assignment4

March 3, 2020

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:

[1]:
```
NAME = "Aseem Sachdeva"
```

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



from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

```
<IPython.core.display.HTML object>
```

---

1 Information Visualization I

1.1 School of Information, University of Michigan

1.2 Week 4:

- Data Types
- Design

1.3 Assignment Overview

1.3.1 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

1.3.2 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)

1.3.3 Resources:

- Article by Five Thirty Eight available online (Hickey, Koeze, Dottle, Wezerek 2017)

- Datasets from Five Thirty Eight, we have download a subset of these datasets to ./assets but the original can be found on Five Thirty Eight Mayweather vs McGregor

1.4 Part 1. Data Types & Design (30 points)

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

1.4.1 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)

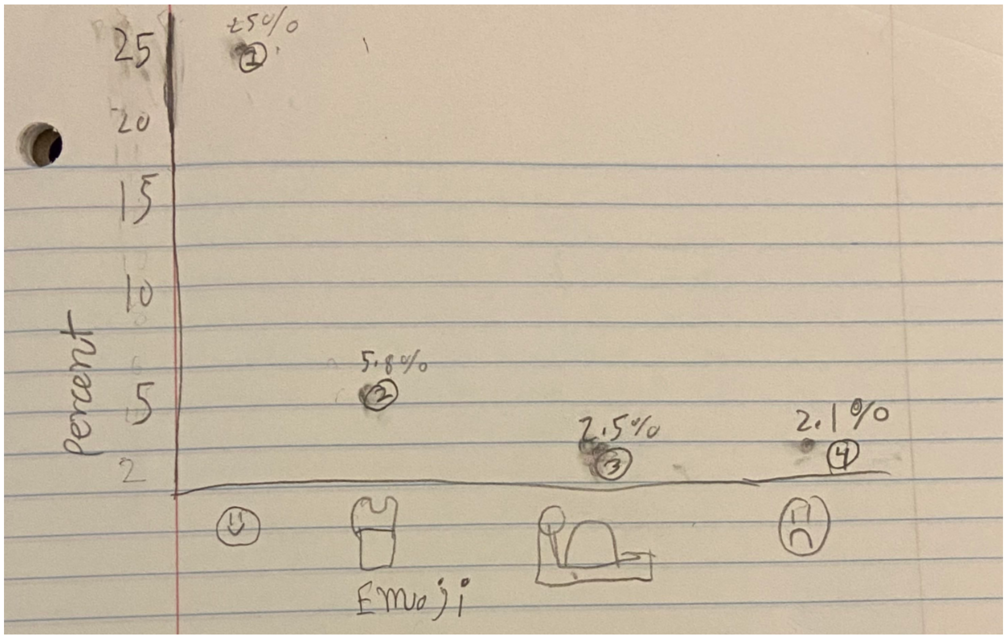
For the two visualizations, the variables included, and their corresponding data types and encodings, are as follows:

Top visualization These are the minimum number of variables needed to reconstruct the top visualization: 1. Type of emoji, which is a qualitative variable, encoded using the emoji text 2. The percentage of 240,000 tweets that contained each respective emoji, which is a quantitative variable, encoded numerically(percentage), and by bar length. 3. The ranking of each emoji's frequency, which is a qualitative variable, encoded numerically in the index

Bottom visualization These are the minimum number of variables needed to reconstruct the top visualization 1. Time of observation(timestamp), which is a quantitative variable, encoded numerically on the x axis 2. The tweet count for each emoji grouping(one group for Mayweather and one group for McGregor), which is a quantitative variable, encoded by the height of each line graph 3. What 'team' each tweet belonged to, which is a qualitative variable, encoded by the color of each line graph(green for McGregor and yellow for Mayweather)

1.4.2 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.



answer1.2

On account of the manageable number of variables and encodings - namely, emoji, percentage, and ranking - it is more than feasible to refactor the horizontal bar chart pictured above as a circle chart. The nominal variable 'emoji' can be encoded on the x-axis, while percentage can be encoded on the y-axis and in label form above/adjacent to each point, while rank can be encoded by overlaying text over each circle and in the position of each emoji on the x-axis. Converting the bar chart to a circle chart with this likeness would not be less expressive than 538's bar chart - for, all of the necessary facts in the dataset needed to convey the desired message have been represented, and only these facts. Nor would this alteration be less effective, since the information conveyed here is not less readily perceived than the information conveyed in 538's chart. In regards to Tufte's data-ink ratio, it could be argued that the quotient obtained from the calculation(data ink ratio = data ink/total ink used to render the entire graphic) will decrease with the circle chart approach, since the ink utilized to represent the data itself will naturally decrease(circles use less ink than filled bars), and thereby the total ink used to render the entire graphic will also decrease. Superfluous chart junk is not present in either approach, and proportionality is not a great concern with the circle chart approach, since the sizes of each point are uniform, though it is easier to create bars with erroneous proportions with the horizontal bar plot approach. All this being said, the clear and concise labeling possible with both approaches can make up for any potential graphical distortion, of which there could be a potential for depending on the visualization framework being utilized.

1.4.3 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

Although the above visualization is eye catching and memorable, it does suffer from some excess chart junk. The two images of Muhammad Ali could have been removed, for instance, in addition to the image of his downed opponent, the boxing gloves in the upper right corner, and the boxing gloves surrounding Ali's victory/loss totals - though the numerical text can retain the color encoding, so the viewer can differentiate between wins/losses. Indeed, the viewer will almost certainly be able to infer that the chart pertains to the boxer Muhammad Ali on account of the visualization's title, making these extra images - though striking - somewhat superfluous. Once these items have been removed, the visualization will be in a state where all of the components are the minimum number necessary in order communicate what it is intending to in a manner that is still understandable.

All of this being said, it is possible to refactor the visualization entirely, and may even be the prudent choice, since the quasi-pie chart nature of the visualization makes the numerous labels difficult to read. Instead, a stacked bar chart can be utilized, with year on the y-axis, and the number of wins/losses Ali experienced for a certain year stacked on top of each other. The names of Ali's opponents can be overlayed each bar as text, and the color encoding for wins/losses can be retained. The legend that displays the type of win/loss can be retained as well, within the whitespace next to 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. This article is based on the dataset:

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

To earn points for this assignment, you must:

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

1.4.4 Before you begin

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

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

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

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

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

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

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

[5]:
```python
# 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['']
```

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

[6]:
```
            created_at  emojis                  id  \
0  2017-08-27 00:05:34       1  901656910939770881
1  2017-08-27 00:05:35       5  901656917281574912
2  2017-08-27 00:05:35       2  901656917105369088
3  2017-08-27 00:05:35       2  901656917747142657
4  2017-08-27 00:05:35       2  901656916828594177

                                              link  retweeted     screen_name  \
0  https://twitter.com/statuses/901656910939770881      False         aaLiysr
1  https://twitter.com/statuses/901656917281574912      False  zulmafrancozaf
2  https://twitter.com/statuses/901656917105369088      False      Adriana11D
3  https://twitter.com/statuses/901656917747142657      False    Nathan_Caro_
4  https://twitter.com/statuses/901656916828594177      False      sahouraxox

                                                 text                \
0  Ringe çkmadan ate etmeye balad #McGregor ...   0   0  0  0  0  0  0
1         @lalylourbet2 https://t.co/ERUGHhQINE   0   0  0  0  0  0  0
2                      #MayweathervMcgregor   0   3  0  0  0  0  0
3                Cest partit #MayweatherMcGregor    0   0  0  0  0  0  0
4  Low key feeling bad for ppl who payed to watch...   0   0  0  0  0  0  0

                     irish_pride  money_team
0  0  0  0  0  0  0  0  0  0            0           0
1  0  0  0  0  0  0  0  0  0            0           0
2  0  0  0  0  0  0  2  0  0            3           0
3  0  0  0  0  0  0  1  0  0            0           0
4  0  0  2  0  0  0  0  0  0            0           0
```

2 The Mayweather-McGregor Fight, As Told Through Emojis

We laughed, cried and cried some more.

Original article available at FiveThirtyEight
By Dhrumil Mehta, Oliver Roeder and Rachael Dottle
Filed under Mayweather vs. McGregor
Get the data on GitHub

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

How do we know all this? Emojis.

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

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

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

[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') + ' %'
```

[8]:
```
# uncomment to see what's inside
percentages_df
percentages_df.index = percentages_df.index + 1
percentages_df
```

[8]:
```
   EMOJI  PERCENT  rank PERCENT_TEXT
1             23.1     1       23.1 %
```

```
2             5.7     2         5.7 %
3             3.5     3         3.5 %
4             3.0     4         3.0 %
5             2.5     5         2.5 %
6            2.4     6         2.4 %
7             2.3     7         2.3 %
8             2.3     8         2.3 %
9             2.0     9         2.0 %
10            1.8     10        1.8 %
```

[9]:
```
add_space = lambda x: str(x) + " "
percentages_df['concat'] = percentages_df['EMOJI'].map(add_space) +  " " +␣
 ↪percentages_df['PERCENT_TEXT']
percentages_df.head()
```

[9]:
```
  EMOJI  PERCENT  rank PERCENT_TEXT     concat
1          23.1     1       23.1 %    23.1 %
2           5.7     2        5.7 %     5.7 %
3           3.5     3        3.5 %     3.5 %
4           3.0     4        3.0 %     3.0 %
5           2.5     5        2.5 %     2.5 %
```

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

[10]:
```
#Let's create a copy of the original dataframe to avoid manipulating it, and␣
 ↪create a sort order to match 538
percentages_copy = percentages_df
rank = ['', '', '', '', '', '', '', '', '', '' ]



#Now lets create the bars
bars = alt.Chart(percentages_copy).mark_bar(size=17).encode(
    # encode x as the percent, and hide the axis
    x=alt.X(
        'PERCENT',
        axis=None),
    y=alt.Y(
        # encode y using the name, use the movie name to label the axis, sort␣
 ↪using the names_l
        'concat:N',
         axis=alt.Axis(tickCount=0, title=''),
         # we give the sorting order to avoid alphabetical order
        sort = rank
    )
)

# at this point we don't really have a great plot (it's missing the␣
 ↪annotations, titles, etc.)
```

```
bars



#raise NotImplementedError()
```

[10]: alt.Chart(...)

[11]:
```
#Now lets style the text and combine the bars and the text




# finally, we're going to combine the bars and the text and do some styling
emojis = ( bars).configure_mark(
    # we don't love the blue
    color='#FFA500'
).configure_view(
    # we don't want a stroke around the bars
    strokeWidth=0
).configure_scale(
    # add some padding
    bandPaddingInner=0.2
).properties(
    # set the dimensions of the visualization
    width=500,
    height=200
).properties(

)

emojis
```

[11]: alt.Chart(...)

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

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

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

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

[12]:
```
# 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')

# next we're going to creat a rolling average
# first for the money team
mdf = (tweets['money_team'].rolling('4Min').mean().groupby(pd.
 ↪Grouper(freq='15S')).mean() * 15).reset_index()
mdf['team'] = ''
mdf = mdf.rename(columns={'money_team':'tweet_count'})

# next for the irish team
idf = (tweets['irish_pride'].rolling('4Min').mean().groupby(pd.
 ↪Grouper(freq='15S')).mean() * 15).reset_index()
idf['team'] = ''
idf = idf.rename(columns={'irish_pride':'tweet_count'})

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

[13]:
```
# uncomment to see what's inside
ndf.sample(5)
```

[13]:
```
               datetime  tweet_count  team
177 2017-08-27 00:49:45     0.432232
10  2017-08-27 00:08:00     0.870714
37  2017-08-27 00:14:45     2.437985
228 2017-08-27 01:02:30     0.712231
89  2017-08-27 00:27:45     1.223707
```

[14]:
```
# we're also going to create an annotations data frame to help you
annotations = [['2017-08-27 00:15:00',3, 'Fight begins'],
               ['2017-08-27 00:20:00',3.5, 'McGregor does OK in the early␣
 ↪rounds'],
               ['2017-08-27 00:40:00',2.8, 'Mayweather takes over and wins by␣
 ↪TKO']]
a_df = pd.DataFrame(annotations, columns=['date','count','note'])
```

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

a_df_1 = a_df.loc[a_df['date'] == '2017-08-27 00:15:00']

a_df_2 =  a_df.loc[a_df['date'] == '2017-08-27 00:20:00']

a_df_3 =  a_df.loc[a_df['date'] == '2017-08-27 00:40:00']
```

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

[16]:
```
ndf_copy = ndf


#We'll create an hours and minutes column to aid in our filtering done below
ndf_copy['hours and minutes'] = ndf_copy['datetime'].dt.strftime("%I:%M")


ndf_copy.head()



ndf_copy_1215 = ndf_copy.loc[ndf_copy['hours and minutes'] == '12:15']

ndf_copy_1230 = ndf_copy.loc[ndf_copy['hours and minutes'] == '12:30']

ndf_copy_1245 = ndf_copy.loc[ndf_copy['hours and minutes'] == '12:45']

ndf_copy_1 = ndf_copy.loc[ndf_copy['hours and minutes'] == '01:00']


##We can utilize the datetime columns in the following filtered dataframes to␣
↪find the values to specify for our tickmarks for our values parameter below,␣
↪which will place the x-axis tickmarks at those locations

#ndf_copy_1215.head()

#ndf_copy_1230.head()

#ndf_copy_1245.head()

#ndf_copy_1.head()
```

[17]:
```
#Tickmark placements(values) obtained from dataframes above

domain = ['', '']
range_ = ['#00902a', '#FFA500']




text = alt.Chart(a_df_1).mark_text(dx = 30, dy = -10).encode(
    x=alt.X('date:T'),
    y=alt.Y('count:Q'),
    text=alt.Text('note:N')
)
```

```
text_2 = alt.Chart(a_df_2).mark_text(dx = 100, dy = -50).encode(
    x=alt.X('date:T'),
    y=alt.Y('count:Q'),
    text=alt.Text('note:N')
)


text_3 = alt.Chart(a_df_3).mark_text(dx = 100, dy = -30).encode(
    x=alt.X('date:T'),
    y=alt.Y('count:Q'),
    text=alt.Text('note:N')
)


chart = alt.Chart(ndf_copy).mark_line().encode(
    x= alt.X('datetime',
              axis=alt.Axis(title='', values = ['2017-08-27 00:15:00',␣
 ↪'2017-08-27 00:30:00', '2017-08-27 00:45:00', '2017-08-27 01:00:00']),

            ),
    y=alt.Y('tweet_count',
            axis=alt.Axis(title = 'Four-minute rolling average', values =␣
 ↪[0,2,4,6,8])),
     color=alt.Color('team', legend=alt.Legend(orient='top', title =None ),␣
 ↪scale=alt.Scale(domain=domain, range=range_, ))


).properties(
    # add a title
     title={
      "text": ["Irish Pride VS The Money Team"],

    }
)


text + text_2 + text_3 + chart
```

[17]: alt.LayerChart(...)

While I was able to derive the correct scaling on both axes, the correct legend, the correct color encoding for each line, and was able to postion the labels correctly, I was unable to draw arrows from the labels to the relevant peaks in the lines, as in 538's visualization.

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

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

[18]:
```
tweets_copy = tweets

tweets_copy['snooze'] = tweets_copy['']

tweets_copy['fire'] = tweets_copy['']

tweets_copy.head()
```

[18]:
```
                              created_at  emojis                  id  \
datetime
2017-08-27 00:05:34  2017-08-27 00:05:34       1  901656910939770881
2017-08-27 00:05:35  2017-08-27 00:05:35       5  901656917281574912
2017-08-27 00:05:35  2017-08-27 00:05:35       2  901656917105369088
2017-08-27 00:05:35  2017-08-27 00:05:35       2  901656917747142657
2017-08-27 00:05:35  2017-08-27 00:05:35       2  901656916828594177

                                                                link  \
datetime
2017-08-27 00:05:34  https://twitter.com/statuses/901656910939770881
2017-08-27 00:05:35  https://twitter.com/statuses/901656917281574912
2017-08-27 00:05:35  https://twitter.com/statuses/901656917105369088
2017-08-27 00:05:35  https://twitter.com/statuses/901656917747142657
2017-08-27 00:05:35  https://twitter.com/statuses/901656916828594177

                     retweeted     screen_name  \
datetime
2017-08-27 00:05:34      False          aaLiysr
2017-08-27 00:05:35      False  zulmafrancozaf
2017-08-27 00:05:35      False       Adriana11D
2017-08-27 00:05:35      False     Nathan_Caro_
2017-08-27 00:05:35      False       sahouraxox

                                                                  text    \
datetime
2017-08-27 00:05:34  Ringe çkmadan ate etmeye balad #McGregor ...   0
2017-08-27 00:05:35         @lalylourbet2 https://t.co/ERUGHhQINE   0
2017-08-27 00:05:35                           #MayweathervMcgregor   0
2017-08-27 00:05:35                    Cest partit #MayweatherMcGregor    0
2017-08-27 00:05:35  Low key feeling bad for ppl who payed to watch...    0

                                                 \
datetime
2017-08-27 00:05:34   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
2017-08-27 00:05:35   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
2017-08-27 00:05:35   3  0  0  0  0  0  0  0  0  0  0  0  2  0  0
2017-08-27 00:05:35   0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
2017-08-27 00:05:35   0  0  0  0  0  0  0  0  2  0  0  0  0  0  0

                     irish_pride  money_team  snooze  fire
datetime
2017-08-27 00:05:34            0           0       0     0
2017-08-27 00:05:35            0           0       0     0
2017-08-27 00:05:35            3           0       0     0
2017-08-27 00:05:35            0           0       0     0
2017-08-27 00:05:35            0           0       0     0
```

[19]:
```
#Set up the data and create rolling average for each emoji just like the␣
 ↪previous problem


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


mdf = (tweets_copy['snooze'].rolling('4Min').mean().groupby(pd.
 ↪Grouper(freq='15S')).mean() * 15).reset_index()
mdf['sentiment'] = ''
mdf = mdf.rename(columns={'snooze':'tweet_count'})


idf = (tweets_copy['fire'].rolling('4Min').mean().groupby(pd.
 ↪Grouper(freq='15S')).mean() * 15).reset_index()
idf['sentiment'] = ''
idf = idf.rename(columns={'fire':'tweet_count'})


ndf = pd.concat([mdf,idf])
```

[20]:
```
ndf_copy = ndf
ndf.sample(5)
```

[20]:
```
               datetime  tweet_count sentiment
255 2017-08-27 01:09:15     0.513669
98  2017-08-27 00:30:00     0.443102
122 2017-08-27 00:36:00     0.882218
269 2017-08-27 01:12:45     0.565770
146 2017-08-27 00:42:00     0.284307
```

[21]:
```
#Create an annotation dataframe like the previous problem

annotations = [['2017-08-27 00:15:00',0.5, 'Fight begins'],
               ['2017-08-27 00:20:00',1.5, 'Mayweather takes control in middle␣
 ↪rounds']]

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

a_df

a_df_1 = a_df.loc[a_df['date'] == '2017-08-27 00:15:00']

a_df_2 =  a_df.loc[a_df['date'] == '2017-08-27 00:20:00']
```

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

[22]:
```
#Tickmark placements(values) obtained from dataframes above

domain = ['', '']
range_ = ['#FF0000', '#ADD8E6']




text = alt.Chart(a_df_1).mark_text(dx = 15).encode(
    x=alt.X('date:T'),
    y=alt.Y('count:Q'),
    text=alt.Text('note:N')
)



text_2 = alt.Chart(a_df_2).mark_text(dx = 100).encode(
    x=alt.X('date:T'),
    y=alt.Y('count:Q'),
    text=alt.Text('note:N')
)





chart = alt.Chart(ndf_copy).mark_line().encode(
    x= alt.X('datetime',
             axis=alt.Axis(title='', values = ['2017-08-27 00:15:00',␣
 ↪'2017-08-27 00:30:00', '2017-08-27 00:45:00', '2017-08-27 01:00:00']),
            ),
    y=alt.Y('tweet_count',
            axis=alt.Axis(title = 'Four-minute rolling average', values = [0,0.
↪5,1,1.5,2, 2.5, 3])),
     color=alt.Color('sentiment', legend=alt.Legend(orient='top', title =None␣
↪), scale=alt.Scale(domain=domain, range=range_, ))


).properties(

     title={
      "text": ["Much hype, some boredom"],

    }
)


text + text_2 + chart
```

[22]:
```
alt.LayerChart(...)
```

While I was able to derive the correct scaling on both axes, the correct legend, the correct color encoding for each line, and was able to postion the labels correctly, I was unable to draw arrows from the labels to the relevant peaks in the lines, as in 538's visualization.

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

[23]:
```
tweets_copy['tears'] = tweets_copy['']

tweets_copy['laughs'] = tweets_copy['']

tweets_copy.head()
```

[23]:
```
                               created_at  emojis                  id  \
datetime
2017-08-27 00:05:34  2017-08-27 00:05:34       1  901656910939770881
2017-08-27 00:05:35  2017-08-27 00:05:35       5  901656917281574912
2017-08-27 00:05:35  2017-08-27 00:05:35       2  901656917105369088
2017-08-27 00:05:35  2017-08-27 00:05:35       2  901656917747142657
2017-08-27 00:05:35  2017-08-27 00:05:35       2  901656916828594177

                                                        link  \
datetime
2017-08-27 00:05:34  https://twitter.com/statuses/901656910939770881
2017-08-27 00:05:35  https://twitter.com/statuses/901656917281574912
```

```
2017-08-27 00:05:35  https://twitter.com/statuses/901656917105369088
2017-08-27 00:05:35  https://twitter.com/statuses/901656917747142657
2017-08-27 00:05:35  https://twitter.com/statuses/901656916828594177

                     retweeted     screen_name  \
datetime
2017-08-27 00:05:34      False          aaLiysr
2017-08-27 00:05:35      False  zulmafrancozaf
2017-08-27 00:05:35      False       Adriana11D
2017-08-27 00:05:35      False     Nathan_Caro_
2017-08-27 00:05:35      False       sahouraxox

                                                                  text    \
datetime
2017-08-27 00:05:34  Ringe çkmadan ate etmeye balad #McGregor ...    0
2017-08-27 00:05:35         @lalylourbet2 https://t.co/ERUGHhQINE    0
2017-08-27 00:05:35                           #MayweathervMcgregor   0
2017-08-27 00:05:35                     Cest partit #MayweatherMcGregor    0
2017-08-27 00:05:35  Low key feeling bad for ppl who payed to watch...    0

                                                         \
datetime
2017-08-27 00:05:34  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
2017-08-27 00:05:35  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
2017-08-27 00:05:35  3  0  0  0  0  0  0  0  0  0  0  0  2  0  0
2017-08-27 00:05:35  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
2017-08-27 00:05:35  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0

                     irish_pride  money_team  snooze  fire  tears  laughs
datetime
2017-08-27 00:05:34            0           0       0     0      0       0
2017-08-27 00:05:35            0           0       0     0      0       0
2017-08-27 00:05:35            3           0       0     0      0       0
2017-08-27 00:05:35            0           0       0     0      0       0
2017-08-27 00:05:35            0           0       0     0      0       2
```

[24]:
```
#Set up the data and create rolling average for each emoji just like the␣
↪previous problem




mdf = (tweets_copy['laughs'].rolling('4Min').mean().groupby(pd.
↪Grouper(freq='15S')).mean() * 15).reset_index()
mdf['sentiment'] = ''
mdf = mdf.rename(columns={'laughs':'tweet_count'})
```

```
idf = (tweets_copy['tears'].rolling('4Min').mean().groupby(pd.
 ↪Grouper(freq='15S')).mean() * 15).reset_index()
idf['sentiment'] = ''
idf = idf.rename(columns={'tears':'tweet_count'})


ndf = pd.concat([mdf,idf])
```

[25]:
```
ndf_copy = ndf
ndf.sample(5)
```

[25]:
```
               datetime  tweet_count sentiment
129 2017-08-27 00:37:45     0.611223
216 2017-08-27 00:59:30     0.676355
159 2017-08-27 00:45:15     1.070801
202 2017-08-27 00:56:00     1.807654
111 2017-08-27 00:33:15     0.499741
```

[26]:
```
#Create an annotation dataframe like the previous problem

annotations = [['2017-08-27 00:15:00',0.5, 'Fight begins'],
               ['2017-08-27 00:25:00',1.5, 'McGregor impresses early']]

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

a_df

a_df_1 = a_df.loc[a_df['date'] == '2017-08-27 00:15:00']

a_df_2 =  a_df.loc[a_df['date'] == '2017-08-27 00:25:00']
```

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

[27]:
```
#Tickmark placements(values) obtained from dataframes above

domain = ['', '']
range_ = ['#ADD8E6', '#FFA500']





text = alt.Chart(a_df_1).mark_text(dx = 15, dy = -150).encode(
    x=alt.X('date:T'),
    y=alt.Y('count:Q'),
    text=alt.Text('note:N')
)
```

```
text_2 = alt.Chart(a_df_2).mark_text(dx = 100).encode(
    x=alt.X('date:T'),
    y=alt.Y('count:Q'),
    text=alt.Text('note:N')
)





chart = alt.Chart(ndf_copy).mark_line().encode(
    x= alt.X('datetime',
              axis=alt.Axis(title='', values = ['2017-08-27 00:15:00',␣
 ↪'2017-08-27 00:30:00', '2017-08-27 00:45:00', '2017-08-27 01:00:00']),

            ),
    y=alt.Y('tweet_count',
            axis=alt.Axis(title = 'Four-minute rolling average', values = [0,0.
 ↪5,1,1.5,2, 2.5])),
     color=alt.Color('sentiment', legend=alt.Legend(orient='top', title =None␣
 ↪), scale=alt.Scale(domain=domain, range=range_, ))


).properties(

     title={
      "text": ["Tears were shed - of joy and sorrow"],

    }
)


text + text_2 + chart
```

```
[27]: alt.LayerChart(...)
```

While I was able to derive the correct scaling on both axes, the correct legend, the correct color encoding for each line, and was able to postion the labels correctly, I was unable to draw arrows from the labels to the relevant peaks in the lines, as in 538's visualization.
They laughed. They cried. And they laughed some more. And they cried some more.

3 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. (20 points/ 15 points plot + 5 justification)

[28]:
```
percentages_df
```

[28]:
```
   EMOJI  PERCENT  rank PERCENT_TEXT     concat
1           23.1     1       23.1 %    23.1 %
2            5.7     2        5.7 %     5.7 %
3            3.5     3        3.5 %     3.5 %
4            3.0     4        3.0 %     3.0 %
5            2.5     5        2.5 %     2.5 %
6           2.4     6        2.4 %    2.4 %
7            2.3     7        2.3 %     2.3 %
8            2.3     8        2.3 %     2.3 %
9            2.0     9        2.0 %     2.0 %
10           1.8    10        1.8 %     1.8 %
```

[29]:
```
sort_ = ['', '', '', '', '', '', '', '', '', '']


points = alt.Chart(percentages_df).mark_point().encode(
    x=alt.X('EMOJI', sort = sort_),
    y=alt.Y('PERCENT:Q')
).properties(width= 500)

text = points.mark_text(
    align='left',
    baseline='middle',
    dx=7
).encode(
    text='rank'
)

points + text
```

[29]:
```
alt.LayerChart(...)
```

For my proposed alternative, I attempted to emulate my hand-drawn sketch to the best of my ability using altair, as an alternative to the very first visualization we created(538's horizontal bar chart). I believe that this approach would be a perfectly viable alternative to 538's chart, and may arguably be a superior one, on account of the fact that the three variables are encoded more naturally, instead of in a forced tabular form. The ranking of each emoji is annotated next to each point, and is also implicit in the ordering of each emoji on the x-axis, while percentage is encoded on the y-axis. In regards to Tufte's data-in ratio, less ink is undoubtedly used to represent the data using this approach, and thereby less ink overall. There is virtually no chart junk here, and graphical integrity is not difficult to ensure, as the overall readability can be adjusted simply by altering the width of the graph. It is equally as expressive as 538's chart(all of the necessary data and only that data is displayed), and is equally as effective in terms of interpreting the primary message(ranking of each emoji).

4 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. (20 points/ 15 points plot + 5 justification)

[30]:
```
# We'll help you out with a table that has the percentages for each emoji

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

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

# for each emoji, figure out how prevalent it is
emojis = ['','','','','','','','','','']
for emoji in emojis:
    sums[emoji] = (tweets[emoji].sum(),1)

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

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

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

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

# modify the text
sums_df['SUMS_TEXT'] = sums_df['SUMS'].astype('str')
```

[31]:
```
add_space = lambda x: str(x) + " "
sums_df['concat'] = sums_df['EMOJI'].map(add_space) +  " " +␣
 ↪sums_df['SUMS_TEXT']
sums_df.drop(columns=[1], inplace = True)
sums_df.head()
```

[31]:
```
  EMOJI  SUMS  rank SUMS_TEXT   concat
0        7203     1      7203   7203
1        1763     2      1763   1763
2        1077     3      1077   1077
3         920     4       920    920
4         785     5       785    785
```

[32]:
```
rank = ['', '', '', '', '', '', '', '', '', '' ]
```

```
#Now lets create the bars
bars = alt.Chart(sums_df).mark_bar(size=17).encode(
    # encode x as the percent, and hide the axis
    x=alt.X(
        'SUMS',
        axis=None),
    y=alt.Y(
        # encode y using the name, use the movie name to label the axis, sort␣
 ↪using the names_l
        'concat:N',
         axis=alt.Axis(tickCount=0, title=''),
         # we give the sorting order to avoid alphabetical order
        sort = rank
    )
)

# at this point we don't really have a great plot (it's missing the␣
 ↪annotations, titles, etc.)
bars



#raise NotImplementedError()
```

[32]:
```
alt.Chart(...)
```

[33]:
```
#Now lets style the text and combine the bars and the text



# finally, we're going to combine the bars and the text and do some styling
emojis = ( bars).configure_mark(
    # we don't love the blue
    color='#FFA500'
).configure_view(
    # we don't want a stroke around the bars
    strokeWidth=0
).configure_scale(
    # add some padding
    bandPaddingInner=0.2
).properties(
    # set the dimensions of the visualization
    width=500,
    height=200
).properties(

).properties(
     title={
      "text": ["Total count of the top 10 emojis tweeted during the fight, of␣
↪240,000 tweets"],

    }
)

emojis
```

[33]: `alt.Chart(...)`

Although a rather simple plot to construct from a code complexity standpoint, I believe that the visualization shown above - nothing more than a horizontal bar chart displaying the total occurences of each of the top ten emojis over the course of the entire fight - should have been created to complement to 538's similar horizontal bar chart, where the occurences of these emojis were displayed as percentages compromising a whole. It could feasibly be argued that percentages, on their own, are not very useful in regards to referring information about the underlying data if the underlying magnitude of said percentage is not known. In other words, 25% of 2000 observations, vs. 25% of 25 observations, are drastically different in their magnitude, and have signifiantly different implications in the real world, depending on the domain. In lieu of actually disclosing the total number of emojis tweeted during the fight(never given in 538's article), I felt that the next best thing to do in order to enable the viewer to glean a better understanding of the magnitude of the data they are interpreting would be to simply visualize some hard totals.

It would have been possible to display this exact data using a scatter plot if one were driven solely by Tufte's data-ink rationale, but those are more fit for comparison purposes rather than exemplifying magnitude, both of which can be handled by bar charts. The chart is devoid of superfluous chart-junk, and is adequately expressive enough to get across its intended message(total count of each emoji). There truly isn't a reasonably devisable alternative that wouldn't introduce needless complexity, thereby causing the visualization to be less effective.