Information Visualization I

School of Information, University of Michigan

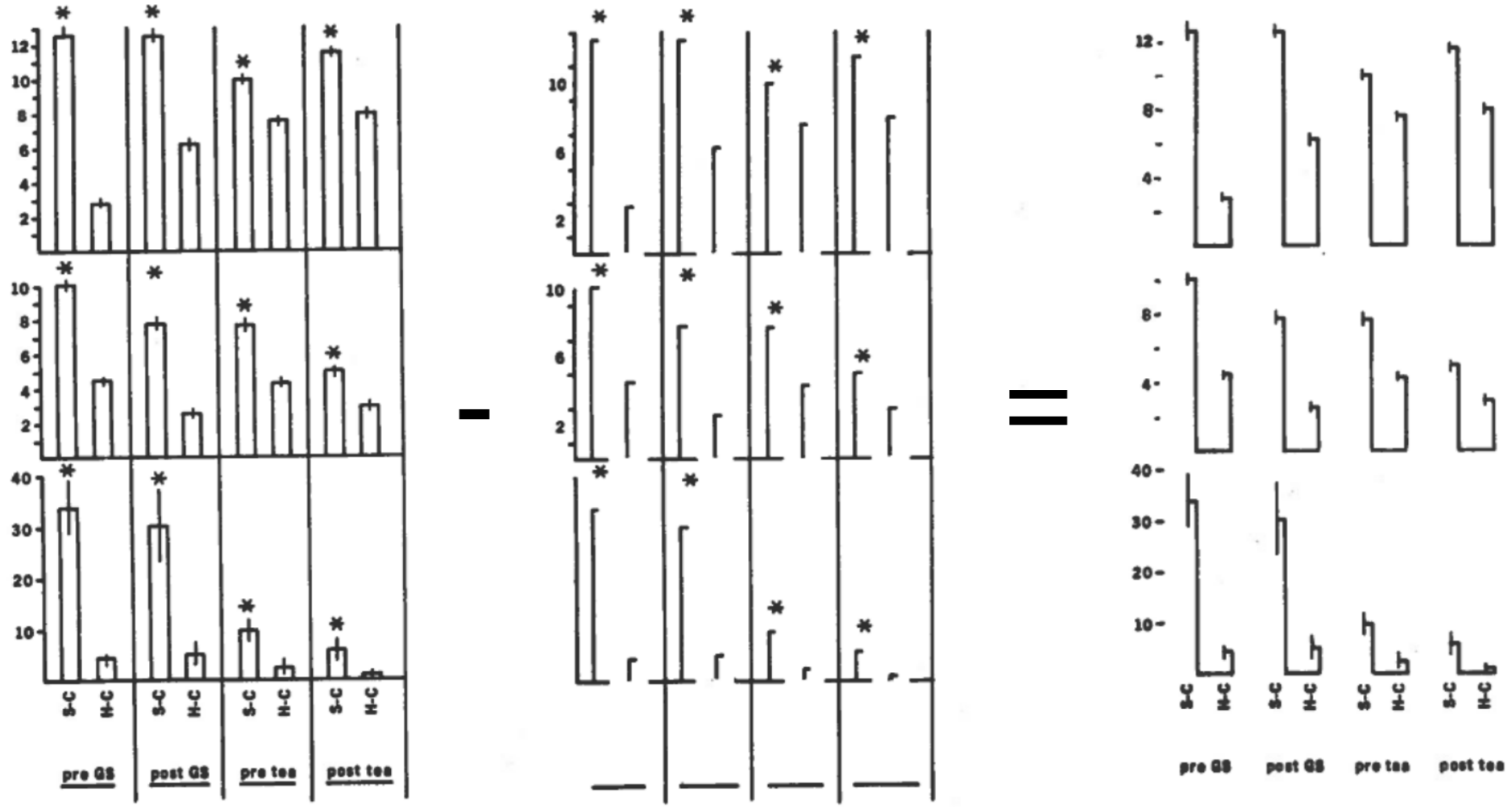
Week 4:

- 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 (https://altair-viz.github.io/)

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 (https://fivethirtyeight.com) available online (https://fivethirtyeight.com/features/the-mayweather-mcgregor-fight-as-told-through-emojis/) (Hickey, Koeze, Dottle, Wezerek 2017)
- Datasets from Five Thirty Eight, we have download a subset of these datasets to ./assets (assets) but the original can be found on Five Thirty Eight Mayweather vs McGregor (https://github.com/fivethirtyeight/data/tree/master/mayweather-mcgregor)

Important notes:

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

2) Grading for this assignment is entirely done by a human grader. They will be running tests on the functions we ask you to create. This means there is no autograding (submitting through the autograder will result in an error). You are expected to test and validate your own code.

3) Keep your notebooks clean and readable. If your code is highly messy or inefficient you will get a deduction.

4) Follow the instructions for submission on Coursera. You will be providing us a generated link to a read-only version of your notebook and a PDF. 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. If you're having trouble with printing, take a look at this video (https://youtu.be/PiO-K7AoWjk).

Part 1. Data Types & Design (30 points)

Read the article published in Five Thirty Eight The Mayweather-McGregor Fight, As Told Through Emojis (https://fivethirtyeight.com/features/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)

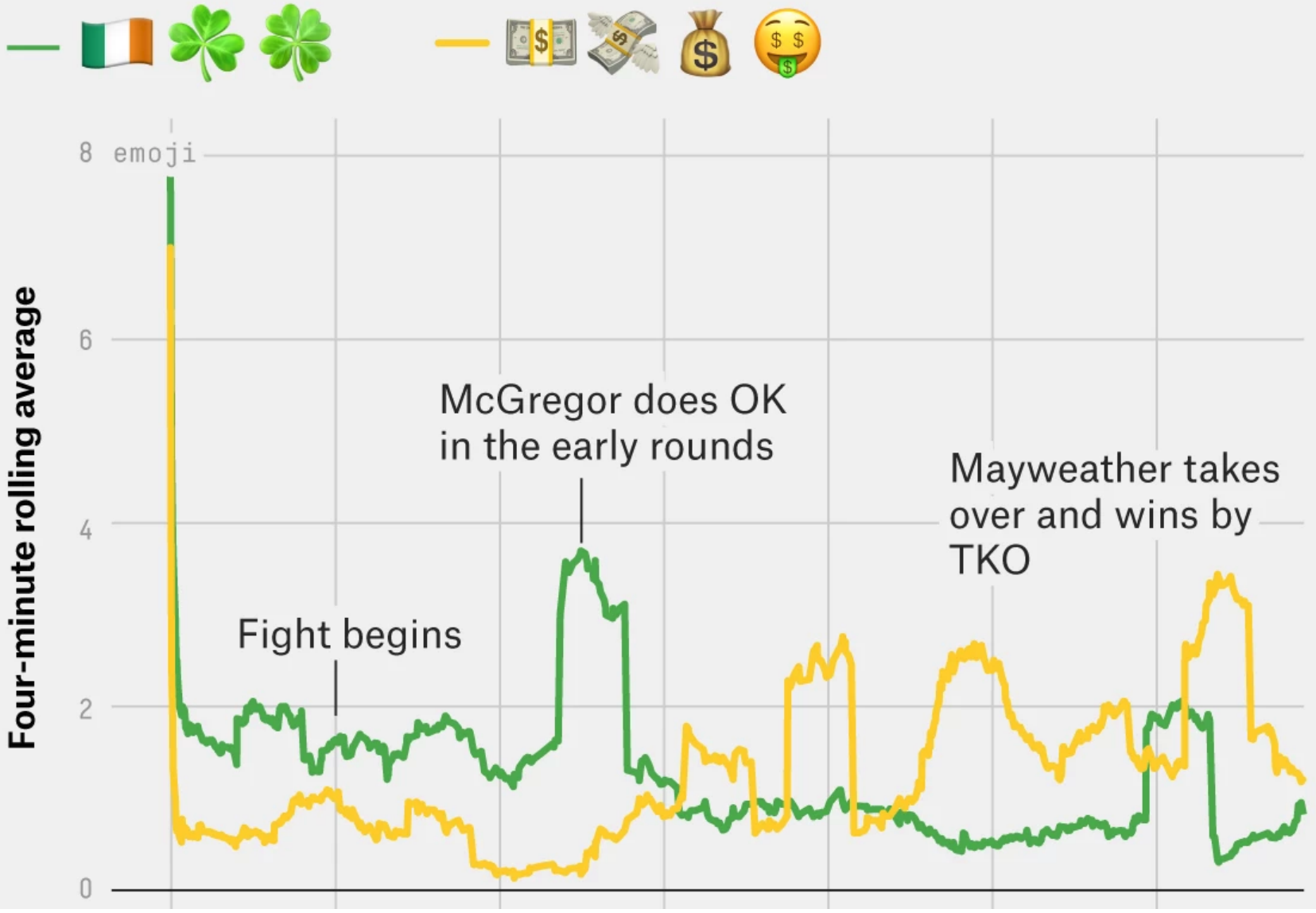
For each chart, describe the variable name, type, and encoding (e.g., weight, quantitative, bar length)

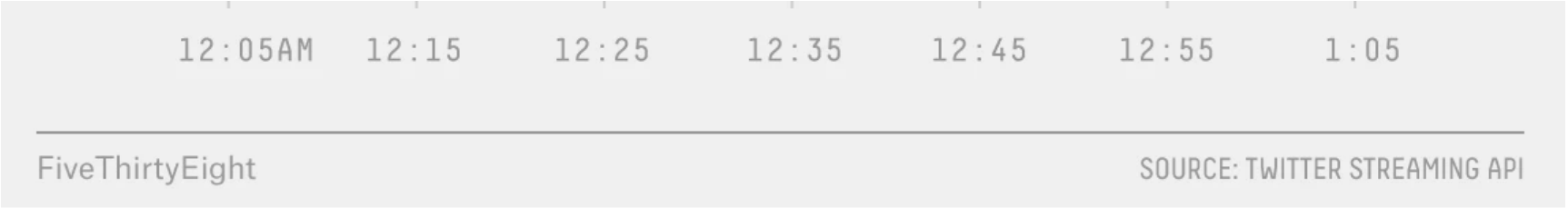
	EMOJI	PERCENT
1	😂	25.2%
2	🥊	5.6
3	👊	3.4
4	👏	3.3
5	🤣	2.5
6	💪	2.5
7	IE	2.4
8	🔥	2.3
9	😭	2.1
10	💰	1.8

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.

Irish pride vs. The Money Team

Four-minute rolling average of the number of uses of selected emoji in sampled tweets during the Mayweather-McGregor fight





1.1 Answer

A. Chart 1:

1. Variable: emoji ('EMOJI'), Type: nominal, Encoding: Y-axis
2. Variable: percentage of emoji usage ('PERCENT'), Type: quantitative, Encoding: length of bars (X-axis) and numbers next to bars

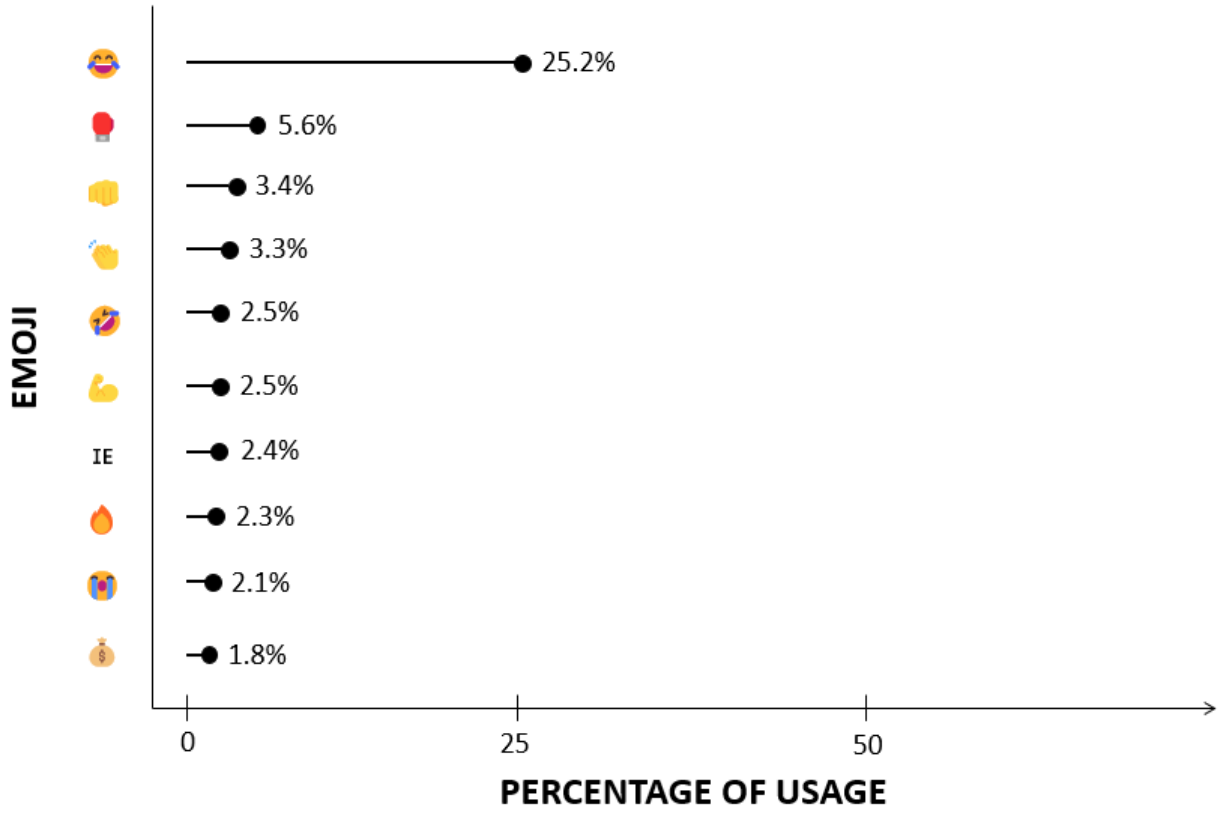
B. Chart 2:

1. Variable: Four-minute rolling average ('tweet_count' in Y-axis, 'count' in mark_text and mark_line encodings), Type: quantitative, Encoding: Y-axis, text, lines connecting annotations with chart
2. Variable: 'datetime' ('date' in mark_text and mark_line encodings), Type: temporal, Encoding: lines (X-axis, lines connecting annotations with chart), text
3. Variable: 'team', Type: nominal, Encoding: color of lines
4. Variable: annotations ('note'), Type: string (Python type), Encoding: annotation text

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.

1.2 Answer



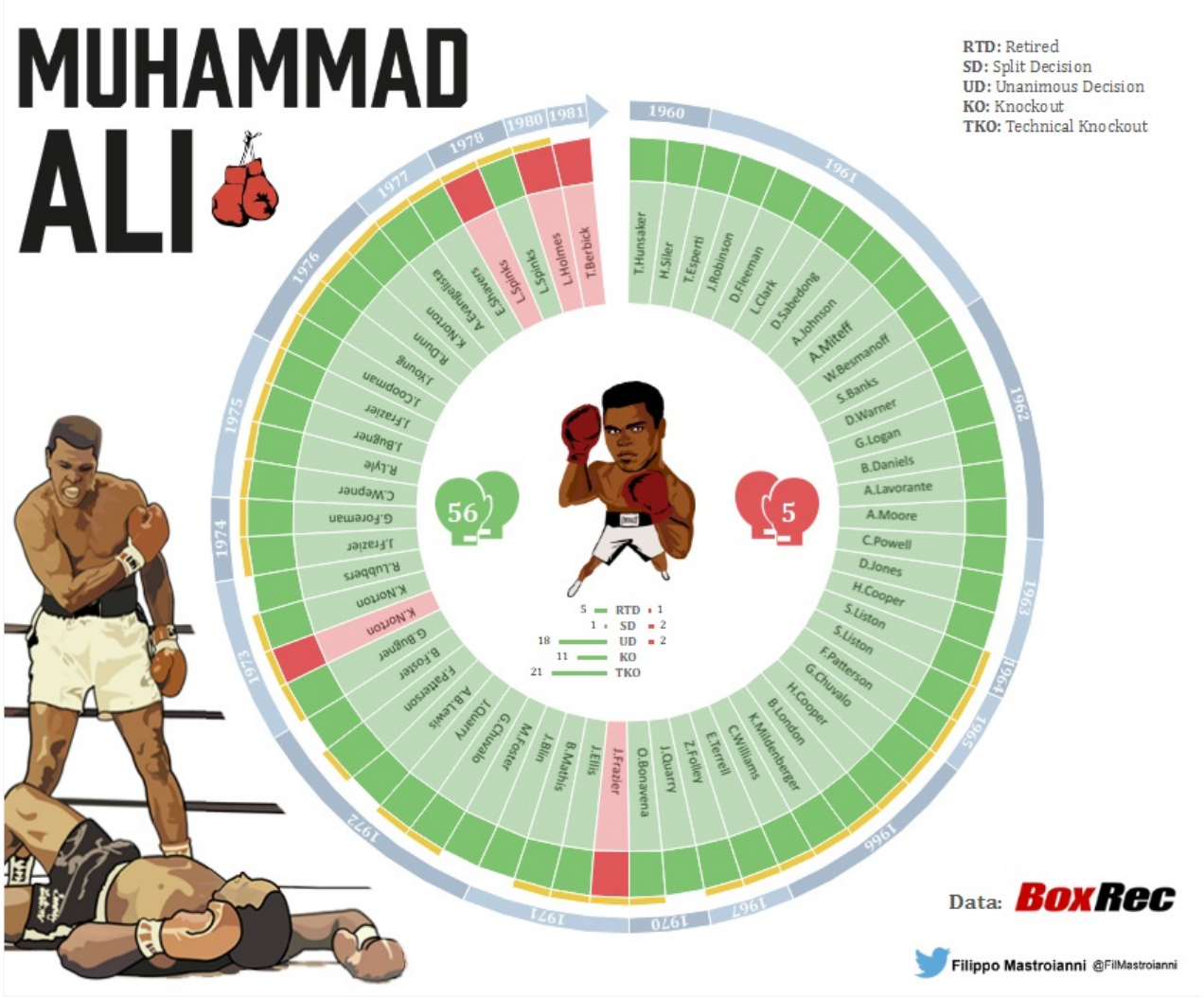
The alternative encoding I have proposed and created for the first chart above is a Stem Plot.

The Stem Plot is as expressive as the original bar chart as it conveys the percentage of emoji use in tweets as in the original in terms of length of stems as opposed to the length of a bars. Moreover, the Stem Plot maximizes the data-ink ratio by doing away with the bars This minimalistic design makes it more aesthetically pleasing, less cluttered and more elegant than the original chart.

However, in terms of effectiveness, the original bar chart is probably better to accurately visualize the minute differences in percentages of emoji usage.

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

1.3 Answer

Design Principle: Chart Junk

Pros: The presence of the Muhammad Ali graphics give the immediate impression that this chart is talking about and is sort of a testimonial to his career. The graphics frame the chart and also fill up the empty space within, which makes them quite eye-catching and the chart as a whole memorable. The nature of some of the graphics (i.e. Muhammad Ali standing victorious over his opponent on the ground) also unequivocally indicate that the chart provides emphasis on Muhammad Ali's wins (56 wins and only 5 losses in his whole career!), which would likely make any fan of his want to go over the chart in more detail.

Cons: The actual data lies in the Sunburst Diagram part of the chart. The graphics, while relevant to the subject matter of the data, tend to take up a lot of the chart space. This has the effect of dominating the visual and taking away focus from the data (especially for a person who isn't a fan or has never heard of Muhammad Ali before, discouraging them from scrutinizing the data), thereby diminishing the intent of the chart. Moreover, the graphic in the middle of the chart seems to add nothing to the narrative that the chart is trying to convey, making it a redundant and unnecessary part of the chart.

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 (https://fivethirtyeight.com/features/the-mayweather-mcgregor-fight-as-told-through-emojis/)](https://fivethirtyeight.com/features/the-mayweather-mcgregor-fight-as-told-through-emojis/). This article is based on the dataset:

1. [tweets (data/tweets.csv)] 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 (https://github.com/fivethirtyeight/data/tree/master/mayweather-mcgregor)]

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. (10 points/ 5 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. (30 points/ 20 points plot + 10 justification)

Before you begin

IMPORTANT BROWSER ISSUE: For some non-ES6 Browsers there are problems with date/time conversions (see [this (https://altair-viz.github.io/user_guide/times_and_dates.html)]). 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 (https://github.com/altair-viz/altair/issues/1721)]). 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 (https://www.figma.com/)], [InkScape (https://inkscape.org/)], or [Adobe Illustrator (https://www.adobe.com/products/illustrator.html)]. 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')
```

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.

def loadData(filename='assets/tweets.csv'):
    # load the tweets
    tweets = pd.read_csv(filename)

    # 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['💵']

    # create a datetime column in the right format so it's easier to use later
    tweets['datetime'] = pd.to_datetime(tweets['created_at'])
    tweets = tweets.set_index('datetime')

    return(tweets)

tweets = loadData()
```

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

Out[5]:

datetime	created_at	emojis	id	link	retweeted	screen_name	text	☘	🇮🇪	🍀	...	😂	🤣	🥊	👊	👏	💪	🔥	🇮🇪	irish
2017-08-27 00:05:34	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	0	0	0	0	0	0	0	0
2017-08-27 00:05:35	2017-08-27 00:05:35	5	901656917281574912	https://twitter.com/statuses/901656917281574912	False	zulmafrancozaf	😳😳😳😳😳 @lalylourbet2 https://t.co/ERUGHhQINE	0	0	0	...	0	0	0	0	0	0	0	0	0
2017-08-27 00:05:35	2017-08-27 00:05:35	2	901656917105369088	https://twitter.com/statuses/901656917105369088	False	Adriana11D	🇮🇪🇮🇪🇮🇪 💪💪 #MayweathervMcgregor	0	3	0	...	0	0	0	0	0	2	0	0	0
2017-08-27 00:05:35	2017-08-27 00:05:35	2	901656917747142657	https://twitter.com/statuses/901656917747142657	False	Nathan_Caro_	Cest partit #MayweatherMcGregor 💪🏿	0	0	0	...	0	0	0	0	0	1	0	0	0
2017-08-27 00:05:35	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	2	0	0	0	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 (https://fivethirtyeight.com/features/the-mayweather-mcgregor-fight-as-told-through-emojis/)](https://fivethirtyeight.com/features/the-mayweather-mcgregor-fight-as-told-through-emojis/)

By [Dhrumil Mehta (https://fivethirtyeight.com/contributors/dhrumil-mehta/)](https://fivethirtyeight.com/contributors/dhrumil-mehta/), [Oliver Roeder (https://fivethirtyeight.com/contributors/oliver-roeder/)](https://fivethirtyeight.com/contributors/oliver-roeder/) and [Rachael Dottle (https://fivethirtyeight.com/contributors/rachael-dottle/)](https://fivethirtyeight.com/contributors/rachael-dottle/)

Filed under [Mayweather vs. McGregor (https://fivethirtyeight.com/tag/mayweather-vs-mcgregor/)](https://fivethirtyeight.com/tag/mayweather-vs-mcgregor/)

Get the data on [GitHub (https://github.com/fivethirtyeight/data/tree/master/mayweather-mcgregor)](https://github.com/fivethirtyeight/data/tree/master/mayweather-mcgregor)

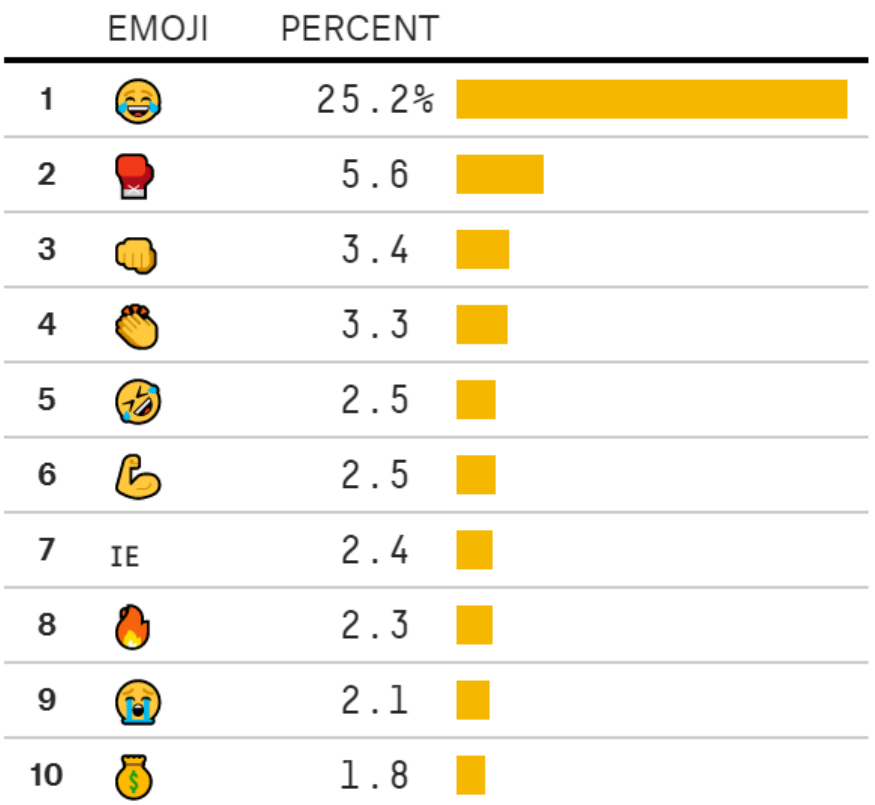
For the nearly 15,000 people in Las Vegas's T-Mobile Arena on Saturday night, and the millions more huddled around TVs across the world, the Floyd Mayweather–Conor McGregor fight was a roller coaster of emotions. They were anxious as pay-per-view [technical problems (http://www.espn.com/boxing/story/_/id/20469815/floyd-mayweather-conor-mcgregor-delay-ppv-problems)](http://www.espn.com/boxing/story/_/id/20469815/floyd-mayweather-conor-mcgregor-delay-ppv-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 (https://dev.twitter.com/streaming/overview) 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 (assets/altair_chart1.png) to see a sample output from Altair.

In [6]:

```python
# We'll help you out with a table that has the percentages for each emoji

def createPercentagesDF(tweets):
    # input: the tweets dataframe as formatted above
    # 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 = ['😂','🤣','🥊','👊','👏','ɪᴇ','💪','🔥','😭','💰']
    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') + ' %'
    return(percentages_df)

percentages_df = createPercentagesDF(tweets)
```

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	IE	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 %

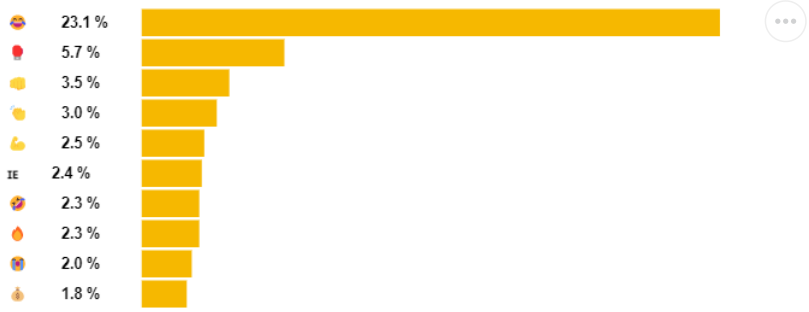
2.1 Replicate the vis (9 points)

Construct your solution to the chart above ("Emoji Percent") in the cell below. Click here (assets/emoji_distrib_altair.png) to see a sample output we created with Altair.

In [8]:
```
# use percentages_df to recreate the visualization above

def create_percentages_vis(indf):
    # input: indf (a frame formated like percentages_df)
    # return an Altair vis matching the example above
    # YOUR CODE HERE

    chart=alt.Chart(indf).transform_calculate(
    emo_text = alt.datum.EMOJI + '         ' + alt.datum.PERCENT_TEXT
    ).mark_bar(color="#f6b800"
              ).encode(alt.X('PERCENT:Q',axis=None),
                       alt.Y('emo_text:N', sort='-x', title=None,
                             axis=alt.Axis(labelAlign='left', labelPadding=90))
    ).configure_view(strokeOpacity=0
                    ).configure_axis(
                        grid=False,
                        domain=False,
                        ticks=False,
                        labelColor="black",
                        labelFontWeight="bolder"
)

    return chart

#     raise NotImplementedError()
```

In [9]:
```
# test our solution
create_percentages_vis(percentages_df)
```

Out[9]:



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]

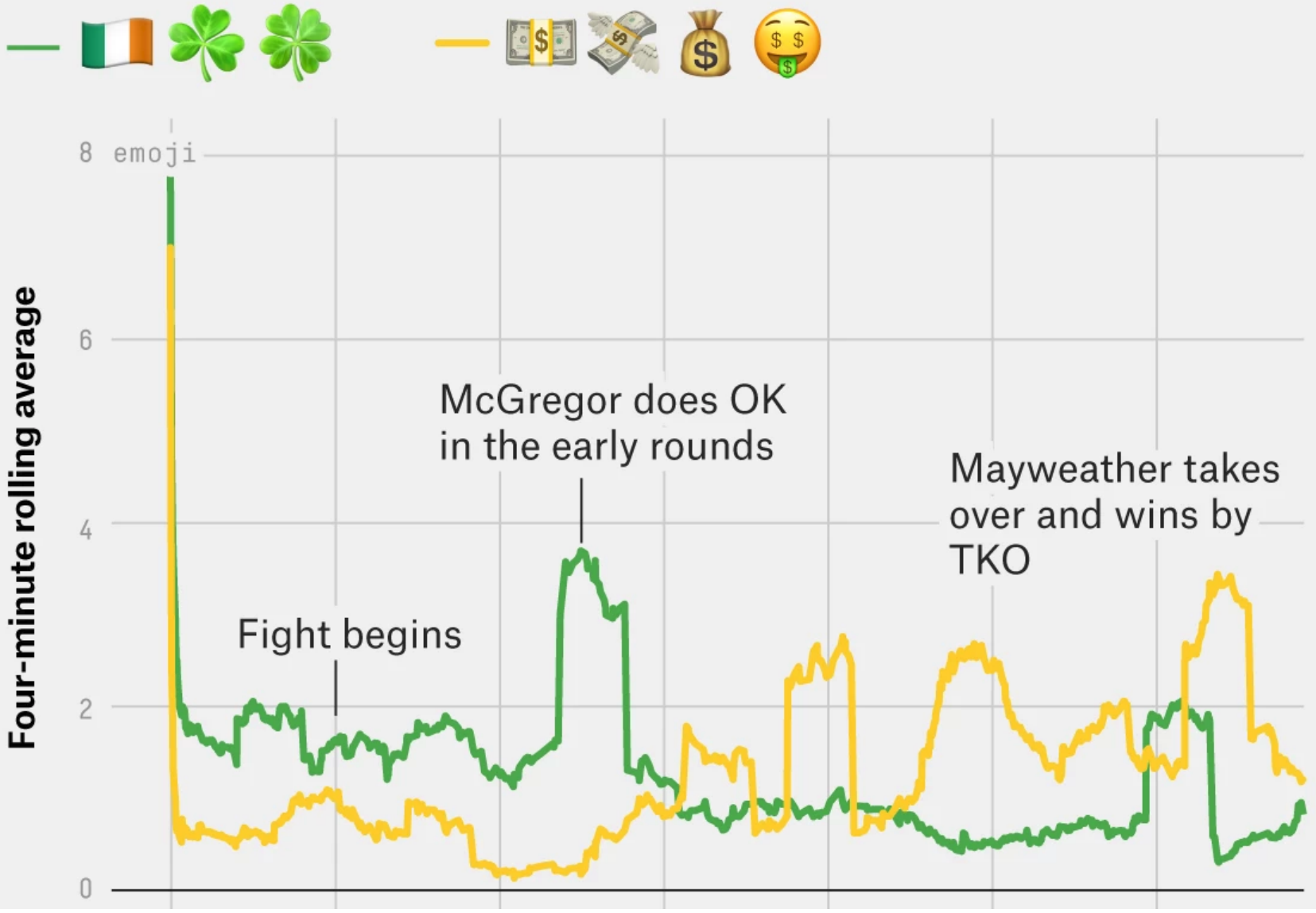
> 1.2. That's certainly appropriate for this spectacle, but it should be noted that 😂 is also the most tweeted (http://emojitracker.com/) emoji generally.

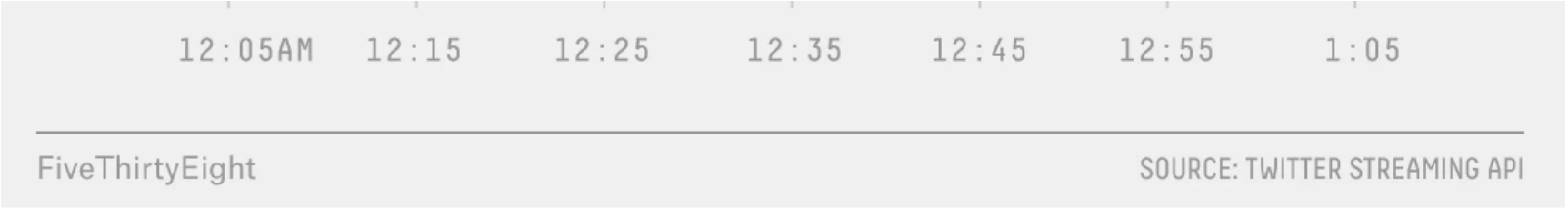
Here's how the night unfolded, emoji-wise. (All of the charts below show them on a four-minute rolling average.)

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.

Irish pride vs. The Money Team

Four-minute rolling average of the number of uses of selected emoji in sampled tweets during the Mayweather-McGregor fight





In [10]:
```
# 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

def createTimeSeries(indf):
    # input: indf, a df like the tweets database
    # ouput: the time series object limited to certain teams
    teams = indf.copy()
    teams['irish_pride']
    teams = teams.resample('1s').sum()
    teams  = teams[(teams['💸']>0) | (teams['🤑']>0) | (teams['💰']>0) | (teams['💵']>0) | (teams['☘']>0) | (teams['🍀']>0) | (teams['IE']>0) ]

    # next we're going to create 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'] = '☘🍀IE'
    idf = idf.rename(columns={'irish_pride':'tweet_count'})

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

In [11]:
```
ndf = createTimeSeries(tweets)
```

In [12]:
```
# uncomment to see what's inside
ndf.sample(5)
```

Out[12]:

	datetime	tweet_count	team
458	2017-08-27 00:47:18	0.923077	🍀🍀IE
109	2017-08-27 00:11:05	1.971429	🍀🍀IE
259	2017-08-27 00:20:45	1.739130	🍀🍀IE
703	2017-08-27 01:05:21	0.750000	🍀🍀IE
760	2017-08-27 01:10:24	3.363636	💸🤑💰💵

In [13]:
```
# we're also going to create an annotations data frame to help you

def createKeyPointsAnnotationsDF():
    # output: a data frame capturing the annotations at desired times (and placements in the vis)
    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'])
    return(a_df)
```

In [14]:
```
a_df = createKeyPointsAnnotationsDF()
```

In [15]:
```
# uncomment to see what's inside
a_df
```

Out[15]:

	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

2.2 Replicate the vis (7 points)

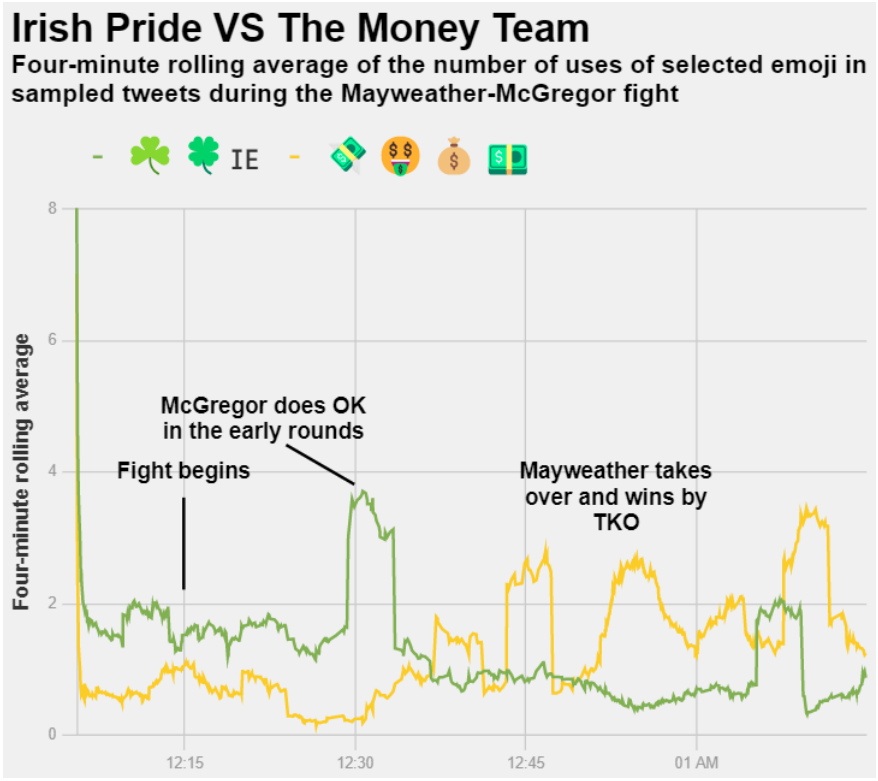
Construct your solution to the chart above ("Irish Money vs. The Money Team") in the cell below. Click here (assets/altair_chart2.png) to see a sample output we created with Altair.

In [16]:
```
# your turn, create your solution

def create_pride_vis(timeseries,annotations):
    # input: timeseries (a frame formatted like ndf above)
    # input: annotations (a frame formatted like a_df above)
    # return an Altair vis matching the example above
    # YOUR CODE HERE

    alt.renderers.set_embed_options(theme="fivethirtyeight")

    chart=alt.Chart(ndf).mark_line().encode(
    alt.X('datetime:T',
          axis=alt.Axis(tickCount=4),
          title=None),
    alt.Y('tweet_count:Q',
          scale=alt.Scale(bins=[0,2,4,6,8]),
          title="Four-minute rolling average"),
    alt.Color('team:N',
              scale=alt.Scale(domain=['☘️🍀IE', '💸🤑💰💵'], range=['#81b052', '#fccc26']),
              legend=alt.Legend(orient='top',
                                symbolType="stroke",
                                labelFontSize=26,
                                symbolSize=40),
              title=None)
    ).properties(height=350, width=525,
                 title={"text":"Irish Pride VS The Money Team",
                        "subtitle":["Four-minute rolling average of the number of uses of selected emoji in",
                                    "sampled tweets during the Mayweather-McGregor fight"],
                        "fontWeight":"bolder",
                        "fontSize":26,
                        "anchor":"start",
                        "subtitleFontSize":17,
                        "subtitleFontStyle":"bold"})

    text=alt.Chart(a_df).mark_text(align="center",
                                   baseline="middle",
                                   fontStyle="bold",
                                   fontSize=15,
                                   lineBreak='\n').encode(
    alt.X('date:T'),
    alt.Y('count'),
    text=alt.Text('note'))

    def createKeyPointsAnnotationsDF_lines_1():
        # output: a data frame capturing the annotations at desired times (and placements in the vis)
        annotations = [['2017-08-27 00:15:00',2.2],
                       ['2017-08-27 00:15:00',3.6]]
        a_df = pd.DataFrame(annotations, columns=['date','count'])
        return(a_df)

    lines_1_df=createKeyPointsAnnotationsDF_lines_1()

    lines_1=alt.Chart(lines_1_df).mark_line(color='black').encode(
    x='date:T',
    y='count')
```

```
    def createKeyPointsAnnotationsDF_lines_2():
        # output: a data frame capturing the annotations at desired times (and placements in the vis)
        annotations = [['2017-08-27 00:24:00',4.4],
                       ['2017-08-27 00:30:00',3.8]]
        a_df = pd.DataFrame(annotations, columns=['date','count'])
        return(a_df)

    lines_2_df=createKeyPointsAnnotationsDF_lines_2()

    lines_2=alt.Chart(lines_2_df).mark_line(color='black').encode(
    x='date:T',
    y='count')

    return (chart+text+lines_1+lines_2).configure_view(strokeOpacity=0)

#     raise NotImplementedError()
```

In [17]:
```
create_pride_vis(ndf,a_df)
```

Out[17]:



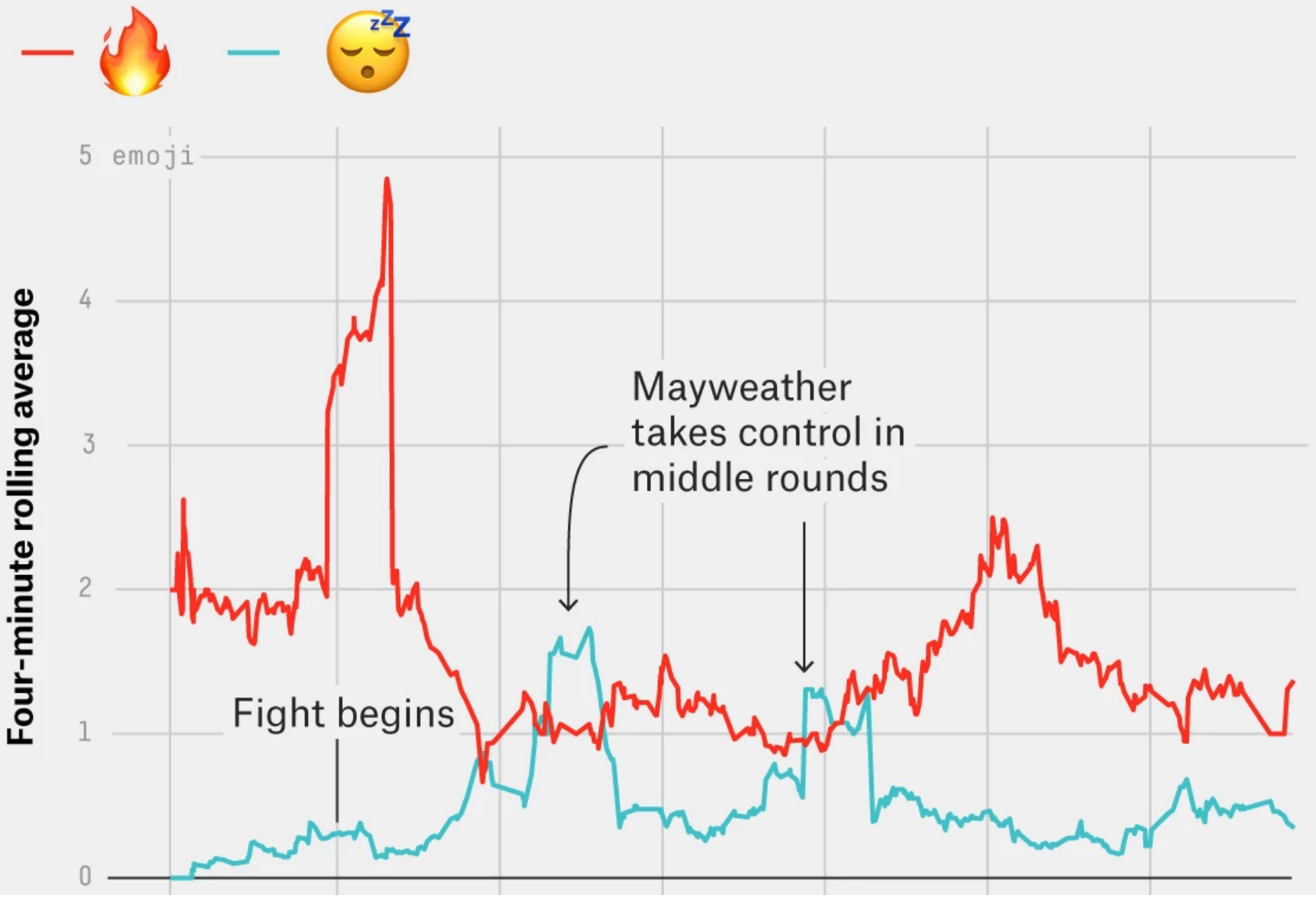
To the surprise of many (of the neutral and pro-Mayweather viewers, anyway) McGregor won the first round. The next couple were washes, and a quarter of the way into the [scheduled 12 rounds (https://www.nytimes.com/2017/08/26/sports/mayweather-mcgregor.html)](https://www.nytimes.com/2017/08/26/sports/mayweather-mcgregor.html) … the Irish underdog may have been winning! The Irish flags and shamrocks followed on Twitter. Things slowly (perhaps even 😴

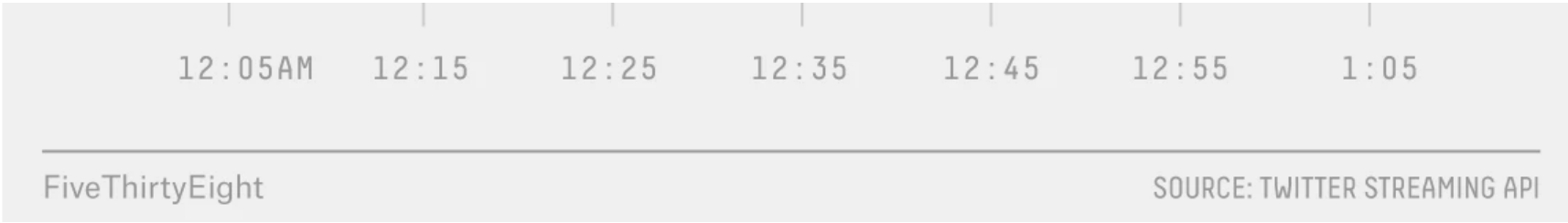
ly) turned around as one of the best pound-for-pound boxers in history took control of the man making his pro debut — an outcome which was predicted by precisely everyone. Out came the emoji money bags.

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.

Much hype, some boredom

Four-minute rolling average of the number of uses of selected emoji in sampled tweets during the Mayweather-McGregor fight





2.3 Replicate the vis (7 points)

Construct your solution to the chart above ("Much Hype, Some Boredom") in the cell below. Click here (assets/altair_chart3.png) to see a sample output we created with Altair.

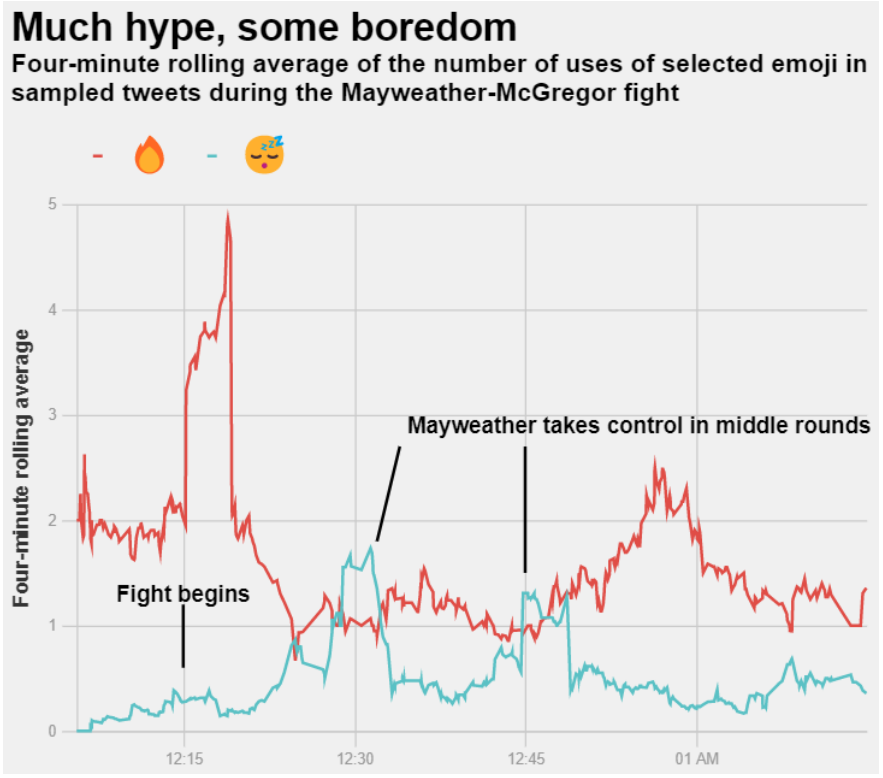
In [18]:
```
# your solution goes here, use the example above for the sampling and annotation

def create_hype_vis(indf):
    # input: indf (a frame formatted like tweets above)
    # return an Altair vis matching the example above

    # YOUR CODE HERE

    alt.renderers.set_embed_options(theme="fivethirtyeight")

    def createTimeSeries(indf):
        # input: indf, a df like the tweets database
        # ouput: the time series object limited to certain teams
        teams = indf.copy()
        teams = teams.resample('1s').sum()
        teams  = teams[(teams['🔥']>0)|(teams['😴']>0)]

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

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

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

    ndf = createTimeSeries(indf)

    chart=alt.Chart(ndf).mark_line().encode(
    alt.X('datetime:T',
          axis=alt.Axis(tickCount=4),
          title=None),
    alt.Y('tweet_count:Q',
          scale=alt.Scale(bins=[0,1,2,3,4,5]),
          axis=alt.Axis(tickMinStep=1),
          title="Four-minute rolling average"),
    alt.Color('team:N',
              scale=alt.Scale(domain=['🔥', '😴'], range=['#e04d46', '#5ac1c5']),
              legend=alt.Legend(orient='top',
                                symbolType="stroke",
                                labelFontSize=26,
                                symbolSize=40),
              title=None)
    ).properties(height=350, width=525,
                 title={"text":"Much hype, some boredom",
                        "subtitle":["Four-minute rolling average of the number of uses of selected emoji in",
                                    "sampled tweets during the Mayweather-McGregor fight"],
                        "fontWeight":"bolder",
                        "fontSize":26,
```

```
                    "anchor":"start",
                    "subtitleFontSize":17,
                    "subtitleFontStyle":"bold"})

def createKeyPointsAnnotationsDF():
    # output: a data frame capturing the annotations at desired times (and placements in the vis)
    annotations = [['2017-08-27 00:15:00',1.3, 'Fight begins'],
                   ['2017-08-27 00:55:00',2.9, 'Mayweather takes control in middle rounds']]
    a_df = pd.DataFrame(annotations, columns=['date','count','note'])
    return(a_df)

a_df = createKeyPointsAnnotationsDF()

text=alt.Chart(a_df).mark_text(align="center",
                           baseline="middle",
                           fontStyle="bold",
                           fontSize=15,
                           lineBreak='\n').encode(
alt.X('date:T'),
alt.Y('count'),
text=alt.Text('note'))

def createKeyPointsAnnotationsDF_lines_1():
    # output: a data frame capturing the annotations at desired times (and placements in the vis)
    annotations = [['2017-08-27 00:15:00',0.6],
                   ['2017-08-27 00:15:00',1.2]]
    a_df = pd.DataFrame(annotations, columns=['date','count'])
    return(a_df)

lines_1_df=createKeyPointsAnnotationsDF_lines_1()

lines_1=alt.Chart(lines_1_df).mark_line(color='black').encode(
    x='date:T',
    y='count')

def createKeyPointsAnnotationsDF_lines_2():
    # output: a data frame capturing the annotations at desired times (and placements in the vis)
    annotations = [['2017-08-27 00:32:00',1.8],
                   ['2017-08-27 00:34:00',2.7]]
    a_df = pd.DataFrame(annotations, columns=['date','count'])
    return(a_df)

lines_2_df=createKeyPointsAnnotationsDF_lines_2()

lines_2=alt.Chart(lines_2_df).mark_line(color='black').encode(
x='date:T',
y='count')

def createKeyPointsAnnotationsDF_lines_3():
    # output: a data frame capturing the annotations at desired times (and placements in the vis)
    annotations = [['2017-08-27 00:45:00',1.5],
                   ['2017-08-27 00:45:00',2.7]]
    a_df = pd.DataFrame(annotations, columns=['date','count'])
    return(a_df)

lines_3_df=createKeyPointsAnnotationsDF_lines_3()
```

```
    lines_3=alt.Chart(lines_3_df).mark_line(color='black').encode(
    x='date:T',
    y='count')

    return (chart+text+lines_1+lines_2+lines_3).configure_view(strokeOpacity=0)

#     raise NotImplementedError()
```

In [19]:
```
# test our solution
create_hype_vis(tweets)
```

Out[19]:



It ended just over 37 minutes after it began. Five seconds later, Mayweather leapt up on the corner ropes, victorious — [50-0 (https://fivethirtyeight.com/features/mayweather-is-defined-by-the-zero-next-to-his-name/)](https://fivethirtyeight.com/features/mayweather-is-defined-by-the-zero-next-to-his-name/). Some observers declared it a [satisfying spectacle (https://www.si.com/boxing/2017/08/27/after-months-hype-mayweather-and-mcgregor-deliver-boxing-spectacle)](https://www.si.com/boxing/2017/08/27/after-months-hype-mayweather-and-mcgregor-deliver-boxing-spectacle). Others, McGregor chief among them, [were frustrated with the finish (https://www.cbssports.com/boxing/news/conor-mcgregor-frustrated-with-refs-fight-stoppage-let-the-man-put-me-down/)](https://www.cbssports.com/boxing/news/conor-mcgregor-frustrated-with-refs-fight-stoppage-let-the-man-put-me-down/). 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 😴😴😴😴.

K.Zoldik
@John__Alp

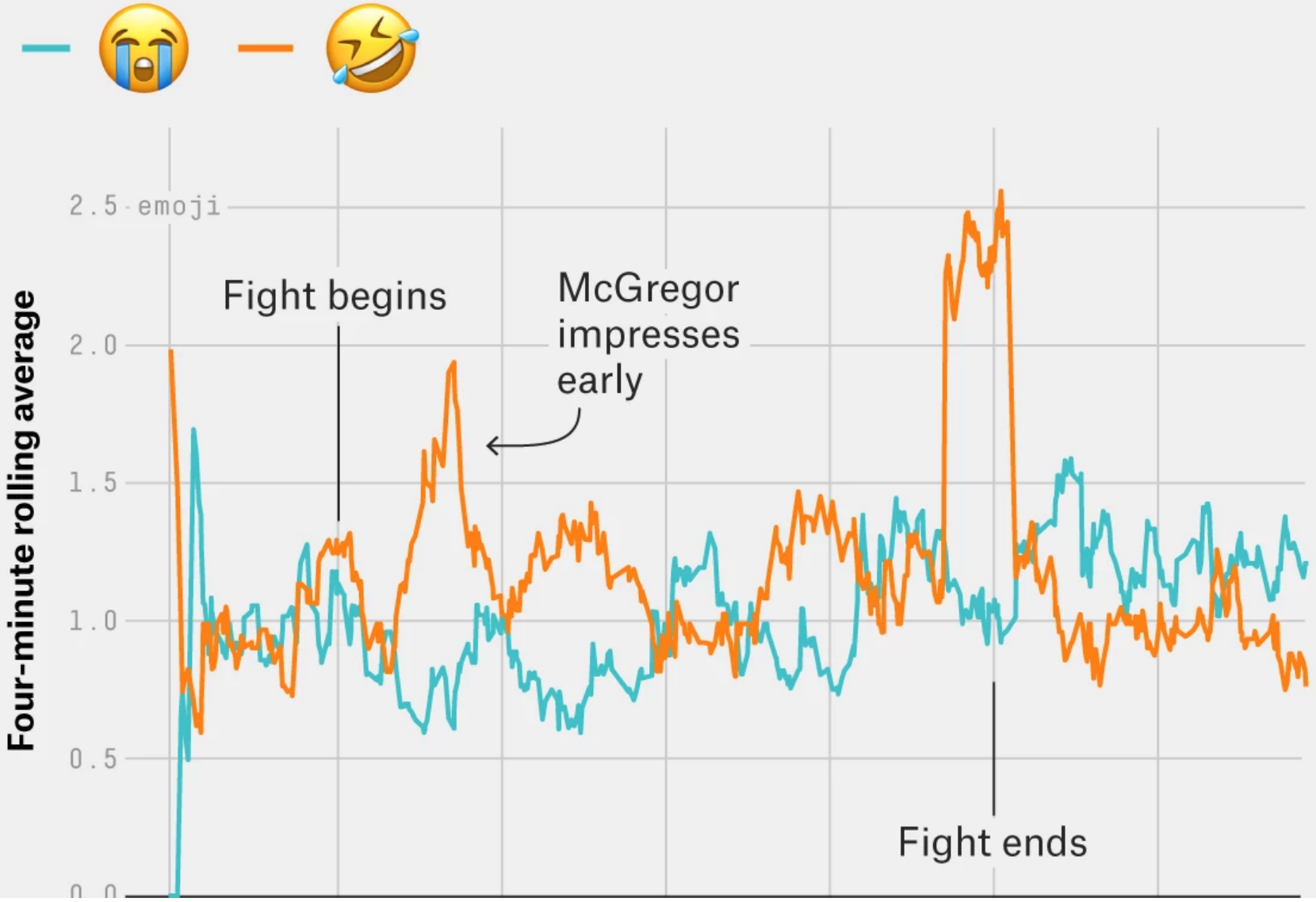
Yeeesssss !! 🔥 #Mayweather

12:51 AM - Aug 27, 2017

See K.Zoldik's other Tweets

Tears were shed — of joy and sorrow

Four-minute rolling average of the number of uses of selected emoji in sampled tweets during the Mayweather-McGregor fight



2.4 Replicate the vis (7 points)

Construct your solution to the chart above ("Tears were shed") in the cell below. Click here (assets/altair_chart4.png) to see a sample output we created with Altair.

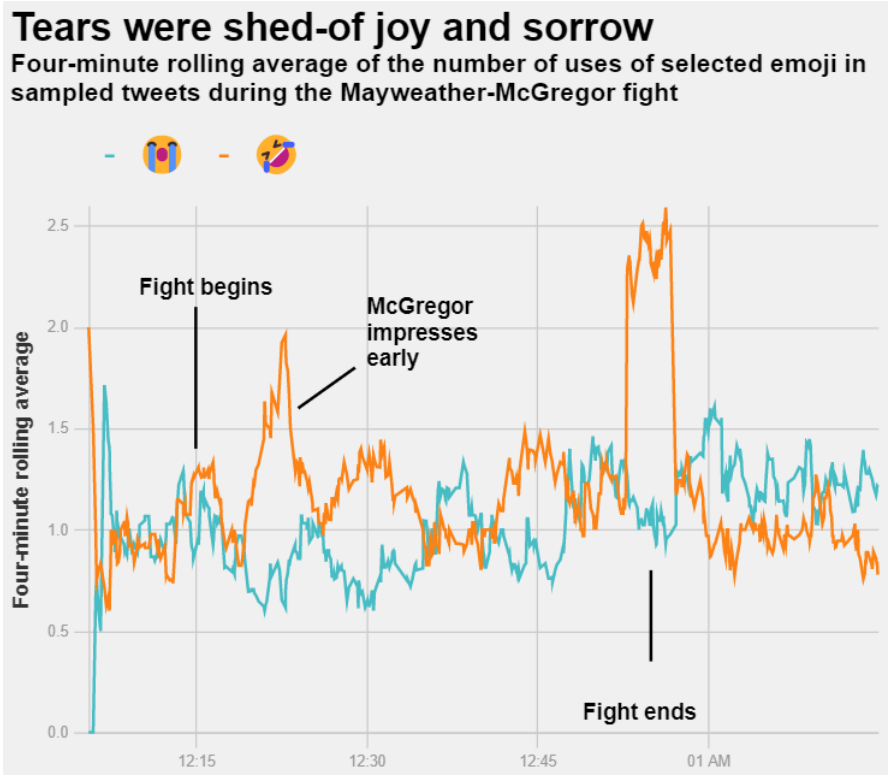
In [20]:
```
# your solution goes here, use the example above for the sampling and annotation

def create_tears_vis(indf):
    # input: indf (a frame formatted like tweets above)
    # return an Altair vis matching the example above

    # YOUR CODE HERE

    alt.renderers.set_embed_options(theme="fivethirtyeight")

    def createTimeSeries(indf):
        # input: indf, a df like the tweets database
        # ouput: the time series object limited to certain teams
        teams = indf.copy()
        teams = teams.resample('1s').sum()
        teams  = teams[(teams['😭']>0)|(teams['🤣']>0)]

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

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

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

    ndf = createTimeSeries(indf)

    chart=alt.Chart(ndf).mark_line().encode(
    alt.X('datetime:T',
          axis=alt.Axis(tickCount=4),
          title=None),
    alt.Y('tweet_count:Q',
          scale=alt.Scale(bins=[0,0.5,1,1.5,2,2.5]),
#           axis=alt.Axis(tickMinStep=1),
          title="Four-minute rolling average"),
    alt.Color('team:N',
              scale=alt.Scale(domain=['😭', '🤣'], range=['#46bcc4', '#fc8215']),
              legend=alt.Legend(orient='top',
                                symbolType="stroke",
                                labelFontSize=26,
                                symbolSize=40),
              title=None)
    ).properties(height=350, width=525,
                 title={"text":"Tears were shed-of joy and sorrow",
                        "subtitle":["Four-minute rolling average of the number of uses of selected emoji in",
                                    "sampled tweets during the Mayweather-McGregor fight"],
                        "fontWeight":"bolder",
                        "fontSize":26,
```

```
                    "anchor":"start",
                    "subtitleFontSize":17,
                    "subtitleFontStyle":"bold"})

def createKeyPointsAnnotationsDF():
    # output: a data frame capturing the annotations at desired times (and placements in the vis)
    annotations = [['2017-08-27 00:10:00',2.2, 'Fight begins'],
                   ['2017-08-27 00:30:00',2.1, 'McGregor\nimpresses\nearly'],
                   ['2017-08-27 00:49:00',0.1, 'Fight ends']]
    a_df = pd.DataFrame(annotations, columns=['date','count','note'])
    return(a_df)

a_df = createKeyPointsAnnotationsDF()

text=alt.Chart(a_df).mark_text(align="left",
                           baseline="middle",
                           fontStyle="bold",
                           fontSize=15,
                           lineBreak='\n').encode(
alt.X('date:T'),
alt.Y('count'),
text=alt.Text('note'))

def createKeyPointsAnnotationsDF_lines_1():
    # output: a data frame capturing the annotations at desired times (and placements in the vis)
    annotations = [['2017-08-27 00:15:00',1.4],
                   ['2017-08-27 00:15:00',2.1]]
    a_df = pd.DataFrame(annotations, columns=['date','count'])
    return(a_df)

lines_1_df=createKeyPointsAnnotationsDF_lines_1()

lines_1=alt.Chart(lines_1_df).mark_line(color='black').encode(
    x='date:T',
    y='count')

def createKeyPointsAnnotationsDF_lines_2():
    # output: a data frame capturing the annotations at desired times (and placements in the vis)
    annotations = [['2017-08-27 00:24:00',1.6],
                   ['2017-08-27 00:29:00',1.8]]
    a_df = pd.DataFrame(annotations, columns=['date','count'])
    return(a_df)

lines_2_df=createKeyPointsAnnotationsDF_lines_2()

lines_2=alt.Chart(lines_2_df).mark_line(color='black').encode(
x='date:T',
y='count')

def createKeyPointsAnnotationsDF_lines_3():
    # output: a data frame capturing the annotations at desired times (and placements in the vis)
    annotations = [['2017-08-27 00:55:00',0.35],
                   ['2017-08-27 00:55:00',0.8]]
    a_df = pd.DataFrame(annotations, columns=['date','count'])
    return(a_df)
```

```
    lines_3_df=createKeyPointsAnnotationsDF_lines_3()

    lines_3=alt.Chart(lines_3_df).mark_line(color='black').encode(
    x='date:T',
    y='count')

    return (chart+text+lines_1+lines_2+lines_3).configure_view(strokeOpacity=0)

#     raise NotImplementedError()
```

In [21]:
```
# test our code
create_tears_vis(tweets)
```

Out[21]:



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

2.5 Make your own (part 1-alternative, 10 points)

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. (10 points/ 5 points plot + 5 justification)

In [22]:

```
# add your code here
def create_tears_vis_scatter(indf):
    # input: indf (a frame formatted like tweets above)
    # return an Altair vis matching the example above

    # YOUR CODE HERE

    alt.renderers.set_embed_options(theme="fivethirtyeight")

    def createTimeSeries(indf):
        # input: indf, a df like the tweets database
        # ouput: the time series object limited to certain teams
        teams = indf.copy()
        teams = teams.resample('1s').sum()
        teams  = teams[(teams['😭']>0)|(teams['🤣']>0)]

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

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

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

    ndf = createTimeSeries(indf)

    chart=alt.Chart(ndf).mark_point().encode(
    alt.X('datetime:T',
          axis=alt.Axis(tickCount=4),
          title=None),
    alt.Y('tweet_count:Q',
          scale=alt.Scale(bins=[0,0.5,1,1.5,2,2.5]),
          title="Four-minute rolling average"),
    alt.Color('team:N',
              scale=alt.Scale(domain=['😭', '🤣'], range=['#46bcc4', '#fc8215']),
              legend=alt.Legend(orient='top',
                                symbolType="circle",
                                labelFontSize=26,
                                symbolSize=40),
              title=None),
    shape='team:N'
    ).properties(height=350, width=525,
                 title={"text":"Tears were shed-of joy and sorrow",
                        "subtitle":["Four-minute rolling average of the number of uses of selected emoji in",
                                    "sampled tweets during the Mayweather-McGregor fight"],
                        "fontWeight":"bolder",
                        "fontSize":26,
                        "anchor":"start",
```

```
                        "subtitleFontSize":17,
                        "subtitleFontStyle":"bold"})

    def createKeyPointsAnnotationsDF():
        # output: a data frame capturing the annotations at desired times (and placements in the vis)
        annotations = [['2017-08-27 00:10:00',2.2, 'Fight begins'],
                       ['2017-08-27 00:30:00',2.1, 'McGregor\nimpresses\nearly'],
                       ['2017-08-27 00:49:00',0.1, 'Fight ends']]
        a_df = pd.DataFrame(annotations, columns=['date','count','note'])
        return(a_df)

    a_df = createKeyPointsAnnotationsDF()

    text=alt.Chart(a_df).mark_text(align="left",
                                   baseline="middle",
                                   fontStyle="bold",
                                   fontSize=15,
                                   lineBreak='\n').encode(
    alt.X('date:T'),
    alt.Y('count'),
    text=alt.Text('note'))

    return (chart+text).configure_view(strokeOpacity=0)

#     raise NotImplementedError()

create_tears_vis_scatter(tweets)
```

Out[22]:

Provide your justification here

The alternative visualization I have proposed is a scatter plot for the "Tears" vis.

The original vis had a line chart, while the one I have coded for is a scatter plot. When there are a large number of data points, a scatter plot is often useful for visualizing clusters and outliers. In this case, we can see close to the end of the fight towards the top right corner of the chart that there seems to be a sudden uptick of '🤣' usage since a lot of data points are clustered over there.

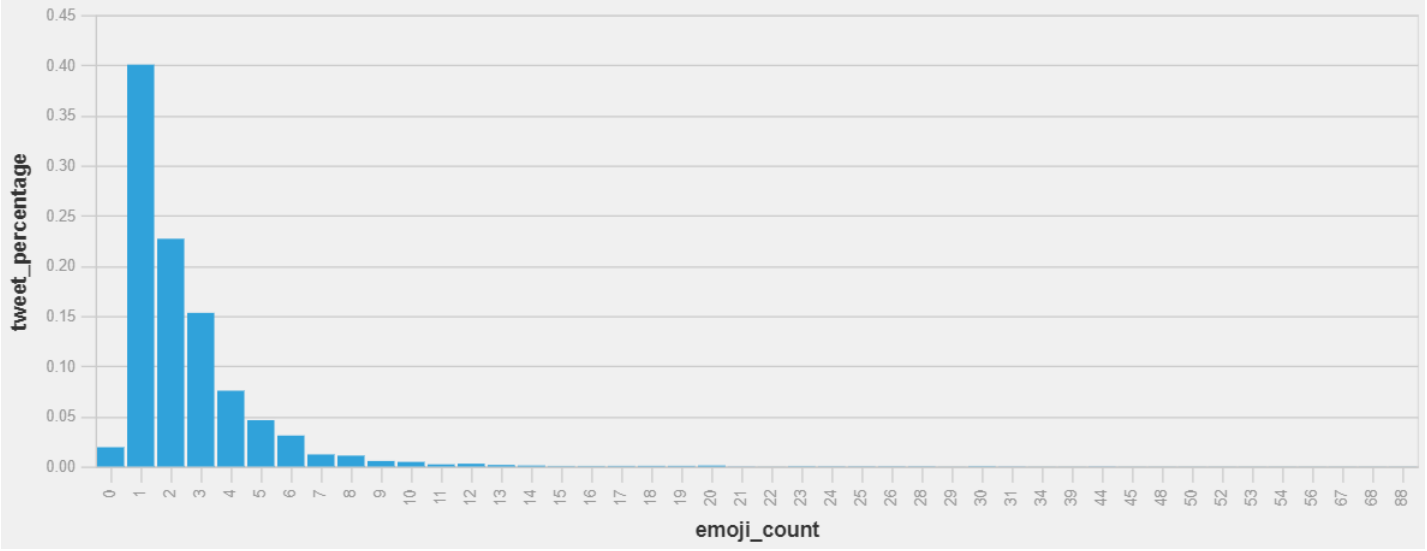
From a Perception and Cognition standpoint, the use of multiple encodings as I have coded for (i.e. the nominal variable 'team' being denoted by different shapes and different colors) makes my chart more effective than the original one and especially useful as it enables clear distinction between both of the categories. It is also worthwhile to consider that the chart in the article wouldn't have made sense if a black-and-white printer were used since both of the lines would've been indistinguishable, whereas the one I have proposed would be easily readable since the shapes are easily distinguishable even without color.

2.6 Make your own (part 2-novel, 30 points)

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. (30 points/ 20 points plot + 10 justification)

In [23]:
```
# add your code here

# create a dataframe containing emoji count and corresponding number of tweets
emoji_df=pd.DataFrame(tweets.emojis.value_counts()).reset_index().rename(columns={"index":"emoji_count","emojis":"tweet_count"})
# create a percentage column
emoji_df["tweet_percentage"]=emoji_df["tweet_count"]/len(tweets)
# create a bar chart from the dataframe above
alt.Chart(emoji_df).mark_bar().encode(x='emoji_count:O',y='tweet_percentage:Q')
```

Out[23]:



Provide your justification here

The complementary visualization I have proposed is a bar chart for Emoji Count and Tweet Percentage.

This bar chart would give us an idea of what number of emojis were most popular in the sample under consideration. In this case, most tweets on this topic (nearly five thousand of them or 40%) had just one emoji, while one tweet had as many as 88 emojis.

This is a really interesting visualization that I thought the article should have included. It is noteworthy that most tweeters in this sample are taciturn at least in terms of emoji usage with a sizable portion of them using just one emoji in their tweets. We can even see that there are nearly 250 of them who do not use a single emoji in their tweets, while some people use more than 80 of them! Since the article talks about trying to tell a story through emoji usage, this vis would've added additional insight to the narrative.