Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menubar, select Kernel→Restart) and then **run all cells** (in the menubar, select Cell →Run All).

Make sure you fill in any place that says `YOUR CODE HERE` or "YOUR ANSWER HERE", as well as your name and collaborators below:

In [1]:
```
NAME = "Cody Lange"
COLLABORATORS = ""
```

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)

A note about autograding

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.

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:

Section deleted

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

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. (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 (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 [2]:
```
# start with the setup
import pandas as pd
import altair as alt
import numpy as np
```

In [3]:
```
# enable correct rendering
alt.renderers.enable('default')
```

Out[3]:
```
RendererRegistry.enable('default')
```

In [4]:
```
# uses intermediate json files to speed things up
alt.data_transformers.enable('json')
```

Out[4]:
```
DataTransformerRegistry.enable('json')
```

In [5]:
```
# 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 [6]:
```
# uncomment to see what's inside
tweets.head()
```

Out[6]:

	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	0	0	0	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	0	0	0	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	0	0	0	0	2	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	0	0	0	0	1	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	2	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/)

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

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

Get the data on GitHub (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.

** 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 [7]:
```
# 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') + ' %'
percentages_df['PERCENT'] = percentages_df.PERCENT /100
```

In [8]:
```
# uncomment to see what's inside
percentages_df
```

Out[8]:

	EMOJI	PERCENT	rank	PERCENT_TEXT
0	😂	0.231	1	23.1 %
1	💩	0.057	2	5.7 %
2	😩	0.035	3	3.5 %
3	😳	0.030	4	3.0 %
4	💪	0.025	5	2.5 %
5	IE	0.024	6	2.4 %
6	😭	0.023	7	2.3 %
7	🔥	0.023	8	2.3 %
8	😫	0.020	9	2.0 %
9	💰	0.018	10	1.8 %

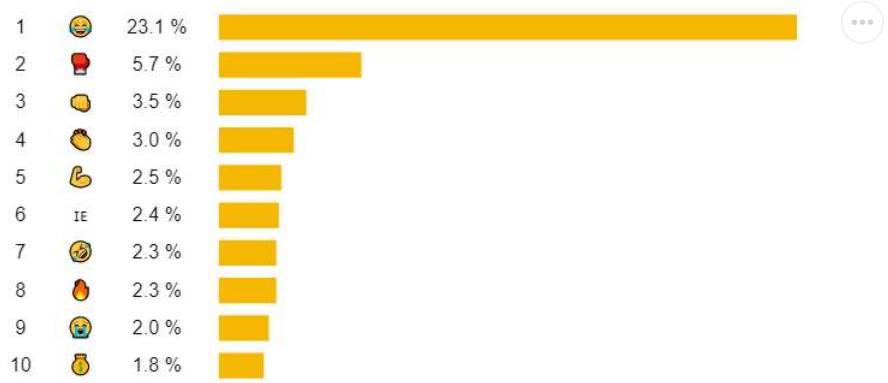
** Homework note, construct your solution to this chart in the cell below. Click here (assets/emoji_distrib_altair.png) to see a sample output from Altair.

2.1 use percentages_df to recreate the visualization above

In [9]:

```
# use percentages_df to recreate the visualization above

bars = alt.Chart(percentages_df).mark_bar(size=17, color='#f6b700').encode(
    # encode x as the percent, and hide the axis
    x=alt.X(
        'PERCENT:Q',
        axis=None,
    ),
    y=alt.Y(
        'EMOJI:N',
         axis=alt.Axis(ticks=False, labels=False, title='', domain=False),
        sort=None
    )
).properties(
    width=400,
    height=250
)

#Create Ranking Column
rankings = alt.Chart(percentages_df).mark_text(size=13).encode(
    y=alt.Y('EMOJI:N', axis=None, sort = None
),
    text=alt.Text('rank:Q'),
).properties(width=20, height = 250)

#Create Emoji Column
emojis = alt.Chart(percentages_df).mark_text(size=13).encode(
    y=alt.Y('EMOJI:N', axis=None, sort = None
),
    text=alt.Text('EMOJI:N'),
).properties(
    width=20,
    height=250
)

percs = alt.Chart(percentages_df).mark_text(size=13).encode(
    y=alt.Y('EMOJI:N', axis=None, sort = None
),
    text=alt.Text('PERCENT_TEXT:O'),
).properties(
    width=20,
    height=250
)

chart21 = alt.hconcat(rankings, emojis, percs, bars)
chart21.configure_view(
    # we don't want a stroke around the bars
    strokeWidth=0
).configure_scale(
    # add some padding
    bandPaddingInner=0.2
)
```

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.

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


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) | (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 [11]:
```
# uncomment to see what's inside
ndf.sample(5)
```

Out[11]:

	datetime	tweet_count	team
756	2017-08-27 01:09:51	0.326087	🏵☘🇮🇪
10	2017-08-27 00:05:56	2.545455	🏵☘🇮🇪
740	2017-08-27 01:08:31	2.625000	🏈😬💰🇬🇧
138	2017-08-27 00:12:22	0.740260	🏈😬💰🇬🇧
353	2017-08-27 00:31:56	0.666667	🏈😬💰🇬🇧

In [12]:
```
# 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 [13]:
```
# uncomment to see what's inside
a_df
```

Out[13]:

	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

** Homework note, construct your solution to this chart in the cell below. Click here (assets/altair_chart2.png) to see a sample output from Altair.

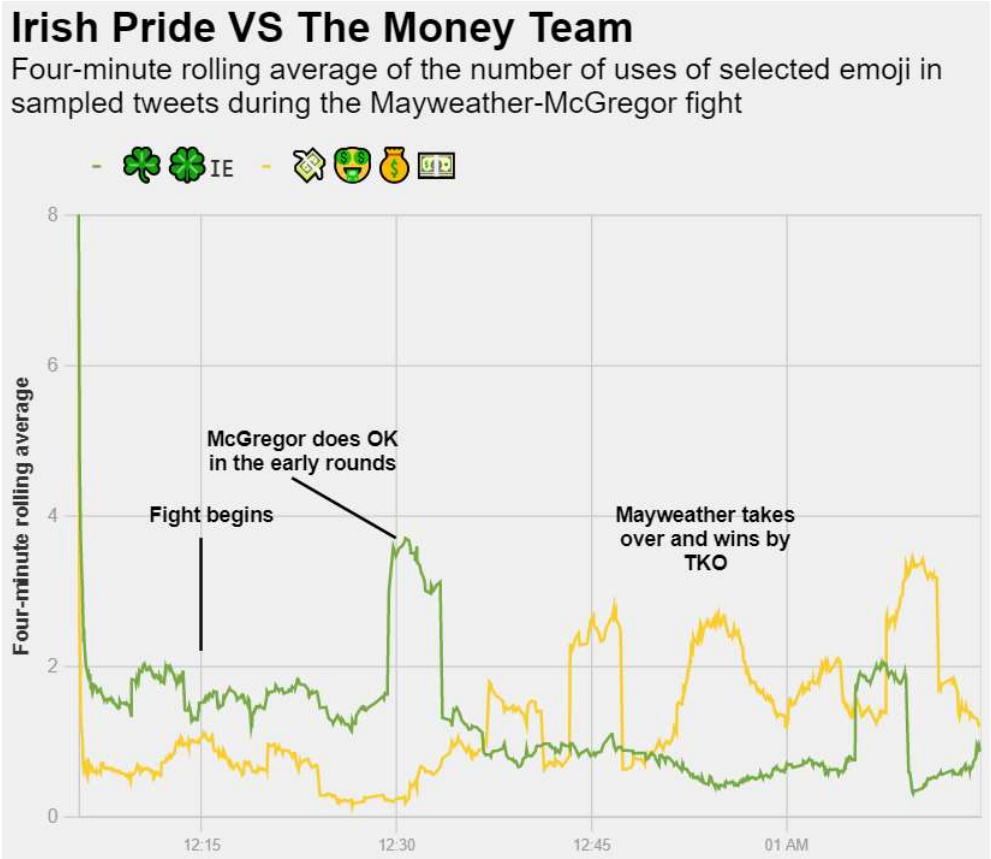
2.2 your turn, create your solution

In [14]:

```
from altair import datum #Needed for subsetting (transforming data)

alt.themes.enable('fivethirtyeight') #Turns out 538 has their own Altair theme

chart = 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-minute rolling average',
        axis=alt.Axis(tickCount=4, labelFontSize =13)
    ),
    color=alt.Color(
          'team:N',
           scale = alt.Scale(domain=['🏵🏵IE','🥊😳💰🇲🇽'], range=['#78AB45', '#FBCE30']),
           legend = alt.Legend(
               orient='top',
               title=None,
               symbolType='stroke',
               labelFontSize= 22
           )
    )
).properties(
    # set the dimensions of the visualization
    width=600,
    height=400,
    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 figh
           "fontSize":28,
           'subtitleFontSize': 20,
           'anchor': 'start'
           },
)

#Create the annotations
notes = alt.Chart(a_df).mark_text(
    baseline='middle',
    fontSize = 14,
    fontWeight='bold',
    lineBreak='\n',
    dx = 7
).encode(
    x='date:T',
    y='count:Q',
    text='note:N'
)

#Create lines pointing to annotations
df = 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': [2.2, 3.7, 4.5, 3.7],
    'class': ['A', 'A', 'B', 'B']
})
```

```
lines = alt.Chart(df).mark_line(color='black').encode(
    x='x:T',
    y='y',
    detail='class'
)

(chart + notes + lines)
```

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 (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.

** Homework note, construct your solution to this chart in the cell below. Click here (assets/altair_chart3.png) to see a sample output from Altair.

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

In [15]:
```
# your solution goes here, use the example above for the sampling and annotation

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

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

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

fsdf = pd.concat([fdf,sdf])
```

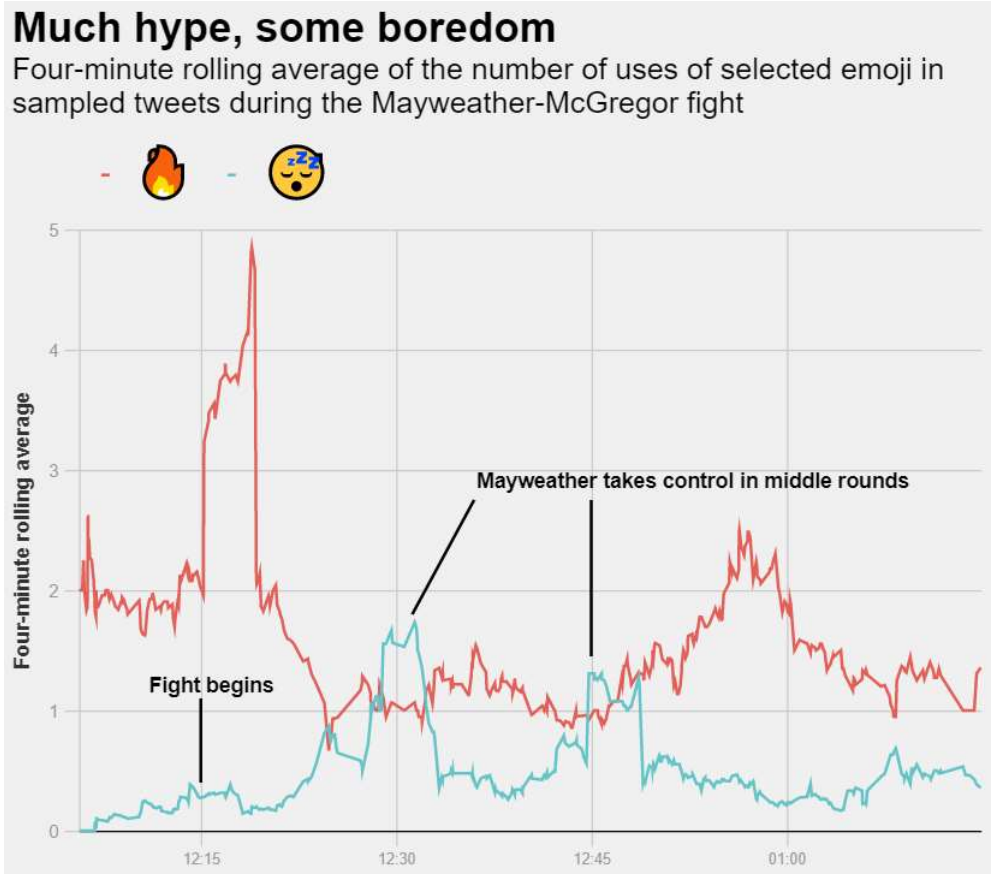
In [16]:
```
#Create Annotations df
annotations23 = [['2017-08-27 00:15:00',1.2, 'Fight begins'],
                 ['2017-08-27 00:52:00',2.9, 'Mayweather takes control in middle rounds']]
a_df23 = pd.DataFrame(annotations23, columns=['date','count','note'])
```

In [17]:

```
t23 = alt.Chart(fsdf).mark_line().encode(
x=alt.X(
    'datetime:T',
    axis=alt.Axis(tickCount=4, domainColor='black', format='%I:%M'),
    title='',
    ),
y=alt.Y(
    'tweet_count:Q',
    title='Four-minute rolling average',
    axis=alt.Axis(tickCount=5, labelFontSize =12)
),
color=alt.Color(
      'emoji:N',
       scale = alt.Scale(domain=['🔥','😴'], range=['#e35f57', '#64c4c7']),
       legend = alt.Legend(
           orient='top',
           title=None,
           symbolType='stroke',
           labelFontSize= 33
       )
)
operties(
# set the dimensions of the visualization
width=600,
eight=400,
itle={"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'],
       "fontSize":28,
       'subtitleFontSize': 20,
       'anchor': 'start'
       },


ate the annotations
s23 = alt.Chart(a_df23).mark_text(
aseline='middle',
ontSize = 14,
ontWeight='bold',
ineBreak='\n',
dx = 7
code(
x='date:T',
y='count',
ext='note:N'


ate lines pointing to annotations
 pd.DataFrame({
'x': ['2017-08-27 00:15:00', '2017-08-27 00:15:00', '2017-08-27 00:31:15', '2017-08-27 00:36:00', '2017-08-27 00:45:00', '2017-08-27 00:45:00'],
'y': [1.1, .4, 1.8, 2.75, 2.75, 1.45],
'class': ['A', 'A', 'B', 'B', 'C', 'C']


s23 = alt.Chart(df).mark_line(color='black').encode(
x='x:T',
y='y',
detail='class'
```

```
rt23 + notes23 + lines23).configure_view(
strokeWidth=0
```

Out[17]:



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 😫😫😫😫.

** Homework note, construct your solution to this chart in the cell below. Click here (assets/altair_chart4.png) to see a sample output from Altair.

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

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

cry_laugh = tweets.copy()
cry_laugh = cry_laugh.resample('1s').sum()

cry_laugh  = cry_laugh[(cry_laugh['😭']>0) | (cry_laugh['😂']>0)]

# next we're going to creat a rolling average
# first for cry usage
cdf = cry_laugh['😭'].rolling('4Min').mean().reset_index()
cdf['emoji'] = '😭'
cdf = cdf.rename(columns={'😭':'tweet_count'})

# next for the laugh usage
ldf = cry_laugh['😂'].rolling('4Min').mean().reset_index()
ldf['emoji'] = '😂'
ldf = ldf.rename(columns={'😂':'tweet_count'})


cldf = pd.concat([cdf,ldf])
```

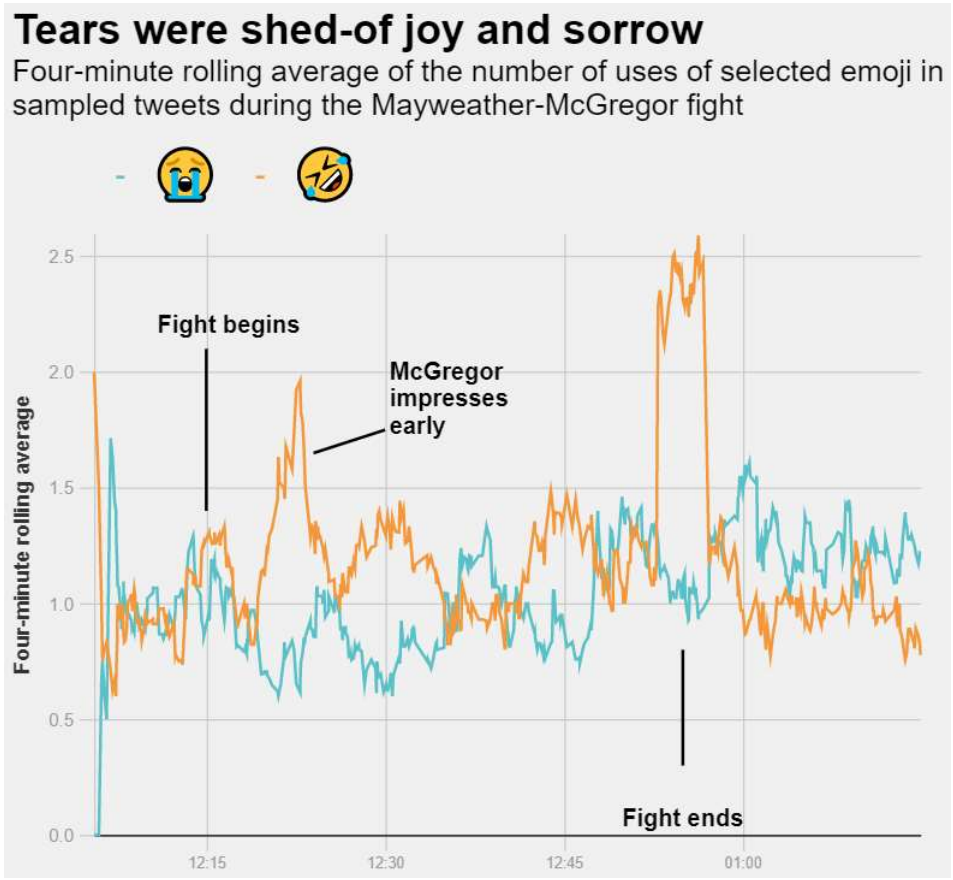
In [19]:
```
annotations24 = [['2017-08-27 00:10:00',2.2, 'Fight begins'],
                 ['2017-08-27 00:29:30',2., 'McGregor\nimpresses\nearly'],
                 ['2017-08-27 00:49:00',0.07, 'Fight ends']]
a_df24 = pd.DataFrame(annotations24, columns=['date','count','note'])
```

In [20]:

```
chart24 = alt.Chart(cldf).mark_line().encode(
    x=alt.X(
        'datetime:T',
        axis=alt.Axis(tickCount=4, domainColor='black', format='%I:%M'),
        title='',
        ),
    y=alt.Y(
        'tweet_count',
        title='Four-minute rolling average',
        axis=alt.Axis(tickCount=5, labelFontSize =12)
    ),
    color=alt.Color(
          'emoji',
           scale = alt.Scale(domain=['😭','😂'], range=['#5bc1c8', '#f5983b']),
           legend = alt.Legend(
               orient='top',
               title=None,
               symbolType='stroke',
               labelFontSize= 33
           )
    )
).properties(
    # set the dimensions of the visualization
    width=550,
    height=400,
    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 figh
           "fontSize":28,
           'subtitleFontSize': 20,
           'anchor': 'start'
           },
)

#Create the annotations
notes24 = alt.Chart(a_df24).mark_text(
    baseline='middle',
    fontSize = 16,
    fontWeight='bold',
    lineBreak='\n',
    dx = 7,
    align='left'
).encode(
    x='date:T',
    y='count',
    text='note:N'
)

#Create lines pointing to annotations
df = pd.DataFrame({
    'x': ['2017-08-27 00:15:00', '2017-08-27 00:15:00', '2017-08-27 00:24:00', '2017-08-27 00:30:00', '2017-08-27 00:55:00', '2017-08-27 00:55:00'],
    'y': [2.1, 1.4, 1.65, 1.75, 0.3, 0.8],
    'class': ['A', 'A', 'B', 'B', 'C', 'C']
})

lines24 = alt.Chart(df).mark_line(color='black').encode(
    x='x:T',
    y='y',
```

```
    detail='class'
)

(chart24 + notes24 + lines24).configure_view(
    strokeWidth=0
)
```

Out[20]:



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

2.5 Make your own (part 1-alternative)

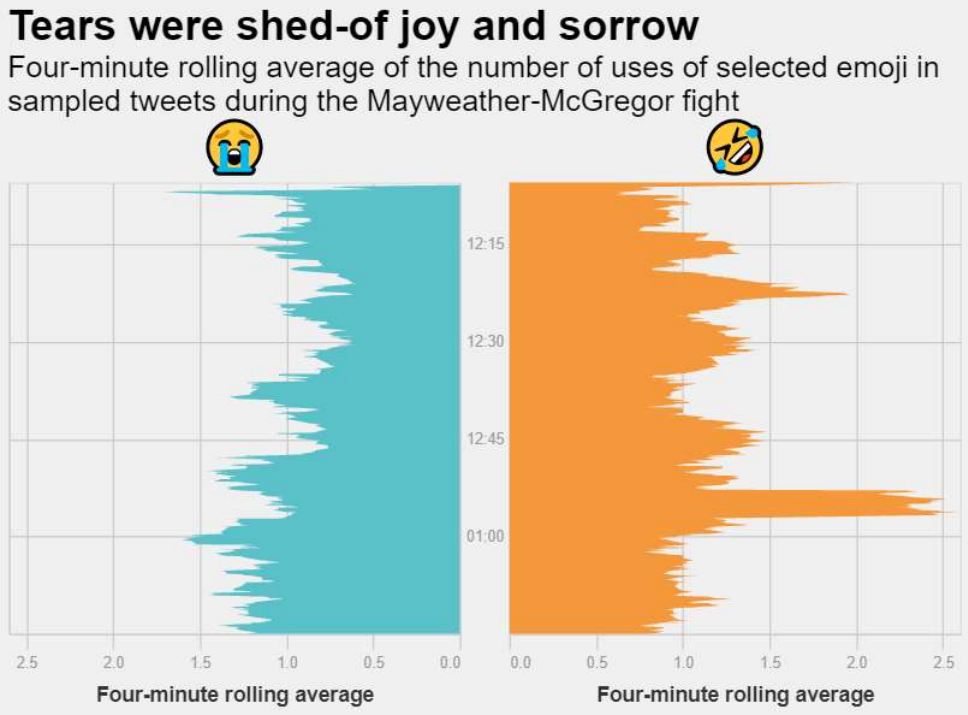
Propose one *alternative* visualization for one of the article's visualizations. Add a short paragraph describing why your visualization is more *effective* based on principles of perception/cognition. If you feel your visualization is worse, that's ok! Just tell us why. (20 points/ 15 points plot +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

In [21]:

```
alt.themes.enable('fivethirtyeight') #Turns out 538 has their own Altair theme

cry = alt.Chart(cldf).mark_area(color='#5bc1c8', size=5).transform_filter(
    alt.datum.emoji == '😭'
).encode(
    y= alt.Y(
        'datetime:T',
        sort='descending',
        axis=alt.Axis(
            labels=False,
            title='',
            ticks=False,
            format='%I:%M'
            )
    ),
    x=alt.X('tweet_count:Q',
            axis=alt.Axis(title='Four-minute rolling average'),
            sort=alt.SortOrder('descending'),
            scale=alt.Scale(domain=(0, 2.5)),
    )
).properties(
    width = 300,
    height= 300,
    title='😭',
)


laugh = alt.Chart(cldf).mark_area(color='#f5983b', size=5).transform_filter(
    alt.datum.emoji == '😂'
).encode(
    y= alt.Y(
        'datetime:T',
        sort='descending',
        title='',
        axis=alt.Axis(
            title='',
            ticks=False,
            format='%I:%M'
            )
    ),
    x=alt.X('tweet_count:Q',
            axis=alt.Axis(title='Four-minute rolling average'),
            sort=alt.SortOrder('ascending'),
            scale=alt.Scale(domain=(0, 2.5)),
           )
).properties(
    width = 300,
    height= 300,
    title= '😂'
)


chart25 = alt.hconcat(cry, laugh,spacing=0).resolve_scale(y='shared'
).configure_title(
    fontSize=33, offset=5, orient='top', anchor='middle'
).properties(
  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 figh
            "fontSize":28,
            'subtitleFontSize': 20,
            'anchor': 'start'
            }
).resolve_scale(y='shared')

chart25
```

Out[21]:



2.6 Make your own (part 2-novel)

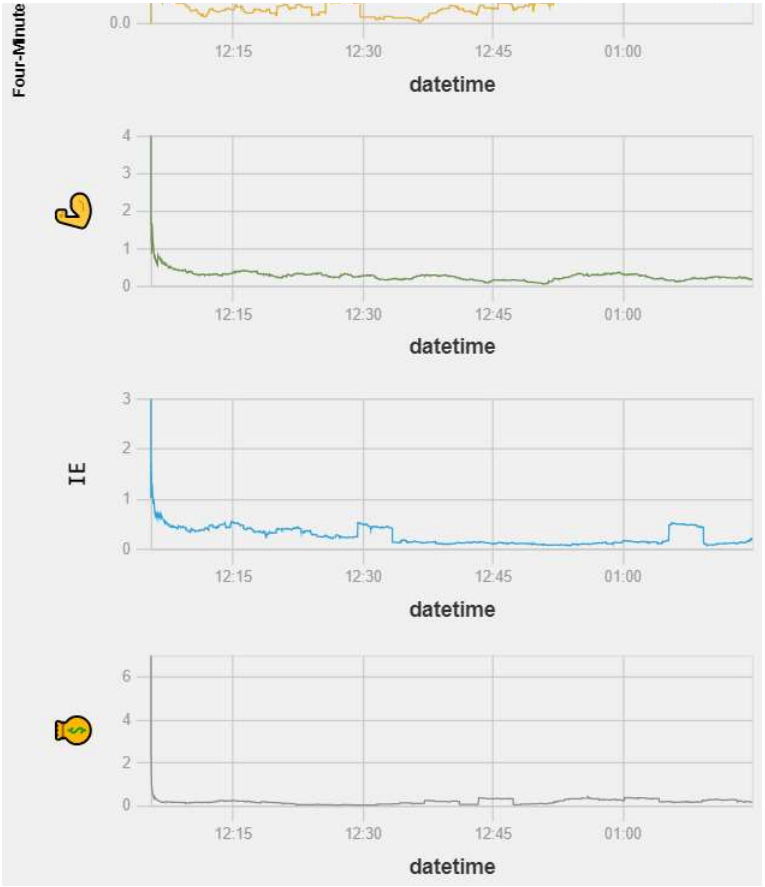
Propose a *new* visualization to complement a part of the article. Add a short paragraph justifying your decisions in terms of Perception/Cognition processes. If you feel your visualization is worse, that's ok! Just tell us why. (20 points/ 15 points plot + 5 justification)

In [22]:

```
others = tweets.copy()
others = others.resample('1s').sum()

others  = others[(others['😂']>0) | (others['👊']>0) | (others['👊']>0) | (others['👏']>0) | (others['💪']>0) | (others['🇮🇪']>0) | (others['💰']>0)]

# next we're going to create a rolling average
# first for horizontal laugh usage
hldf = others['😂'].rolling('4Min').mean().reset_index()
hldf['emoji'] = '😂'
hldf = hldf.rename(columns={'😂':'tweet_count'})

# next for the upward punch usage
updf = others['👊'].rolling('4Min').mean().reset_index()
updf['emoji'] = '👊'
updf = updf.rename(columns={'👊':'tweet_count'})

# next for horizontal punch usage
hpdf = others['👊'].rolling('4Min').mean().reset_index()
hpdf['emoji'] = '👊'
hpdf = hpdf.rename(columns={'👊':'tweet_count'})

# next for the clap usage
clapdf = others['👏'].rolling('4Min').mean().reset_index()
clapdf['emoji'] = '👏'
clapdf = clapdf.rename(columns={'👏':'tweet_count'})

# next for the arm usage
armdf = others['💪'].rolling('4Min').mean().reset_index()
armdf['emoji'] = '💪'
armdf = armdf.rename(columns={'💪':'tweet_count'})

# next for Irish Flag usage
irishdf = others['🇮🇪'].rolling('4Min').mean().reset_index()
irishdf['emoji'] = '🇮🇪'
irishdf = irishdf.rename(columns={'🇮🇪':'tweet_count'})

# next for the bag usage
bagdf = others['💰'].rolling('4Min').mean().reset_index()
bagdf['emoji'] = '💰'
bagdf = bagdf.rename(columns={'💰':'tweet_count'})

allothers = pd.concat([hldf,updf, hpdf, clapdf, armdf, irishdf, bagdf])


alt.Chart(allothers).mark_line(size=1).encode(
    x=alt.X(
        'datetime:T',
        axis=alt.Axis(tickCount=4, format='%I:%M')
    ),
     y=alt.Y(
        'tweet_count:Q',
         axis=alt.Axis(
             title=''),
           ),
    color=alt.Color('emoji', legend =None),
    row=alt.Row(
        'emoji:N',
```

```
         sort=['😂','🥊', '👊', '👏', '💪', 'IE', '💰'],
        header=alt.Header(labelFontSize=22),
        title='Four-Minute Rolling Average'
    )
).properties(
    height=100,
    width=400,
    title={"text": "The other emojis?",
           'subtitle' : ['Four-minute rolling average of the number of uses of', 'the other top 10 emojis not already in a chart'],
           "fontSize":28,
           'subtitleFontSize': 18,
           'anchor': 'start'
           }
).resolve_scale(y='independent', x='independent')
```

Out[22]:


Four-Minute
0.0
12:15
12:30
12:45
01:00
datetime
4
3
2
1
0
12:15
12:30
12:45
01:00
datetime
IE
3
2
1
0
12:15
12:30
12:45
01:00
datetime
6
4
2
0
12:15
12:30
12:45
01:00
datetime

In [ ]: