Information Visualization I

School of Information, University of Michigan

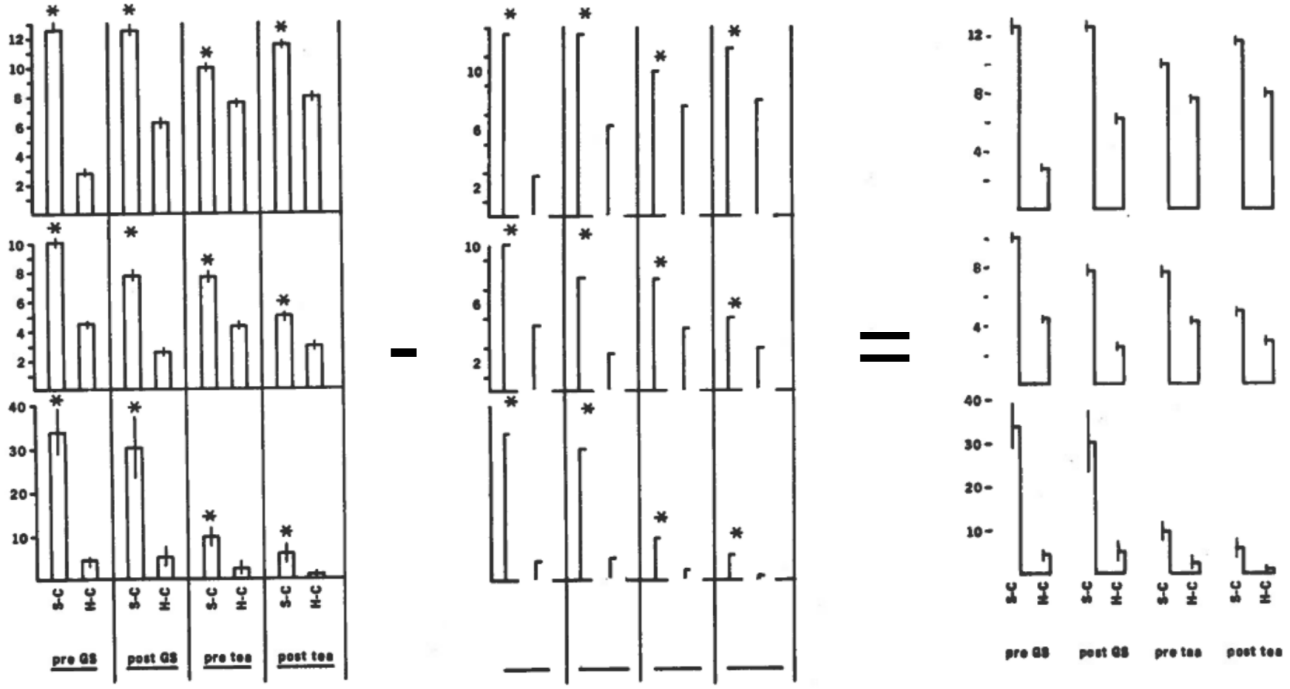
Week 4:

- 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.

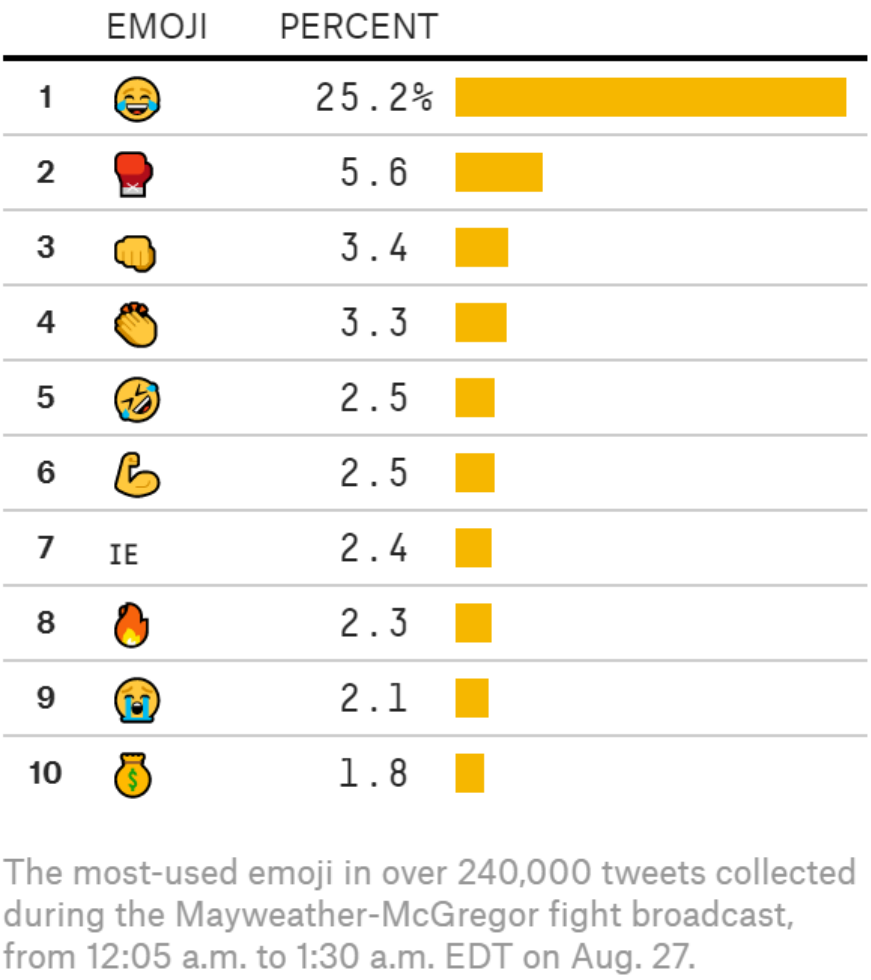
4) Follow the instructions for submission on Coursera. You will be providing us a generated link to a read-only version of your notebook and a PDF. When turning in your PDF, please use the File -> Print -> Save as PDF option ***from your browser***. Do ***not*** use the File->Download as->PDF option. 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)

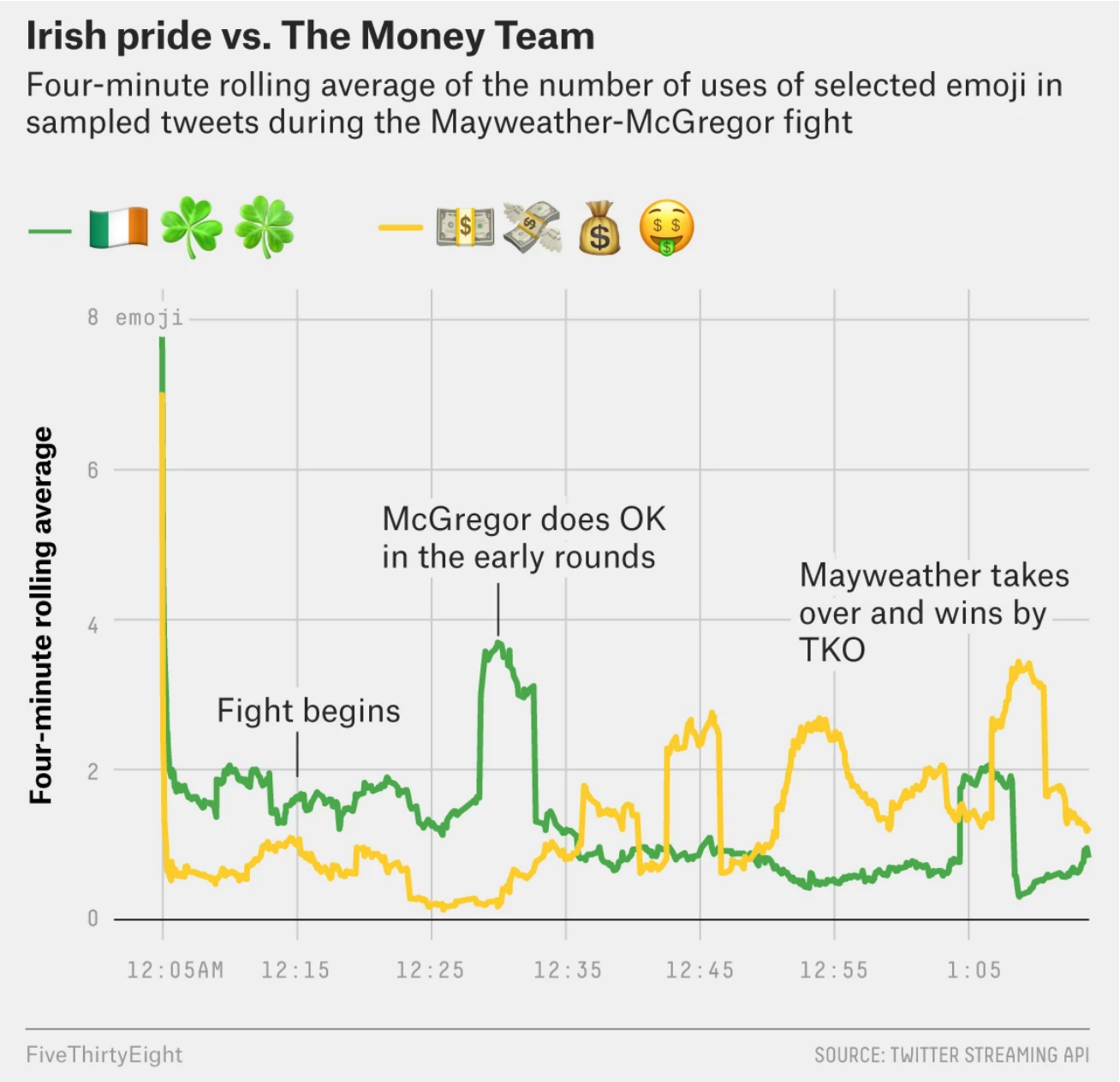
Read the article published in Five Thirty Eight The Mayweather-McGregor Fight, As Told Through Emojis (https://fivethirtyeight.com/features/the-mayweather-mcgregor-fight-as-told-through-emojis/) and answer the following questions:

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)



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.



1.1 Answer

Emoji/Percent Bar Chart:

mark_bar.encode() --> PERCENT, it is Q & N, color of bar, ordered by rank descending mark_text.encode() --> EMOJI, it is just N

Irish Pride vs. The Money Team:

mark_line.encode() --> X axis is datetime:T, Y axis is count of Tweets:Q, color (green and yellow), ordered by time mark_text.encode() --> date, count, note:N

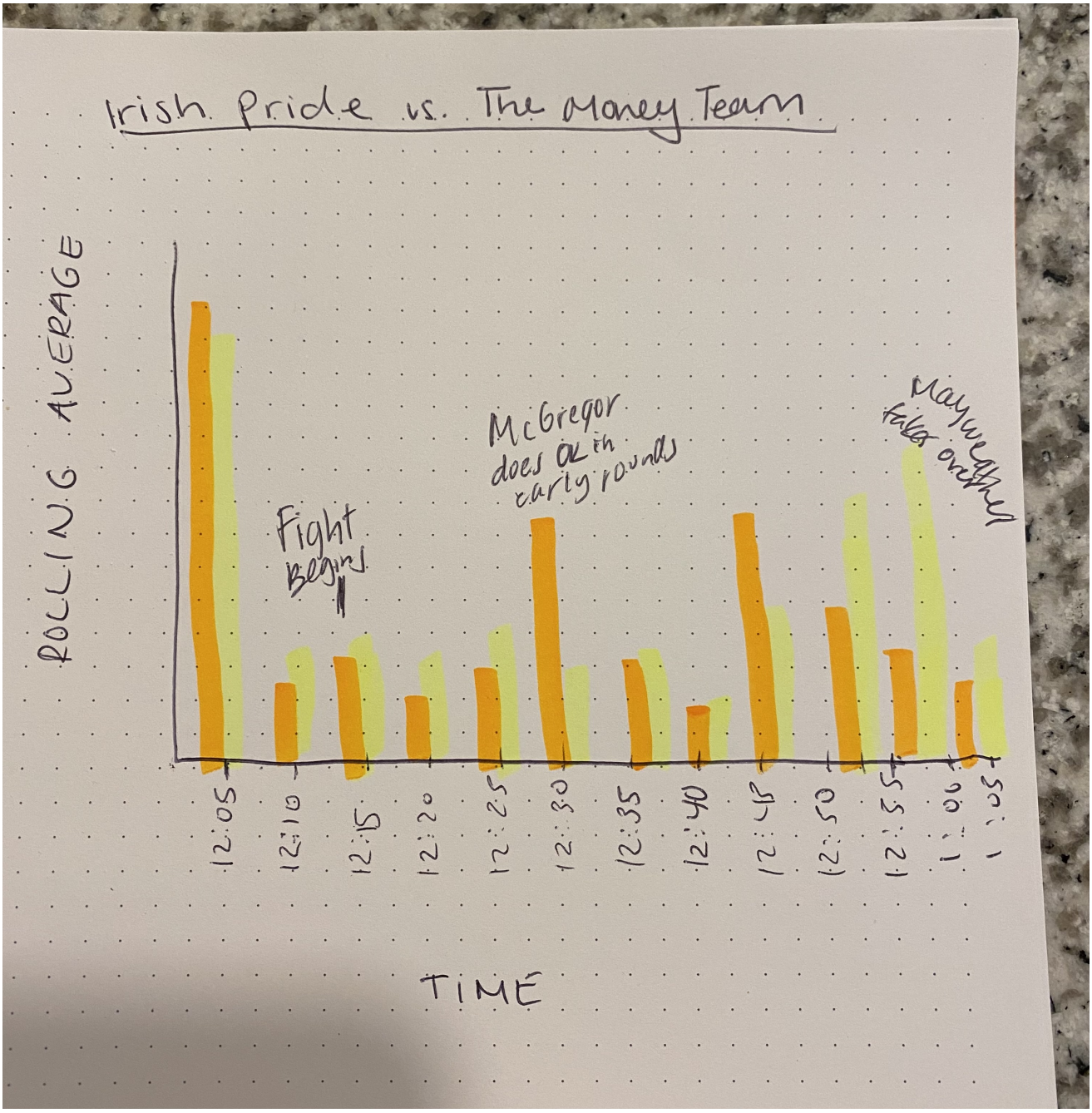
axes are time and rolling average

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.

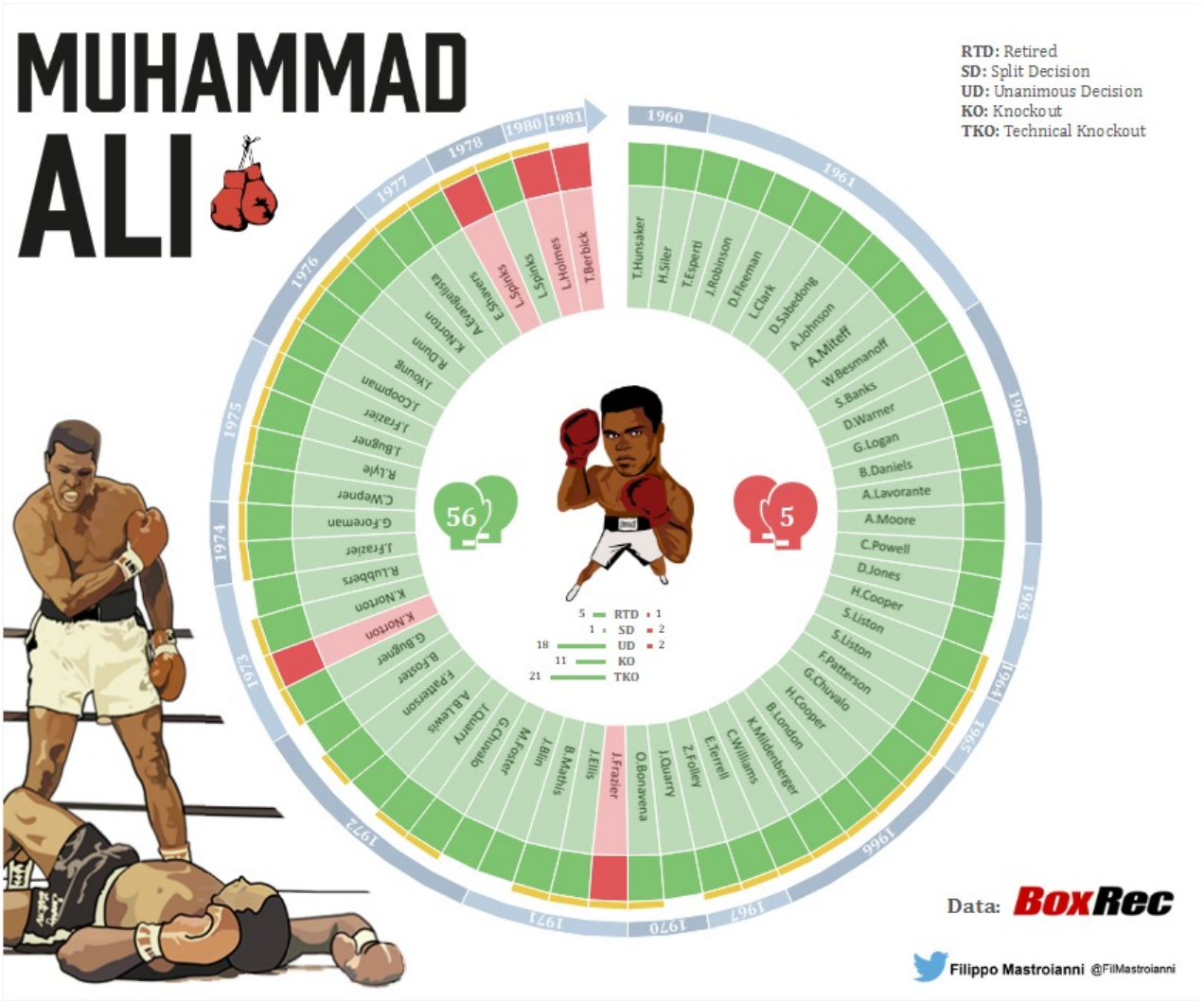
1.2 Answer

My solution and the original solution both follow the laws of proximity and similarity, but where the original solution wins is in its following the law of continuity. While both are relatively simple graphs, the continuity of the original makes a stronger visualization as time is a continuous factor. The original visualization is rather self explanatory where as mine could use additional annotations.



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

Tufte's data-ink ratio, graphical integrity, chart junk, etc.

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.

The data-ink ratio is much too high. As Tufte suggests, elements that do not provide new information to the visualization should be removed so that the data-ink ratio can go down.

- The gold marks are not intutive, remove
- Merge the legends into one, although it is useful to have the top right legend which allows us to understand what the acronyms stand for

He also says to remove chart junk

- I think it's great to have an image of Ali, however there are too many and it is crowding the visualization. The images also do not bring additional information, and his name with the boxing gloves serves as plently to get the message across.
- I like the green and red boxing gloves in the center which depict the number of wins and loses.

White space

- There is not enough whitespace in this visualization. As Tufte says, whitespace improves readability.

Graphical integrity

- Seems to follow this principle. While the visualization is crowded with information, it seems to follow scale.

Overall, this visualization is fun and informative but it violates some of Tufte's main points. 1) Maintain a low data to ink ratio, only keep chart junk if it adds value, ensure plenty of white space for readability.

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/). 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-\U0001I

    # 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	
2017-08-27 00:05:34	2017-08-27 00:05:34	1	901656910939770881	https://twitter.com/statuses/901656910939770881	False	aaLiysr	Ringe çıkmac etmeye ba #McG
2017-08-27 00:05:35	2017-08-27 00:05:35	5	901656917281574912	https://twitter.com/statuses/901656917281574912	False	zulmafrancozaf	😍😍😍😍😍 @laly https://t.co/ERUG
2017-08-27 00:05:35	2017-08-27 00:05:35	2	901656917105369088	https://twitter.com/statuses/901656917105369088	False	Adriana11D	🇮🇪 #MayweathervM
2017-08-27 00:05:35	2017-08-27 00:05:35	2	901656917747142657	https://twitter.com/statuses/901656917747142657	False	Nathan_Caro_	Ce #MayweatherM
2017-08-27 00:05:35	2017-08-27 00:05:35	2	901656916828594177	https://twitter.com/statuses/901656916828594177	False	sahouraxox	Low key feeling ppl who payed to

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) 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.

	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.

** 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]:
```
# 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 = ['😂','🤣','🍣','👊','👏','🇮🇪','💪','🔥','😭','💰']
    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'})

    # 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	PERCENT_TEXT
0	😂	23.1	23.1 %
1	🍣	5.7	5.7 %
2	👊	3.5	3.5 %
3	👏	3.0	3.0 %
4	💪	2.5	2.5 %
5	🇮🇪	2.4	2.4 %
6	🤣	2.3	2.3 %
7	🔥	2.3	2.3 %
8	😭	2.0	2.0 %
9	💰	1.8	1.8 %

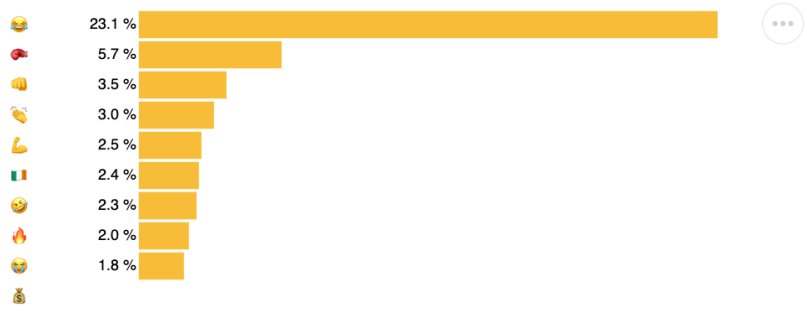
2.1 Replicate the vis (9 points)

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):

    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'})

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

    right = alt.Chart(indf).mark_bar(color='#F8BD38').encode(
        alt.X('PERCENT:Q',axis=None),
        alt.Y('PERCENT_TEXT:N', sort='-x', title=None)
    )

    left = alt.Chart(indf).mark_text().encode(
        alt.Y('EMOJI:N', sort=alt.EncodingSortField('PERCENT', order="descending"), title=None)
    )

    chart = alt.hconcat(
        left, right
    ).configure_view(strokeOpacity=0).configure_axis(
        grid=False, domain=False,ticks=False
    )

    return chart

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

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

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

Out[9]:



(Missing a bar for one emoji as two of them equal 2.3%)

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]

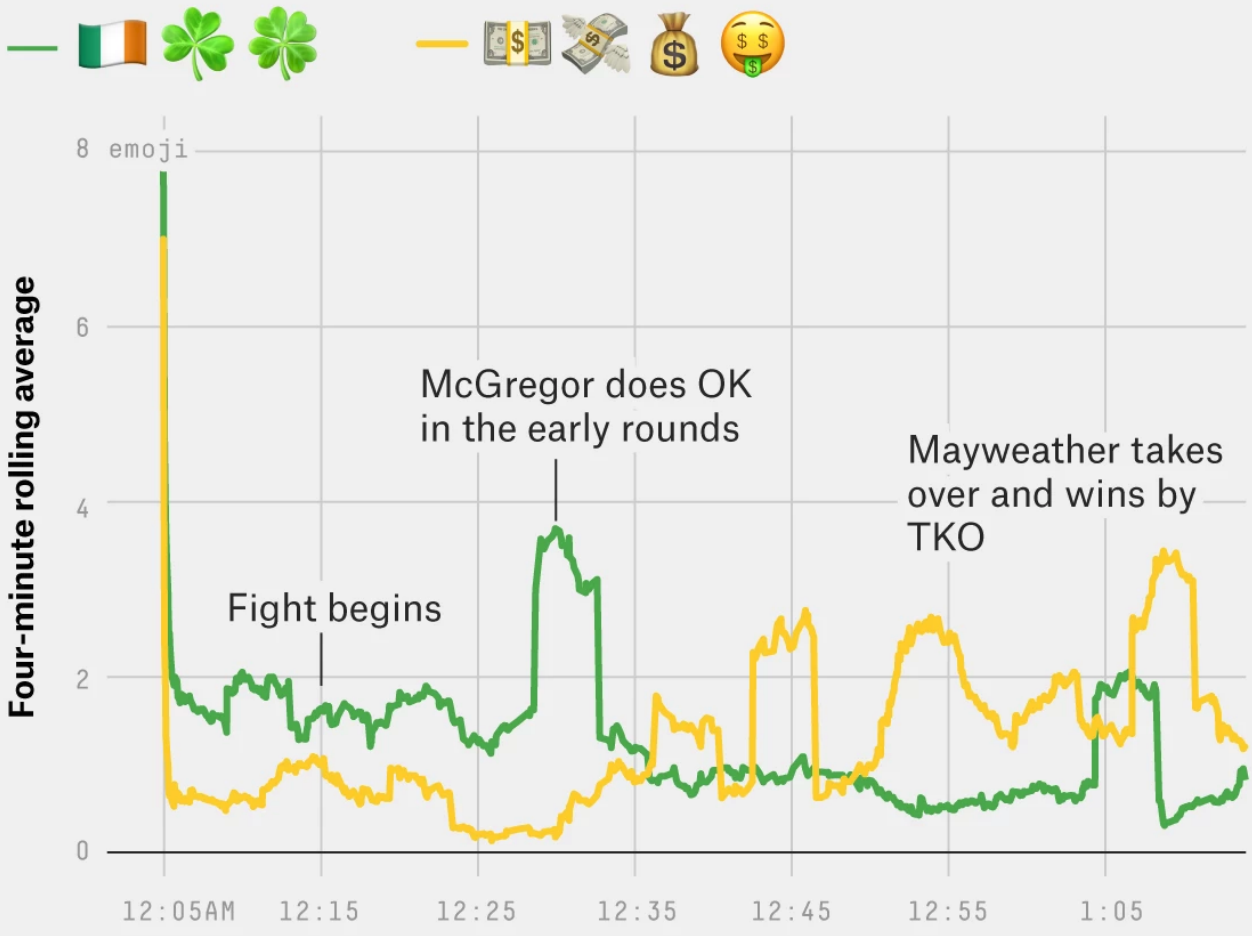
> 1.2. That's certainly appropriate for this spectacle, but it should be noted that 😂 is also the most tweeted (http://emojitracker.com/) emoji generally.

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



FiveThirtyEight

SOURCE: TWITTER STREAMING API

In [10]:
```
, we're going to help you set up the data

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


eateTimeSeries(indf):
Input: indf, a df like the tweets database
Ouput: the time series object limited to certain teams
ms = indf.copy()
ms['irish_pride']
ms = teams.resample('1s').sum()
ms  = teams[(teams['🚀']>0) | (teams['🤑']>0) | (teams['💰']>0) | (teams['💵']>0) | (teams['☘']>0) | (tea

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

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

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

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

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

Out[12]:

	datetime	tweet_count	team
124	2017-08-27 00:11:40	0.581081	🚀🤑💰💵
120	2017-08-27 00:11:27	0.539474	🚀🤑💰💵
460	2017-08-27 00:47:46	0.875000	☘🍀🇮🇪
777	2017-08-27 01:11:57	1.711111	🚀🤑💰💵
36	2017-08-27 00:06:54	1.729730	☘🍀🇮🇪

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',2.7, 'Fight begins'],
                   ['2017-08-27 00:30:00',5, 'McGregor does OK \nin the early rounds'],
                   ['2017-08-27 00:55:00',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	2.7	Fight begins
1	2017-08-27 00:30:00	5.0	McGregor does OK \nin the early rounds
2	2017-08-27 00:55:00	4.0	Mayweather takes \nover and wins by \nTKO

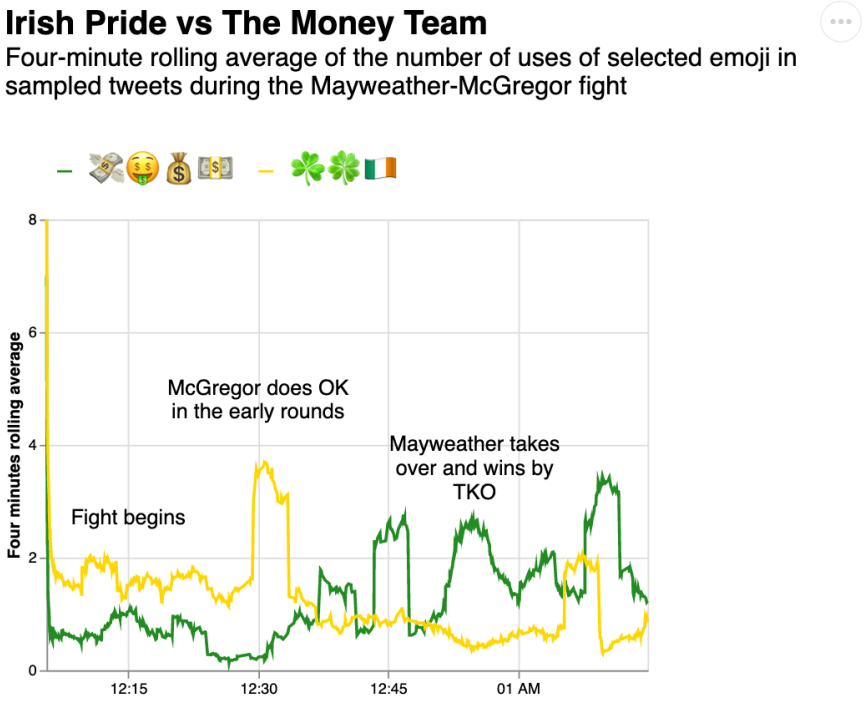
2.2 Replicate the vis (7 points)

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
    #raise NotImplementedError()

    time1 = alt.Chart(timeseries).mark_line().encode(
        x = alt.X('datetime:T', axis = alt.Axis(tickCount=4, title='')),
        y = alt.Y('tweet_count:Q', title = 'Four minutes rolling average', axis=alt.Axis(tickCount=4)),
        color = alt.Color('team', scale = alt.Scale(domain=['💸🤑💰💵','☘️🍀🇮🇪'], range=['forestgreen', 'gc
                          legend = alt.Legend(labelFontSize=24, symbolSize=100, direction = 'horizontal',
                                              orient='top')
                          ))

    note1 = alt.Chart(annotations).mark_text(fontSize=14, lineBreak='\n').encode(
        x = alt.X('date:T', axis=None),
        y = alt.Y('count:Q', axis=None),
        text = alt.Text('note:N'))

    visualization1 = (time1 + note1).properties(
        title = {
            "text": ["Irish Pride vs The Money Team"],
            "subtitle": ["Four-minute rolling average of the number of uses of selected emoji in", "sampl
            "fontSize":22,
            "subtitleFontSize":17,
            "anchor":"start",
            "offset":35
        }
    ).configure_title(
        anchor = 'start')

    return visualization1
```

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

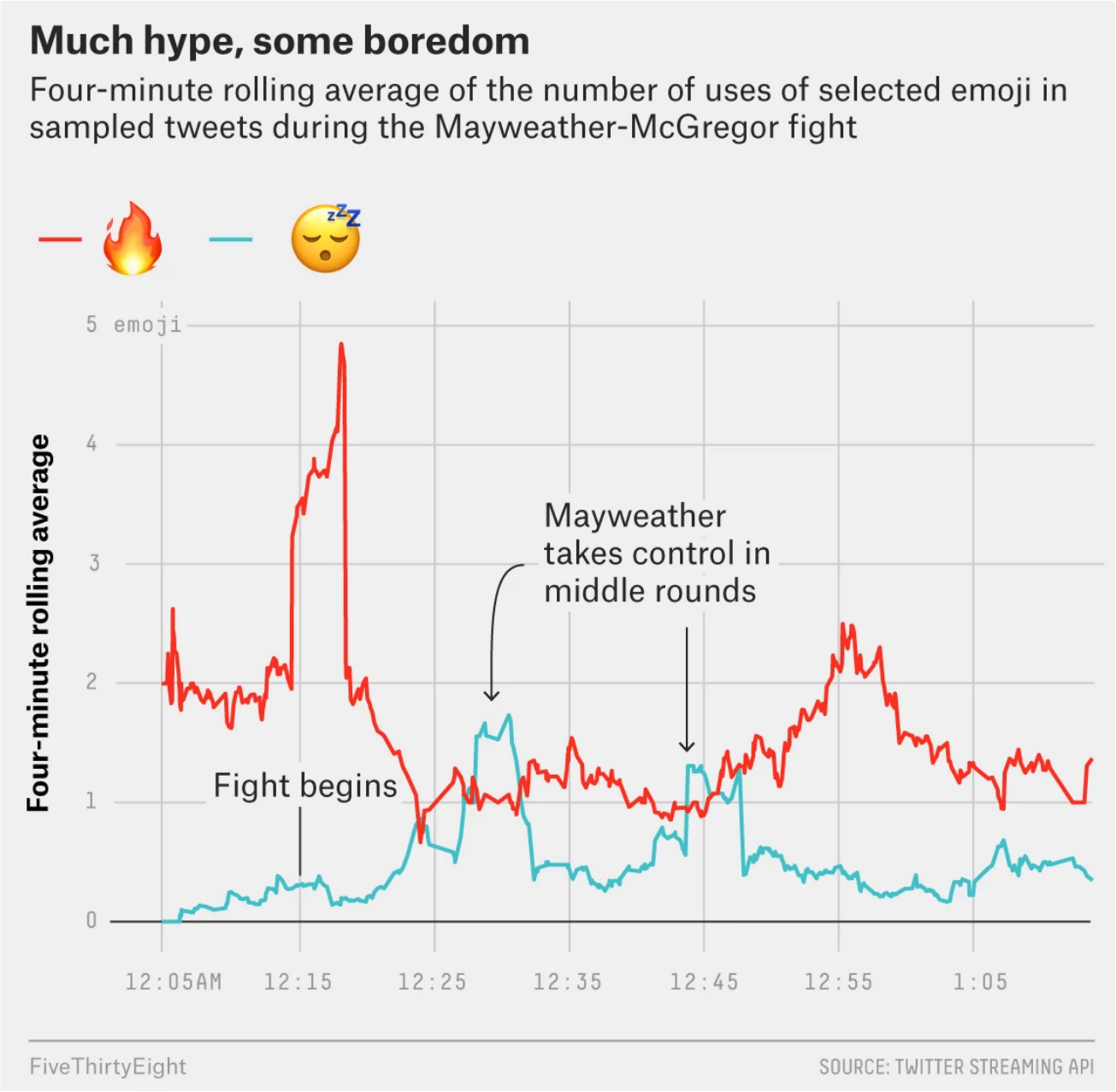
Out[17]:



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) … 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.



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.

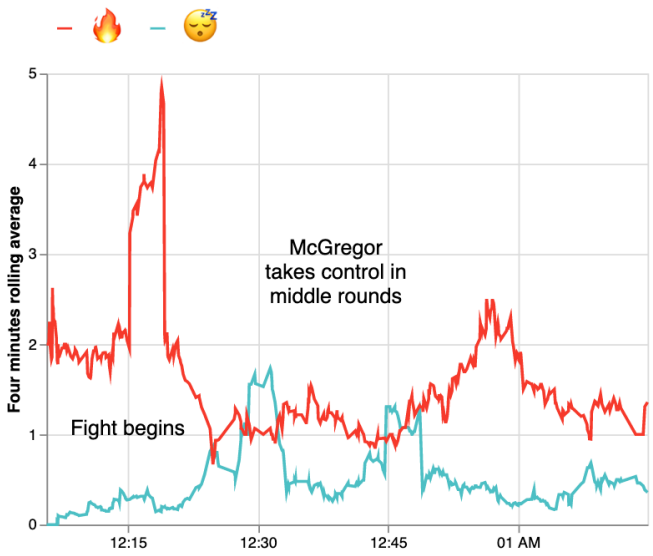
In [18]:
```
# your solution goes here, use the example above for the sampling and annotation

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

    # YOUR CODE HERE
    #raise NotImplementedError

    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])

    annotations = [['2017-08-27 00:15:00', 1.06, 'Fight begins'],
                   ['2017-08-27 00:39:00', 3.06, 'McGregor\ntakes control in\nmiddle rounds']]

    a_df = pd.DataFrame(annotations, columns=['date','count','note'])

    time2 = alt.Chart(ndf).mark_line().encode(
        x = alt.X('datetime:T', axis = alt.Axis(tickCount=4, title='')),
        y = alt.Y('tweet_count:Q', title = 'Four minutes rolling average', axis=alt.Axis(tickCount=4)),
        color = alt.Color('team', scale = alt.Scale(domain=['🔥','😴'], range=['#f6392b', '#4cbfc4']),
                          legend = alt.Legend(labelFontSize=24, symbolSize=100, direction = 'horizontal',
                                              orient='top')
                          ))

    note2 = alt.Chart(a_df).mark_text(fontSize=14, lineBreak='\n').encode(
        x = alt.X('date:T', axis=None),
        y = alt.Y('count:Q', axis=None),
        text = alt.Text('note:N'))

    visualization2 = (time2 + note2).properties(
        title = {
            "text": ["Much hype, some boredom"],
            "subtitle": ["Four-minute rolling average of the number of uses of selected emoji in", "sampl
            "fontSize":22,
            "subtitleFontSize":17,
            "anchor":"start",
            "offset":35
        }
    ).configure_title(
        anchor = 'start')

    return visualization2
```

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

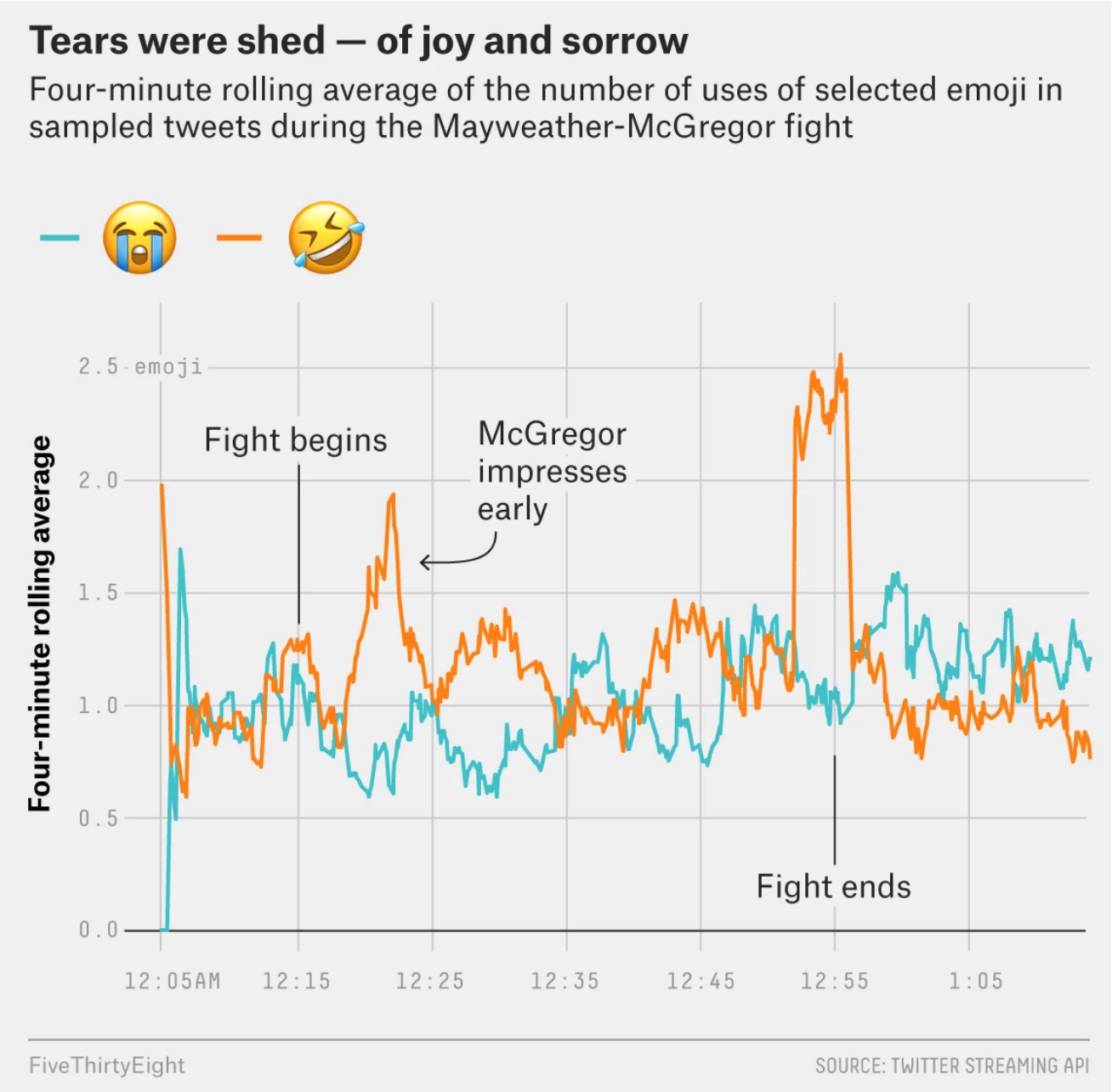
Out[19]:

Much hype, some boredom

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



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/). Some observers declared it a satisfying 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/). 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 😴😴😴😴.



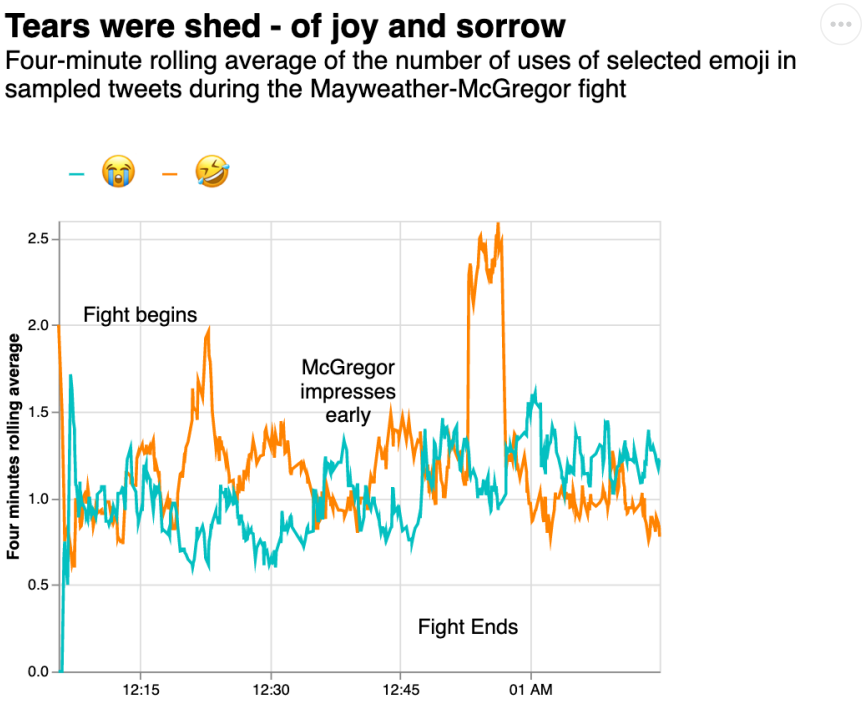
2.4 Replicate the vis (7 points)

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 [63]:
```
# 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
    #raise NotImplementedError()
    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])

    annotations = [['2017-08-27 00:15:00',2.05, 'Fight begins'],
                   ['2017-08-27 00:39:00', 1.75, 'McGregor\nimpresses\nearly'],
                   ['2017-08-27 00:52:50', 0.25, 'Fight Ends']]
    a_df = pd.DataFrame(annotations, columns=['date','count','note'])

    time3 = alt.Chart(ndf).mark_line().encode(
        x = alt.X('datetime:T', axis = alt.Axis(tickCount=4, title='')),
        y = alt.Y('tweet_count:Q', title = 'Four minutes rolling average', axis=alt.Axis(tickCount=4)),
        color = alt.Color('team', scale = alt.Scale(domain=['😭','🤣'], range=['#03C0C1', '#ff8300']),
                          legend = alt.Legend(labelFontSize=24, symbolSize=100,direction = 'horizontal', t
                                              orient='top')
                          ))

    note3 = alt.Chart(a_df).mark_text(fontSize=14, lineBreak='\n').encode(
        x = alt.X('date:T', axis=None),
        y = alt.Y('count:Q', axis=None),
        text = alt.Text('note:N'))


    visualization3 = (time3 + note3).properties(
        title = {
            "text": ["Tears were shed - of joy and sorrow"],
            "subtitle": ["Four-minute rolling average of the number of uses of selected emoji in", "sampl
            "fontSize":22,
           "subtitleFontSize":17,
           "anchor":"start",
           "offset":35
        }
    ).configure_title(
        anchor = 'start')

    return visualization3
```

In [64]:
```
# test our code
create_tears_vis(tweets)
```

Out[64]:



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

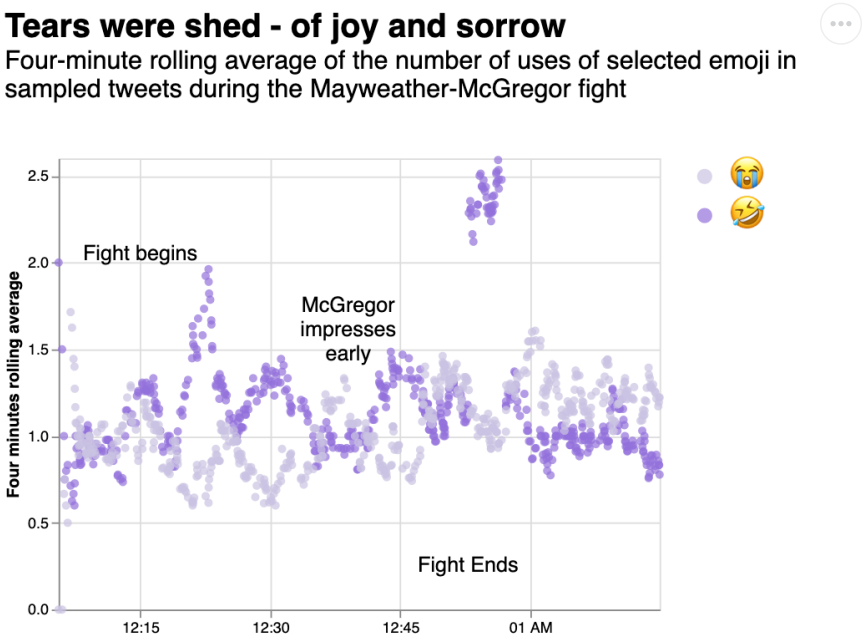
2.5 Make your own (part 1-alternative, 10 points)

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 [65]:
```
def create_tears_vis_alternative(indf):
    # input: indf (a frame formatted like tweets above)
    # return an Altair vis matching the example above

    # YOUR CODE HERE
    #raise NotImplementedError()
    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])

    annotations = [['2017-08-27 00:15:00',2.05, 'Fight begins'],
                   ['2017-08-27 00:39:00', 1.75, 'McGregor\nimpresses\nearly'],
                   ['2017-08-27 00:52:50', 0.25, 'Fight Ends']]
    a_df = pd.DataFrame(annotations, columns=['date','count','note'])

    time3 = alt.Chart(ndf).mark_circle().encode(
        x = alt.X('datetime:T', axis = alt.Axis(tickCount=4, title='')),
        y = alt.Y('tweet_count:Q', title = 'Four minutes rolling average', axis=alt.Axis(tickCount=4)),
        color = alt.Color('team', scale = alt.Scale(domain=['😭','🤣'], range=['#CBC3E3', '#9370db']),
                          legend = alt.Legend(labelFontSize=24, symbolSize=100,direction = 'vertical', tit
                                              orient='right')
                          ))

    note3 = alt.Chart(a_df).mark_text(fontSize=14, lineBreak='\n').encode(
        x = alt.X('date:T', axis=None),
        y = alt.Y('count:Q', axis=None),
        text = alt.Text('note:N'))


    visualization3 = (time3 + note3).properties(
        title = {
            "text": ["Tears were shed - of joy and sorrow"],
            "subtitle": ["Four-minute rolling average of the number of uses of selected emoji in", "samp
            "fontSize":22,
           "subtitleFontSize":17,
           "anchor":"start",
           "offset":35
        }
    ).configure_title(
        anchor = 'start')

    return visualization3
```

In [66]:
```
create_tears_vis_alternative(tweets)
```

Out[66]:



The alternative visualization is worse because it does not follow Gestalt's principle of continuity. This visualization would be better if there was a line drawn connecting the dots to show that while they are connected and represent one element, that they are showing points over a period of time.

The visualization follows the law of proximity to a fault. This visualization's size needs to be expanded in order to attempt to reduce overlap between the points.

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)

Unfortunately I ran out of time, but I wanted to provide my attempted visualization and the reason why I chose it.

I thought it would be really great to see the percentage of emojis used over time throughout the fight like this. The color coordination makes the chart appealing, readable, and digestable.

It follows Gestalt's principles of continuity, common region, proximity and similarity.

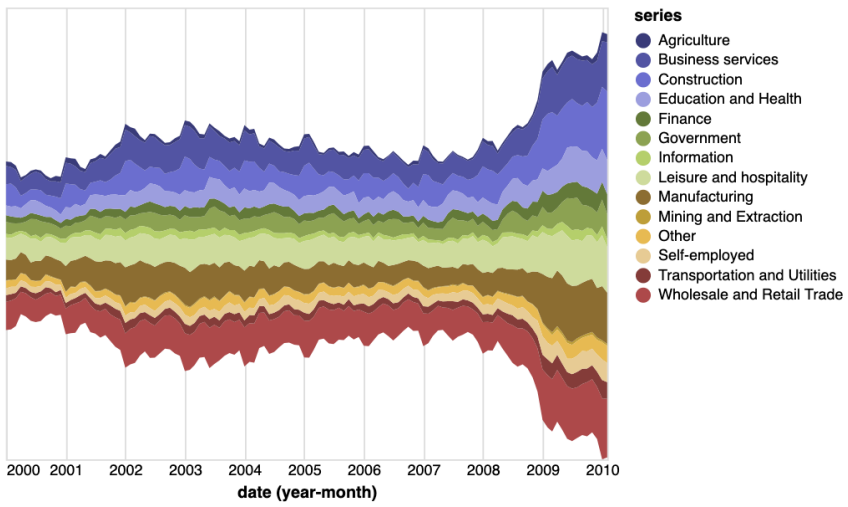
Being that we are not dealing with a background/foreground image, the figure-ground principle is not applicable here.

This visualization would've been stronger than the others, but also different. It would serves as a combination of showing the percentage of a specific emoji used across time and as a time series which combines the ideas of the the plots we created in this assignment.

Thank you for a great class. I really am happy to have gone beyond the usual libraries used and finally expanded my visualization knowledge and capabilities. I look forward to continuing my data visualization work throughout the program.

In [87]:
```
import altair as alt
from vega_datasets import data

source = data.unemployment_across_industries.url

selection = alt.selection_multi(fields=['series'], bind='legend')

alt.Chart(source).mark_area().encode(
    alt.X('yearmonth(date):T', axis=alt.Axis(domain=False, format='%Y', tickSize=0)),
    alt.Y('sum(count):Q', stack='center', axis=None),
    alt.Color('series:N', scale=alt.Scale(scheme='category20b')),
    opacity=alt.condition(selection, alt.value(1), alt.value(0.2))
).add_selection(
    selection
)
```

Out[87]:



In [88]:
```
#import altair as alt

#selection = alt.selection_multi(fields=['series'], bind='legend')

#annotations = [['2017-08-27 00:15:00',2.05, 'Fight begins'],
                    #['2017-08-27 00:39:00', 1.75, 'McGregor\nimpresses\nearly'],
                    #['2017-08-27 00:52:50', 0.25, 'Fight Ends']]

#a_df = pd.DataFrame(annotations, columns=['date','count','note'])

#alt.Chart(a_df).mark_area().encode(
    #alt.X('datetime:T', axis=alt.Axis(domain=False, format='%Y', tickSize=0)),
    #alt.Y('tweet_count:Q', stack='center', axis=None),
    #alt.Color('team:N', scale=alt.Scale(scheme='category20b')),
    #opacity=alt.condition(selection, alt.value(1), alt.value(0.2))
#).add_selection(
    #selection
#)
```

In [ ]: