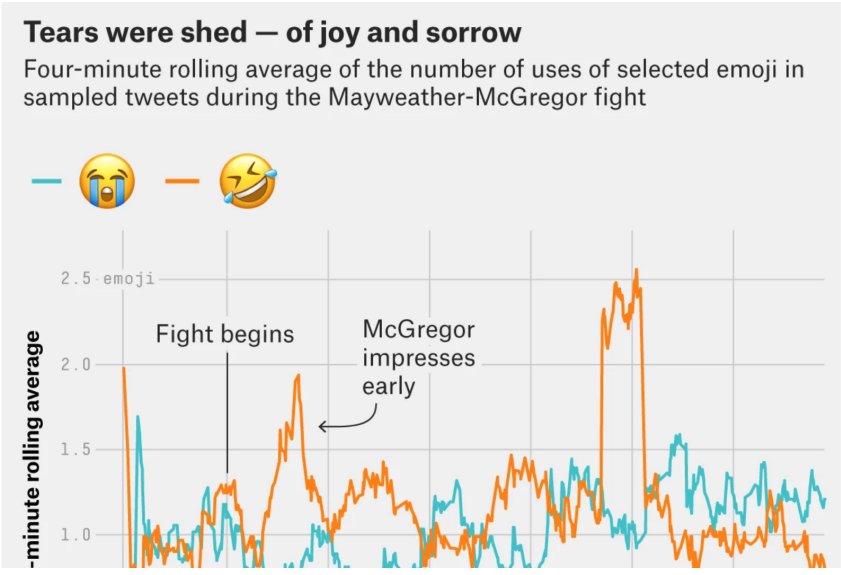
In [18]:
```
# ---- 🔥 line ----
hype = alt.Chart(ndf).transform_filter(
    (alt.datum.emoji == '🔥')
).mark_line(strokeWidth=3).encode(
    x=alt.X(
        'datetime:T',
        axis=alt.Axis(
            title='',
            tickCount=4,
            domainColor='black',
            tickColor='grey'
        )
    ),
    y=alt.Y(
        'tweet_count:Q',
        axis=alt.Axis(
            title='Four-minuite rolling average',
            tickCount=5
        )
    ),
    color=alt.Color(
        'emoji:N',
        scale=alt.Scale(range=['red','#87CEEB']),
        legend=alt.Legend(orient='top',title=' ')
    )
)

# ---- 😴 line ----
boredom = alt.Chart(ndf).transform_filter(
    (alt.datum.emoji == '😴')
).mark_line(strokeWidth=3).encode(
    x=alt.X(
        'datetime:T',
        axis=alt.Axis(
            title='',
            tickCount=4
        )
    ),
    y=alt.Y(
        'tweet_count:Q',
        axis=alt.Axis(tickCount=5)
    ),
```

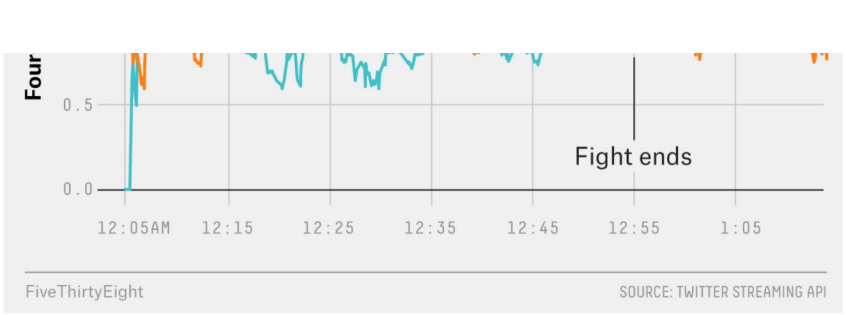
```
    color=alt.Color('emoji:N')
)

# ---- text annotation ----
annotation = alt.Chart(a_df).mark_text(
    align='left',
    fontSize=16,
    lineBreak='\n'
).encode(
    x=alt.X('date:T'),
    y=alt.Y('count:Q'),
    text=alt.Text('note')
)

# ---- lines ----
line_annotations = alt.Chart(lines).mark_line(color='black').encode(
    x=alt.X('x:T'),
    y=alt.Y('y:Q'),
    detail='class'
)

# ---- layering the chart ----
figure = (hype +boredom+annotation+line_annotations).properties(
   title=alt.TitleParams(
       text="Much hype, some boredom",
       subtitle=["Four-minuite rolling average of the number of uses of selected emoji in","sample tweets during the Ma
       fontSize=25,
       subtitleFontSize=16,
       anchor='start'
   ),
    width=600,
    height=400
).configure(
    background='#F5F5F5'
).configure_legend(labelFontSize=20
).configure_axis(labelColor='grey')

# ---- preview the figure ----
figure
```

Out[18]:



It ended just over 37 minutes after it began. Five seconds later, Mayweather leapt up on the corner ropes, victorious — 50-0. Some observers declared it a satisfying spectacle. Others, McGregor chief among them, were frustrated with the finish. The emoji users on Twitter appeared to think the fight was, for the most part, 🔥 — especially as it heated up toward the end. While the result may never have been in question, this was a welcome outcome for many who viewed Mayweather's last megafight against Manny Pacquiao as an epic 😴😴😴😴.





** Homework note, construct your solution to this chart in the cell below. Click here to see a sample output from Altair.

2.4 your solution goes here, use the example above for the sampling and annotation

In [19]:
```
tweets['datetime'] = pd.to_datetime(tweets['created_at'])
tweets = tweets.set_index('datetime')

joy_sorrow = tweets.copy()[['created_at','😂','😢']]
joy_sorrow = joy_sorrow.resample('1s').sum()
joy_sorrow  = joy_sorrow[(joy_sorrow['😂']>0) | (joy_sorrow['😢']>0)]

# next we're going to creat a rolling average
# first for '😢' emoji
joy_df = joy_sorrow['😢'].rolling('4Min').mean().reset_index()
joy_df['emoji'] = '😢'
joy_df = joy_df.rename(columns={'😢':'tweet_count'})

# next for the '😭' emoji
sorrow_df = joy_sorrow['😭'].rolling('4Min').mean().reset_index()
sorrow_df['emoji'] = '😭'
sorrow_df = sorrow_df.rename(columns={'😭':'tweet_count'})

# now we'll combine our datasets
ndf = pd.concat([joy_df,sorrow_df])

ndf.sample(5)
```

Out[19]:

	datetime	tweet_count	emoji
556	2017-08-27 01:11:10	0.939394	😢
503	2017-08-27 01:05:14	0.931818	😢
555	2017-08-27 01:11:02	1.242424	😭
203	2017-08-27 00:30:37	1.366667	😢
200	2017-08-27 00:30:14	0.625000	😭

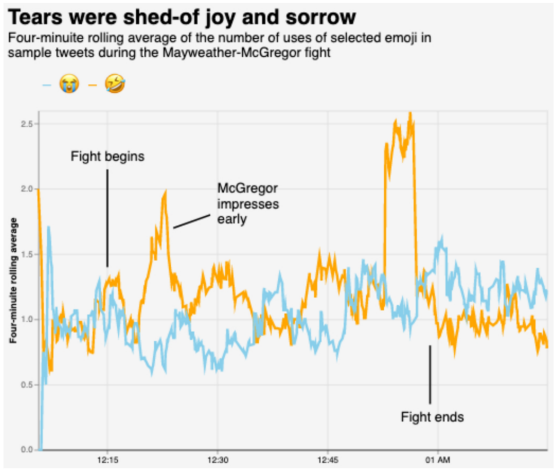
In [20]:
```
annotations = [['2017-08-27 00:10:00',2.25, 'Fight begins'],
               ['2017-08-27 00:30:00',2, 'McGregor \nimpresses \nearly'],
               ['2017-08-27 00:55:00',0.25, 'Fight ends']
              ]
a_df = pd.DataFrame(annotations, columns=['date','count','note'])
```

In [21]:
```
lines = pd.DataFrame({
    'x': ['2017-08-27 00:15:00', '2017-08-27 00:15:00',
          '2017-08-27 00:24:00','2017-08-27 00:29:00',
          '2017-08-27 00:59:00','2017-08-27 00:59:00'
         ],
    'y': [2.15,1.4,
          1.7,1.8,
          0.35,0.8
         ],
    'class': ['A', 'A','B','B','C','C']
})
```

In [22]:
```
# ---- '😢' line ----
joy = alt.Chart(ndf).transform_filter(
    (alt.datum.emoji == '😢')
).mark_line(strokeWidth=3).encode(
    x=alt.X(
        'datetime:T',
        axis=alt.Axis(
            title='',
            tickCount=4,
            domainColor='black'
        )
    ),
    y=alt.Y(
        'tweet_count:Q',
        axis=alt.Axis(
            title='Four-minuite rolling average',
            tickCount=5
        )
    ),
    color=alt.Color(
        'emoji:N',
        scale=alt.Scale(range=['#87CEEB','orange']),
        legend=alt.Legend(orient='top',title=' ')
    )
)

# ---- '😭' line ----
sorrow = alt.Chart(ndf).transform_filter(
    (alt.datum.emoji == '😭')
).mark_line(strokeWidth=3).encode(
    x=alt.X(
        'datetime:T',
        axis=alt.Axis(
            title='',
            tickCount=4
        )
    ),
    y=alt.Y(
        'tweet_count:Q',
        axis=alt.Axis(tickCount=5)
    ),
    color=alt.Color('emoji:N')
)

# ---- text annotations ----
annotation = alt.Chart(a_df).mark_text(
    align='left',
    fontSize=16,
    lineBreak='\n'
).encode(
    x=alt.X('date:T'),
    y=alt.Y('count:Q'),
    text=alt.Text('note')
)

# ---- line annotations ----
line_annotations = alt.Chart(lines).mark_line(color='black').encode(
    x=alt.X('x:T'),
    y=alt.Y('y:Q'),
    detail='class'
)

# ---- layering figures and annotations ----
```

```
figure = (joy + sorrow + annotation + line_annotations).properties(
   title=alt.TitleParams(
       text="Tears were shed-of joy and sorrow",
       subtitle=["Four-minuite rolling average of the number of uses of selected emoji in","sample tweets during the Ma
       fontSize=25,
       subtitleFontSize=16,
       anchor='start'
   ),
    width=600,
    height=400
).configure(
    background='#F5F5F5'
).configure_legend(labelFontSize=20
).configure_axisLeft(labelColor='grey')

# ---- preview figure ----
figure
```

Out[22]:



They laughed. They cried. And they laughed some more. And they cried some more.

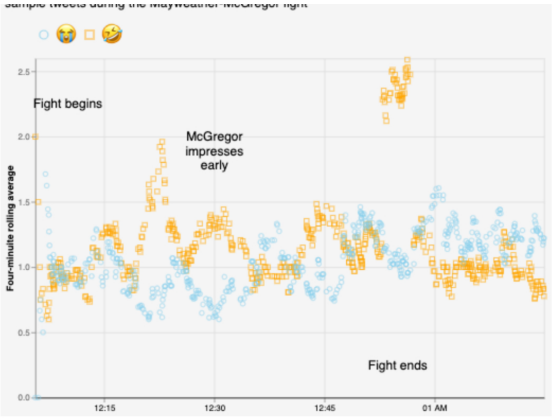
2.5 Make your own (part 1-alternative)

Propose one *alternative* visualization for one of the article's visualizations. Add a short paragraph describing why your visualization is more *effective* based on principles of perception/cognition. If you feel your visualization is worse, that's ok! Just tell us why. (20 points/ 15 points plot + 5 justification)

In [23]:

```
# ---- '😹' line ----
joy = alt.Chart(ndf).transform_filter(
    (alt.datum.emoji == '😹')
).mark_point(strokeWidth=1).encode(
    x=alt.X(
        'datetime:T',
        axis=alt.Axis(
            title='',
            tickCount=4,
            domainColor='black'
        )
    ),
    y=alt.Y(
        'tweet_count:Q',
        axis=alt.Axis(
            title='Four-minuite rolling average',
            tickCount=5
        )
    ),
    color=alt.Color(
        'emoji:N',
        scale=alt.Scale(range=['#87CEEB','orange']),
        legend=alt.Legend(orient='top',title=' ')
    ),
    shape='emoji:N'
)

# ---- '😭' line ----
sorrow = alt.Chart(ndf).transform_filter(
    (alt.datum.emoji == '😭')
).mark_point(strokeWidth=1).encode(
    x=alt.X(
        'datetime:T',
        axis=alt.Axis(
            title='',
            tickCount=4
        )
    ),
    y=alt.Y(
        'tweet_count:Q',
        axis=alt.Axis(tickCount=5)
    ),
    color=alt.Color('emoji:N'),
    shape='emoji:N'
)

# ---- text annotation ----
annotation = alt.Chart(a_df).mark_text(
    align='center',
    fontSize=15,
    lineBreak='\n'
).encode(
    x=alt.X('date:T'),
    y=alt.Y('count:Q'),
    text=alt.Text('note')
)

# ---- layering the chart ----
figure = (joy + sorrow + annotation).properties(
   title=alt.TitleParams(
       text="Tears were shed-of joy and sorrow",
       subtitle=["Four-minuite rolling average of the number of uses of selected emoji in","sample tweets during the Ma
       fontSize=25,
       subtitleFontSize=15,
       anchor='start'
   ),
    width=600,
    height=400
).configure(
    background='#F5F5F5'
).configure_legend(labelFontSize=20
).configure_axisLeft(labelColor='grey')

# ---- display figure ----
figure
```

Out[23]:



I feel that my visualization is less effective based on the principles of perception and cognition. To make this visualization, I changed the line encoding to be a point encoding that varies depending on the emoji. A circle represents the sorrow emoji, and a rectangle represents the joy emoji. Having different shaped points is less perceptually effective than a line because our goal is to display change across time. Because time is continuous, a line demonstrates that the space between points is meaningful while the points do not.

Additionally, my visualization appears more cluttered as the shapes take up more chart space than a line.

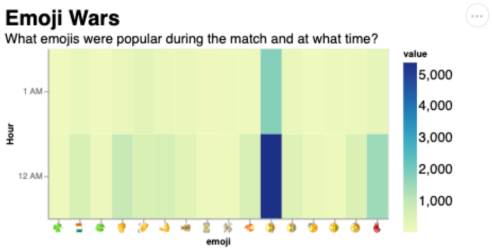
2.6 Make your own (part 2-novel)

Propose a *new* visualization to complement a part of the article. Add a short paragraph justifying your decisions in terms of Perception/Cognition processes. If you feel your visualization is worse, that's ok! Just tell us why. (20 points/ 15 points plot + 5 justification)

In [24]:
```
# keep only emoji and datetime columns, reset index so datetime is a column
new_df=tweets[['😭', '🇮🇪', '🍀', '😢', '😩', '👌', '😹', '😂', '😍', '😘', '👊',
       '👏', '💪', '🔥', '😈']]
new_df = new_df.reset_index()


# group by hour
new_df = new_df.resample('H',on='datetime').sum().reset_index()

# put in long format
new_df = new_df.melt(id_vars='datetime')
new_df = new_df.groupby(['datetime','variable']).sum().reset_index()

new_df['datetime'] = new_df['datetime'].astype(str).apply(lambda x: x.split(' ')[-1])
new_df['hour'] = new_df['datetime'].apply(lambda x: '12 AM' if x == '00:00:00' else '1 AM')
```

In [25]:
```
alt.Chart(new_df).mark_rect().encode(
    x=alt.X('variable:N',title='emoji'),
    y=alt.Y('hour:O',title='Hour'),
    color=alt.Color('value:Q')
).properties(
   title=alt.TitleParams(
       text="Emoji Wars",
       subtitle='What emojis were popular during the match and at what time?',
       fontSize=25,
       subtitleFontSize=16,
       anchor='start'
   ),
    width=400,
    height=200
).configure_legend(labelFontSize=16
).configure_axisLeft(labelColor='grey')
```

Out[25]:



The article was focused on how emoji use varied throughout the match but didn't give us much insight into the magnitude of the number of emojis used. I used a heatmap to show how many emojis of each kind were used within each hour of the game.

I think that this visualization is perceptually/cognitively sound as there is no chart junk, and the colored bins are easy to read. Not only is it easy to perceive the difference in the use of one emoji relative to any other, but the legend provides actual numbers of emojis used, which adds value to our understanding of how big of a cultural phenomenon the event was.

In [ ]:
```
```