assignment3

February 25, 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]:
```
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))
```

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

---

1 Information Visualization I

1.1 School of Information, University of Michigan

1.2 Week 3:

- Perception / Cognition

1.3 Assignment Overview

1.3.1 This assignment's objectives include:

- Review, refect, and apply the concepts of the perception pipeline. Justify how different encodings impact the effectiveness of a visualization depending on the human perception process.

  Preattentive Processing

- Recreate visualizations and 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: America's Favorite 'Star Wars' Movies (And Least Favorite Characters) (30 points)
- Altair programming exercise (70 points)

1.3.3 Resources:

- Article by FiveThirtyEight available online (Hickey, 2014)
- Datasets from FiveThirtyEight, we have downloaded a subset of this data in the folder ./assets
  - The original dataset can be found at FiveThirtyEight Star Wars Survey

1.4 Part 1. Perception and Cognition (30 points)

Read the article "America's Favorite 'Star Wars' Movies (And Least Favorite Characters)," and answer the following questions:

1.4.1 1.1 List the different data types in the following visualizations and their encodings (10 points)

Look at the following visualizations. For each, list the variable, their type, and the encoding used (e.g., Weight, quantitative, color, ...)

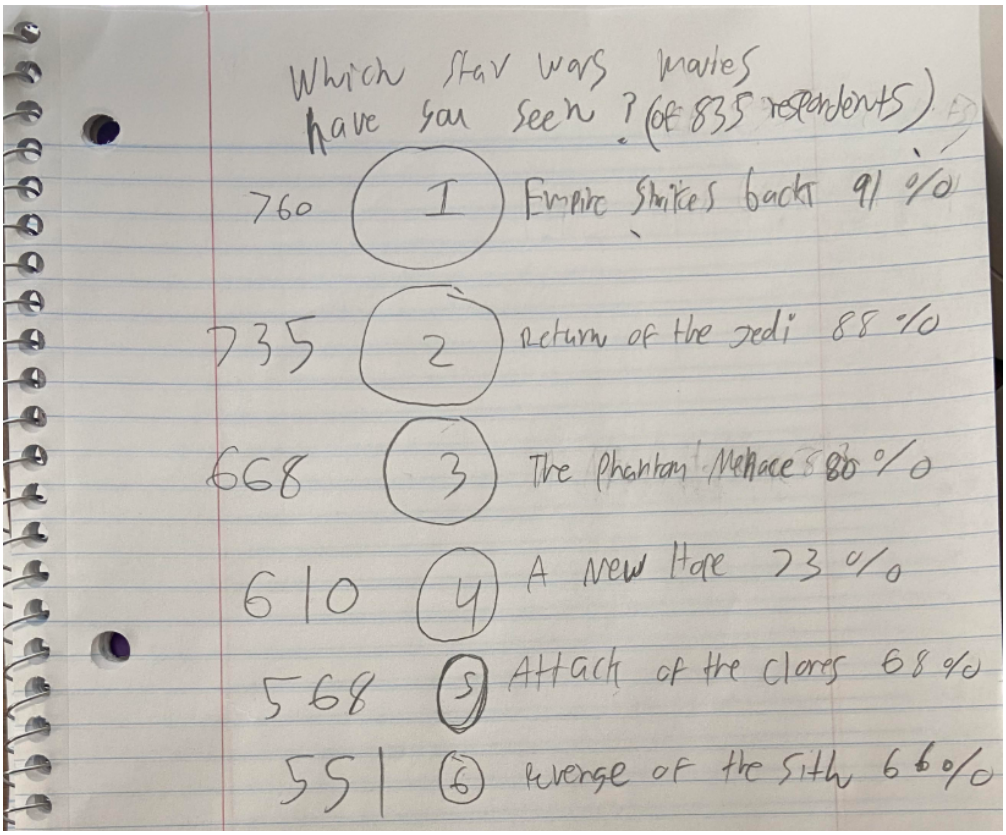
For the two visualizations, the variables included, and their corresponding data types and encodings, are as follows:

Top visualization These are the minimum number of variables needed to reconstruct the top visualization: 1. Name of the movie, which is a qualitative variable, encoded using text 2. The percentage of 471 respondents who place film in Top third, which is a quantitative variable, encoded by color 3. The percentage of 471 respondents who place film in Middle third, which is a quantitative variable, encoded by color 4. The percentage of 471 respondents who place film in Bottom third, which is a quantitative variable, encoded by color

Bottom visualization These are the minimum number of variables needed to reconstruct the top visualization 1. Name of the movie, which is a nominal variable, encoded using text 2. The percentage of 835 respondents who saw the film, which is a quantitative variable, encoded by the size of the bar on the x axis

1.4.2 1.2 Propose an alternative encoding for the following visualization. Compare the visualizations based on perception. (10 points)

Either hand-draw or use an application to create a sketched solution. Upload an image and describe the differences between your solution and the FiveThirtyEight image in terms of perception (specifically for the task of comparing one movie to another).

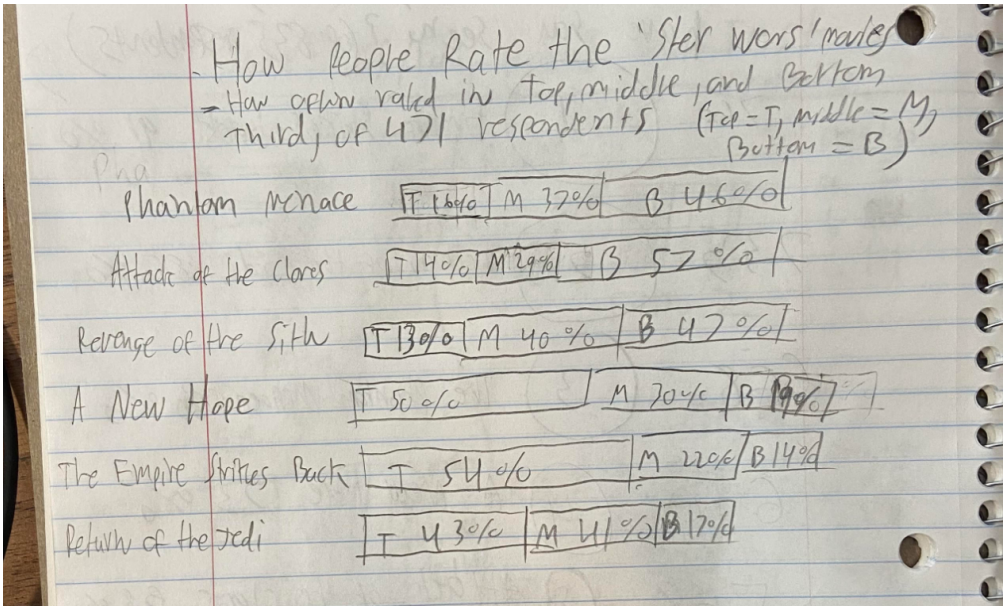


answer1.2

My proposed alternative encoding for 538's visualization is shown directly above. Instead of encoding the percentage of respondents as bars set as rows, I decided to create a bubble chart, and encode each percentage as a scaled circle. In addition to retaining the qualitative variable 'Name of movie'and the quantitative variable 'Percentage of viewers', both encoded on the labels on the right-hand side - as opposed to seperating both pieces of information on either side of a bar, which would necessitate eye scroll on the part of the viewer of the visualization - I added an additional variable of rank, which is encoded in the ordering of the circles and in the number displayed within each circle's center. I personally believe that this additional encoding, as well as the scaling of each circle, would allow the viewer to perceive differences in responses for each movie much more quickly than 538's visualization, especially considering the fact that there is no inherent ordering to the bars displayed in 538's visualization. Furthermore, it is still possible for the viewer to infer the number of respondents that responded in the affirmative for each movie, made easier by the fact that the visualization displays each number to the left of each circle. This gives the viewer the benefit of additional choice, in the sense that, if they are primarily interested in information regarding the specific number of respondents for a single movie, as opposed to the percentage difference in responses between movies, they can take the specific numberical figure they need and be done with the visualization. That being said, this bubble chart could be hamstrung if the number of movies under consideration were to increase greatly. It may be necessary to scale the circles exceptionally small, which would make it impossible to overlay the ranking above them.

1.4.3 1.3 Propose an alternative encoding for the following visualization. Compare the visualizations based on perception. (10 points)

Again, either-hand draw or use an application to create a sketched solution. Upload an image and describe the differences between your solution and the FiveThirtyEight image in terms of perception (specifically for the task of comparing one movie to another).



answer1.2

Since people tend to perceive numbers grouped together as constituting the same whole, which is certainly the case for the 'top', 'middle', and 'bottom' percentages displayed for each movie in 538's visualization, I thought that it would be wholly acceptable to convert the three seperate bars in each row to a single, stacked bar. I have retained all of the same variables as 538's visualization - namely, 'name of the movie,' 'percentage who placed in top', 'percentagewho placed in middle,' and 'percentage who placed in bottom' - with the added benefit of additional whitespace, which inherently makes the visualization less busy and easier to retain in memory. Furthermore, stacking each of the bars on top of one another naturally makes it easier to compare sizes, with the tier and percentage overlayed over each bar. I believe it is worth noting, however, that this approach only works in this scenario because each row shows the percentage coverage for that specific movie. While it would technically not be erroneous to stack bars if each individual bar represented a different movie, so long as all of the observations were retained, a viewer may, naturally, try to group all of the bars together, thinking they constitute the same whole, when in fact they do not, making group interpretation meaningless.

1.5 Part 2. Altair programming exercise (70 points)

We have provided you with some code and parts of the article America's Favorite 'Star Wars' Movies (And Least Favorite Characters). This article is based on the dataset:

1. StarWars Created by FiveThirtyEight based on a survey ran through SurveyMonkey Audience, surveying 1,186 respondents from June 3 to 6 2014. Available [online] (https://github.com/fivethirtyeight/data/tree/master/star-wars-survey)

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). We provide one example. Each visualization is worth 10 points (40 points/ 10 each x 4 total).
  - *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 functional Altair implementation*
- Propose one alternative visualization for one of the article visualizations. Add a short paragraph describing why your visualization is more *effective* based on principles of perception/cognition. (15 points/ 10 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 Perception/Cognition processes. (15 points/ 10 points plot + 5 justification)

[3]:
```
import pandas as pd
import altair as alt
import numpy as np
import math

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

[4]:
```
# enable correct rendering
alt.renderers.enable('default')
```

[4]:
```
RendererRegistry.enable('default')
```

[5]:
```
# uses intermediate json files to speed things up
alt.data_transformers.enable('json')
```

[5]:
```
DataTransformerRegistry.enable('json')
```

[6]:
```
sw = pd.read_csv('assets/StarWars.csv', encoding='latin1')
```

[7]:
```
# Some format is needed for the survey dataframe, we provide the formatted␣
 ↪dataset in a dataframe
sw = sw.rename(columns={'Have you seen any of the 6 films in the Star Wars␣
 ↪franchise?':'seen_any_movie',
                        'Do you consider yourself to be a fan of the Star Wars␣
 ↪film franchise?': 'fan',
                        'Which of the following Star Wars films have you seen?␣
 ↪Please select all that apply.' : 'seen_EI',
                        'Unnamed: 4' : 'seen_EII',
                        'Unnamed: 5' : 'seen_EIII',
                        'Unnamed: 6' : 'seen_EIV',
                        'Unnamed: 7' : 'seen_EV',
                        'Unnamed: 8' : 'seen_EVI',
```

```
                        'Please rank the Star Wars films in order of preference␣
↪with 1 being your favorite film in the franchise and 6 being your least␣
↪favorite film.' : 'rank_EI',
                        'Unnamed: 10' : 'rank_EII',
                        'Unnamed: 11' : 'rank_EIII',
                        'Unnamed: 12' : 'rank_EIV',
                        'Unnamed: 13' : 'rank_EV',
                        'Unnamed: 14' : 'rank_EVI',
                        'Please state whether you view the following characters␣
↪favorably, unfavorably, or are unfamiliar with him/her.' : 'Han Solo',
                        'Unnamed: 16' : 'Luke Skywalker',
                        'Unnamed: 17' : 'Princess Leia Organa',
                        'Unnamed: 18' : 'Anakin Skywalker',
                        'Unnamed: 19' : 'Obi Wan Kenobi',
                        'Unnamed: 20' : 'Emperor Palpatine',
                        'Unnamed: 21' : 'Darth Vader',
                        'Unnamed: 22' : 'Lando Calrissian',
                        'Unnamed: 23' : 'Boba Fett',
                        'Unnamed: 24' : 'C-3PO',
                        'Unnamed: 25' : 'R2 D2',
                        'Unnamed: 26' : 'Jar Jar Binks',
                        'Unnamed: 27' : 'Padme Amidala',
                        'Unnamed: 28' : 'Yoda',
                       })
sw = sw.drop([0])
```

[8]:
```
# take a peak to look at the data
sw.sample(5)
```

[8]:
```
      RespondentID seen_any_movie  fan  \
681   3.289943e+09            Yes   No
1029  3.288635e+09            Yes   No
1133  3.288479e+09             No  NaN
363   3.290712e+09            Yes   No
861   3.289528e+09             No  NaN

                                       seen_EI  \
681                                        NaN
1029  Star Wars: Episode I  The Phantom Menace
1133                                       NaN
363                                        NaN
861                                        NaN

                                      seen_EII  \
681                                        NaN
1029  Star Wars: Episode II  Attack of the Clones
1133                                       NaN
363   Star Wars: Episode II  Attack of the Clones
```

```
861                                          NaN

                                        seen_EIII  \
681                                          NaN
1029  Star Wars: Episode III  Revenge of the Sith
1133                                         NaN
363   Star Wars: Episode III  Revenge of the Sith
861                                          NaN

                               seen_EIV  \
681   Star Wars: Episode IV  A New Hope
1029  Star Wars: Episode IV  A New Hope
1133                                NaN
363                                 NaN
861                                 NaN

                                           seen_EV  \
681   Star Wars: Episode V The Empire Strikes Back
1029                                           NaN
1133                                           NaN
363                                            NaN
861                                            NaN

                                      seen_EVI rank_EI rank_EII rank_EIII  \
681   Star Wars: Episode VI Return of the Jedi       5        6         1
1029                                       NaN       1        2         3
1133                                       NaN     NaN      NaN       NaN
363                                        NaN       6        3         5
861                                        NaN     NaN      NaN       NaN

     rank_EIV rank_EV rank_EVI        Han Solo      Luke Skywalker  \
681         2       4        3  Very favorably      Very favorably
1029        4       5        6  Very favorably      Very favorably
1133      NaN     NaN      NaN             NaN                 NaN
363         4       1        2  Very favorably  Somewhat favorably
861       NaN     NaN      NaN             NaN                 NaN

     Princess Leia Organa      Anakin Skywalker  Obi Wan Kenobi  \
681        Very favorably    Somewhat favorably  Very favorably
1029       Very favorably    Somewhat favorably  Very favorably
1133                  NaN                   NaN             NaN
363        Very favorably  Somewhat unfavorably  Very favorably
861                   NaN                   NaN             NaN

                              Emperor Palpatine         Darth Vader  \
681   Neither favorably nor unfavorably (neutral)  Somewhat favorably
1029                        Somewhat unfavorably    Very unfavorably
```

```
1133                                      NaN                  NaN
363                              Very favorably  Somewhat favorably
861                                       NaN                  NaN

         Lando Calrissian             Boba Fett           C-3P0  \
681     Somewhat favorably  Somewhat unfavorably  Very favorably
1029  Somewhat unfavorably        Very favorably  Very favorably
1133                   NaN                   NaN             NaN
363         Very favorably        Very favorably  Very favorably
861                    NaN                   NaN             NaN

                                        R2 D2  \
681                          Somewhat unfavorably
1029  Neither favorably nor unfavorably (neutral)
1133                                      NaN
363                              Very favorably
861                                       NaN

                                  Jar Jar Binks  \
681                             Somewhat favorably
1029  Neither favorably nor unfavorably (neutral)
1133                                      NaN
363                             Very unfavorably
861                                       NaN

                                   Padme Amidala  \
681   Neither favorably nor unfavorably (neutral)
1029  Neither favorably nor unfavorably (neutral)
1133                                      NaN
363                           Somewhat favorably
861                                       NaN

                                         Yoda Which character shot first?  \
681                              Very favorably                          Han
1029                             Very favorably                          Han
1133                                      NaN                          NaN
363   Neither favorably nor unfavorably (neutral)                          Han
861                                       NaN                          NaN

     Are you familiar with the Expanded Universe?  \
681                                           No
1029                                         Yes
1133                                         NaN
363                                           No
861                                          NaN

     Do you consider yourself to be a fan of the Expanded Universe?Ñæ  \
```

```
681                                                   NaN
1029                                                  Yes
1133                                                  NaN
363                                                   NaN
861                                                   NaN

     Do you consider yourself to be a fan of the Star Trek franchise?  Gender  \
681                                                  No                   Male
1029                                                 Yes                  Male
1133                                                 No                 Female
363                                                  Yes                Female
861                                                  No                 Female

        Age   Household Income                         Education  \
681    > 60  $50,000 - $99,999                   Graduate degree
1029   > 60  $25,000 - $49,999  Some college or Associate degree
1133   > 60       $0 - $24,999  Some college or Associate degree
363   18-29                NaN  Some college or Associate degree
861   18-29       $0 - $24,999                High school degree

     Location (Census Region)
681                  Mountain
1029       West South Central
1133           South Atlantic
363            South Atlantic
861            South Atlantic
```

2 America's Favorite 'Star Wars' Movies (And Least Favorite Characters)

Original article available at FiveThirtyEight

By Walt Hickey

Filed under Movies

Get the data on GitHub

This week, I caught a sneak peek of the X-Wing fighter from the new "Star Wars" films in production. The forthcoming movies — and the middling response to the most recent trilogy — provide a perfect excuse to examine some questions I've long wanted answers to: How many people are "Star Wars" fans? Does the rest of America realize that "The Empire Strikes Back" is clearly the best of the bunch? Which characters are most well-liked and most hated? And who shot first, Han Solo or Greedo?

We ran a poll through SurveyMonkey Audience, surveying 1,186 respondents from June 3 to 6 (the data is available on GitHub). Seventy-nine percent of those respondents said they had watched at least one of the "Star Wars" films. This question, incidentally, had a substantial difference by gender: 85 percent of men have seen at least one "Star Wars" film compared to 72 percent of women. Of people who have seen a film, men were also more likely to consider themselves a fan of the franchise: 72 percent of men compared to 60 percent of women.

We then asked respondents which of the films they had seen. With 835 people responding, here's the probability that someone has seen a given "Star Wars" film given that they have seen any Star Wars film:

[9]:
```
# Sample visualization

# We're going to fix the labels a bit so will create a mapping to the full␣
 ↪names
episodes = ['EI', 'EII', 'EIII', 'EIV', 'EV', 'EVI']
names = {
    'EI' : 'The Phantom Meanance', 'EII' : 'Attack of the clones', 'EIII' :␣
 ↪'Revenge of the Sith',
    'EIV': 'A New Hope', 'EV': 'The Empire Strikes Back', 'EVI' : 'The Return␣
 ↪of the Jedi'
}

# we're also going to use this order to sort, so names_l will now have our sort␣
 ↪order
names_l = [names[ep] for ep in episodes]

print("sort order: ",names_l)
```

```
sort order:  ['The Phantom Meanance', 'Attack of the clones', 'Revenge of the
Sith', 'A New Hope', 'The Empire Strikes Back', 'The Return of the Jedi']
```

[10]:
```
# let's do some data pre-processing... sw (star wars) has everything

# We want to only use those people who have seen at least one movie, let's get␣
 ↪the people, toss NAs
# and get the total count

# find people who have at least on of the columns (seen_*) not NaN
seen_at_least_one = sw.dropna(subset=['seen_' + ep for ep in␣
 ↪episodes],how='all')
total = len(seen_at_least_one)

print("total who have seen at least one: ", total)
```

```
total who have seen at least one:  835
```

[11]:
```
# for each movie, we're going to calculate the percents and generate a new data␣
 ↪frame
percs = []

# loop over each column and calculate the number of people who have seen the␣
 ↪movie
```

```
# specifically, filter out the people who are *NaN* for a specific episode (e.g.
 ↪, ep_EII), count them
# and divide by the percent
for seen_ep in ['seen_' + ep for ep in episodes]:
    perc = len(seen_at_least_one[~ pd.isna(seen_at_least_one[seen_ep])]) / total
    percs.append(perc)

# at this point percs is holding our percentages

# now we're going use a trick to make tuples--pairing names with␣
 ↪percents--using "zip" and then make a dataframe
tuples = list(zip([names[ep] for ep in episodes],percs))
seen_per_df = pd.DataFrame(tuples, columns = ['Name', 'Percentage'])
seen_per_df
```

[11]:

	Name	Percentage
0	The Phantom Meanance	0.805988
1	Attack of the clones	0.683832
2	Revenge of the Sith	0.658683
3	A New Hope	0.726946
4	The Empire Strikes Back	0.907784
5	The Return of the Jedi	0.883832

[12]:
```
# ok, time to make the chart... let's make a bar chart (use mark_bar)
bars = alt.Chart(seen_per_df).mark_bar(size=20).encode(
    # encode x as the percent, and hide the axis
    x=alt.X(
        'Percentage',
        axis=None),
    y=alt.Y(
        # encode y using the name, use the movie name to label the axis, sort␣
 ↪using the names_l
        'Name:N',
         axis=alt.Axis(tickCount=5, title=''),
         # we give the sorting order to avoid alphabetical order
         sort=names_l
    )
)

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

[12]:
```
alt.Chart(...)
```

[13]:
```
# we're going to overlay the text with the percentages, so let's make another␣
 ↪visualization
# that's just text labels
```

```
text = bars.mark_text(
    align='left',
    baseline='middle',
    dx=3  # Nudges text to right so it doesn't appear on top of the bar
).encode(
    # we'll use the percentage as the text
    text=alt.Text('Percentage:Q',format='.0%')
)

# finally, we're going to combine the bars and the text and do some styling
seen_movies = (text + bars).configure_mark(
    # we don't love the blue
    color='#008fd5'
).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=180
).properties(
    # add a title
    title="Which 'Star Wars' Movies Have you Seen?"
)

seen_movies

# note that we are NOT formatting this in the Five Thirty Eight Style yet...␣
 ↪we'll leave that to you to figure out
```

[13]:
```
alt.LayerChart(...)
```

So we can see that "Star Wars: Episode V — The Empire Strikes Back" is the film seen by the most number of people, followed by "Star Wars: Episode VI — Return of the Jedi." Appallingly, more people reported seeing "Star Wars: Episode I — The Phantom Menace" than the original "Star Wars" (renamed "Star Wars: Episode IV — A New Hope").

So, which movie is the best? We asked the subset of 471 respondents who indicated they have seen every "Star Wars" film to rank them from best to worst. From that question, we calculated the share of respondents who rated each film as their favorite.

** Homework note: Click here to see a version of this plot generated in Altair.

[14]:
```
#create mapping to names
episodes = ['EI', 'EII', 'EIII', 'EIV', 'EV', 'EVI']
names = {
    'EI' : 'The Phantom Meanance', 'EII' : 'Attack of the clones', 'EIII' :␣
 ↪'Revenge of the Sith',
```

```
    'EIV': 'A New Hope', 'EV': 'The Empire Strikes Back', 'EVI' : 'The Return␣
 ↪of the Jedi'
}

# create sort for names
names_l = [names[ep] for ep in episodes]

print("sort order: ",names_l)
```

```
sort order:  ['The Phantom Meanance', 'Attack of the clones', 'Revenge of the
Sith', 'A New Hope', 'The Empire Strikes Back', 'The Return of the Jedi']
```

[15]:
```
seen_all = sw.dropna(subset=['seen_' + ep for ep in episodes],how='any')
total = len(seen_all)

print("total who have seen all: ", total)
```

```
total who have seen all:  471
```

[16]:
```
# for each movie, we're going to calculate the percents and generate a new data␣
 ↪frame


# loop over each column and calculate the percentage of people who have ranked␣
 ↪each movie as number 1

list1 = []
for rank in ['rank_' + ep for ep in episodes]:
    total = (seen_all[rank] == '1').sum()
    list1.append(total)

perc = []

for total in list1:
    percentage = total/len(seen_all)
    perc.append(percentage)


# at this point perc is holding our percentages

# now we're going use a trick to make tuples--pairing names with␣
 ↪percents--using "zip" and then make a dataframe
tuples = list(zip([names[ep] for ep in episodes],perc))
all_df = pd.DataFrame(tuples, columns = ['Name', 'Percentage'])
all_df
```

[16]:
```
                      Name  Percentage
0     The Phantom Meanance    0.099788
1     Attack of the clones    0.038217
2      Revenge of the Sith    0.057325
3                A New Hope    0.271762
4  The Empire Strikes Back    0.358811
5   The Return of the Jedi    0.174098
```

[17]:
```
# make bar chart with mark_bar
bars = alt.Chart(all_df).mark_bar(size=20).encode(
    # encode x as the percent, and hide the axis
    x=alt.X(
        'Percentage',
        axis=None),
    y=alt.Y(
        # encode y using the name, use the movie name to label the axis, sort␣
 ↪using the names_l
        'Name:N',
         axis=alt.Axis(tickCount=5, title=''),
         # we give the sorting order to avoid alphabetical order
         sort=names_l
    )
)

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

[17]:
```
alt.Chart(...)
```

[18]:
```
# we're going to overlay the text with the percentages, so let's make another␣
 ↪visualization
# that's just text labels

text = bars.mark_text(
    align='left',
    baseline='middle',
    dx=3  # Nudges text to right so it doesn't appear on top of the bar
).encode(
    # we'll use the percentage as the text
    text=alt.Text('Percentage:Q',format='.0%')
)

# finally, we're going to combine the bars and the text and do some styling
seen_every = (text + bars).configure_mark(
    # we don't love the blue
    color='#008fd5'
).configure_view(
```

```
    # we don't want a stroke around the bars
    strokeWidth=0
).configure_scale(
    # add some padding
    bandPaddingInner=0.2
).configure(
    background='#DCDCDC'
).properties(
    # set the dimensions of the visualization
    width=500,
    height=180
).properties(
    # add a title
    #title="What's the Best 'Star Wars' Movie?"
     title={
      "text": ["What's the Best 'Star Wars' Movie?"],
      "subtitle": ["Of 471 respondents who have seen all six films"],
      "subtitleColor": "grey"
    }
)

seen_every
```

[18]:
```
alt.LayerChart(...)
```

2.1 Make sure to *style* your visualization to match the original the best you can

We can also drill down and find out, generally, how people rate the films. Overall, fans broke into two camps: those who preferred the original three movies and those who preferred the three prequels. People who said "The Empire Strikes Back" was their favorite were also likely to rate "A New Hope" and "Return of the Jedi" higher as well. Those who rated "The Phantom Menace" as the best film were more likely to rate prequels higher.

This chart shows how often each film was rated in the top third (best or second-best), the middle third (third or fourth) or the bottom third (second-worst or worst). It's a more nuanced take on the series:

** Homework note: Click here to see a version of this plot generated in Altair.

[19]:
```
# We're going to fix the labels a bit so will create a mapping to the full␣
 ↪names
episodes = ['EI', 'EII', 'EIII', 'EIV', 'EV', 'EVI']
names = {
    'EI' : 'The Phantom Meanance', 'EII' : 'Attack of the clones', 'EIII' :␣
 ↪'Revenge of the Sith',
    'EIV': 'A New Hope', 'EV': 'The Empire Strikes Back', 'EVI' : 'The Return␣
 ↪of the Jedi'
}

# apply the name sort again
```

```
names_l = [names[ep] for ep in episodes]

print("sort order: ",names_l)
```

```
sort order:  ['The Phantom Meanance', 'Attack of the clones', 'Revenge of the
Sith', 'A New Hope', 'The Empire Strikes Back', 'The Return of the Jedi']
```

[20]:
```
seen_all = sw.dropna(subset=['seen_' + ep for ep in episodes],how='any')
total = len(seen_all)

print("total who have seen all: ", total)
```

```
total who have seen all:  471
```

[21]:
```
#Get percentages of times each film ranked in top third
list1 = []
for rank in ['rank_' + ep for ep in episodes]:
    total_1 = (seen_all[rank] == '1').sum()
    total_2 = (seen_all[rank] == '2').sum()
    total = total_1 + total_2
    list1.append(total)

perc = []

for total in list1:
    percentage = total/len(seen_all)
    perc.append(percentage)

# at this point perc is holding our percentages

# now we're going use a trick to make tuples--pairing names with␣
 ↪percents--using "zip" and then make a dataframe
tuples = list(zip([names[ep] for ep in episodes],perc))
top_df = pd.DataFrame(tuples, columns = ['Name', 'Percentage'])
top_df
```

[21]:

	Name	Percentage
0	The Phantom Meanance	0.163482
1	Attack of the clones	0.138004
2	Revenge of the Sith	0.129512
3	A New Hope	0.498938
4	The Empire Strikes Back	0.641189
5	The Return of the Jedi	0.428875

[22]:
```
# ok, time to make the chart... let's make a bar chart (use mark_bar)
bars = alt.Chart(top_df).mark_bar(size=20, color='#8fd500', strokeWidth=0).
 ↪encode(
    # encode x as the percent, and hide the axis
```

```
    x=alt.X(
        'Percentage',
        axis=None,
        ),
    y=alt.Y(
        # encode y using the name, use the movie name to label the axis, sort␣
 ↪using the names_l
        'Name:N',
         axis=alt.Axis(tickCount=5, title=''),
         # we give the sorting order to avoid alphabetical order
         sort=names_l,
    )
)

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

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

[23]:
```
# we're going to overlay the text with the percentages, so let's make another␣
 ↪visualization
# that's just text labels

text = bars.mark_text(
    align='left',
    baseline='middle',
    dx=3  # Nudges text to right so it doesn't appear on top of the bar
).encode(
    # we'll use the percentage as the text
    text=alt.Text('Percentage:Q',format='.0%')
)

# finally, we're going to combine the bars and the text and do some styling
top_third = (text + bars).properties(
    # set the dimensions of the visualization
    width=75,
    height=300

).properties(
    # add a title
     title={
      "text": ["Top Third"]

    }
)

top_third
```

[23]:
```
alt.LayerChart(...)
```

[24]:
```
#Get percentages of times each film ranked in middle third
list1 = []
for rank in ['rank_' + ep for ep in episodes]:
    total_1 = (seen_all[rank] == '3').sum()
    total_2 = (seen_all[rank] == '4').sum()
    total = total_1 + total_2
    list1.append(total)

perc = []

for total in list1:
    percentage = total/len(seen_all)
    perc.append(percentage)

# at this point perc is holding our percentages

# now we're going use a trick to make tuples--pairing names with␣
 ↪percents--using "zip" and then make a dataframe
tuples = list(zip([names[ep] for ep in episodes],perc))
middle_df = pd.DataFrame(tuples, columns = ['Name', 'Percentage'])
middle_df
```

[24]:

	Name	Percentage
0	The Phantom Meanance	0.373673
1	Attack of the clones	0.288747
2	Revenge of the Sith	0.401274
3	A New Hope	0.309979
4	The Empire Strikes Back	0.220807
5	The Return of the Jedi	0.405520

[25]:
```
# ok, time to make the chart... let's make a bar chart (use mark_bar)
bars = alt.Chart(middle_df).mark_bar(size=20, color='#008fd5', strokeWidth=0).
 ↪encode(
    # encode x as the percent, and hide the axis
    x=alt.X(
        'Percentage',
        axis=None),
    y=alt.Y(
        # encode y using the name, use the movie name to label the axis, sort␣
 ↪using the names_l
        'Name:N',
         axis=None,
         # we give the sorting order to avoid alphabetical order
         sort=names_l
    )
)
```

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

[25]:
```
alt.Chart(...)
```

[26]:
```
# we're going to overlay the text with the percentages, so let's make another␣
 ↪visualization
# that's just text labels

text = bars.mark_text(
    align='left',
    baseline='middle',
    dx=3  # Nudges text to right so it doesn't appear on top of the bar
).encode(
    # we'll use the percentage as the text
    text=alt.Text('Percentage:Q',format='.0%')
)

# finally, we're going to combine the bars and the text and do some styling
middle_third = (text + bars).properties(
    # set the dimensions of the visualization
    width=75,
    height=300
).properties(
    # add a title
     title={
      "text": ["Middle Third"],
    }
)

middle_third
```

[26]:
```
alt.LayerChart(...)
```

[27]:
```
#Get percentages of times each film ranked in bottom third
list1 = []
for rank in ['rank_' + ep for ep in episodes]:
    total_1 = (seen_all[rank] == '5').sum()
    total_2 = (seen_all[rank] == '6').sum()
    total = total_1 + total_2
    list1.append(total)

perc = []

for total in list1:
    percentage = total/len(seen_all)
    perc.append(percentage)
```

```
# at this point perc is holding our percentages

# now we're going use a trick to make tuples--pairing names with␣
 ↪percents--using "zip" and then make a dataframe
tuples = list(zip([names[ep] for ep in episodes],perc))
bottom_df = pd.DataFrame(tuples, columns = ['Name', 'Percentage'])
bottom_df
```

[27]:

	Name	Percentage
0	The Phantom Meanance	0.462845
1	Attack of the clones	0.573248
2	Revenge of the Sith	0.467091
3	A New Hope	0.191083
4	The Empire Strikes Back	0.138004
5	The Return of the Jedi	0.165605

[28]:
```
# ok, time to make the chart... let's make a bar chart (use mark_bar)
bars = alt.Chart(bottom_df).mark_bar(size=20, color='#ff0000', strokeWidth=0).
 ↪encode(
    # encode x as the percent, and hide the axis
    x=alt.X(
        'Percentage',
        axis=None),
    y=alt.Y(
        # encode y using the name, use the movie name to label the axis, sort␣
 ↪using the names_l
        'Name:N',
         axis=None,
         # we give the sorting order to avoid alphabetical order
        sort=names_l
    )
)

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

[28]:
```
alt.Chart(...)
```

[29]:
```
# we're going to overlay the text with the percentages, so let's make another␣
 ↪visualization
# that's just text labels

text = bars.mark_text(
    align='left',
    baseline='middle',
    dx=3  # Nudges text to right so it doesn't appear on top of the bar
).encode(
```

```
    # we'll use the percentage as the text
    text=alt.Text('Percentage:Q',format='.0%')
)

# finally, we're going to combine the bars and the text and do some styling
bottom_third = (text + bars).properties(
    # set the dimensions of the visualization
    width=75,
    height=300
).properties(
    # add a title
     title={
      "text": ["Bottom Third"],
    }
)

bottom_third
```

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

[30]:
```
# Concatenate the three charts and add dark background style
combined_chart = alt.hconcat(top_third, middle_third, bottom_third).properties(
     title={
      "text": ["How People Rate the 'Star Wars' Movies"],
      "subtitle": ["How often each film was rated in the top, middle, and␣
 ↪bottom third", "by 471 respondents who have seen all six films"],
      "subtitleColor": "grey"
    }).configure_view(
    # we don't want a stroke around the bars
    strokeWidth=0
).configure_scale(
    # add some padding
    bandPaddingInner=0.2).configure(
    background='#DCDCDC')
combined_chart




# YOUR CODE HERE
#raise NotImplementedError()
```

[30]:
```
alt.HConcatChart(...)
```

Finally, we took a boilerplate format used by political favorability polls — "Please state whether you view the following characters favorably, unfavorably, or are unfamiliar with him/her" — and asked respondents to rate characters in the series.

** Homework note. Here's an example solution generated in Altair:

[31]:
```
# We're going to fix the labels a bit so will create a mapping to the full␣
 ↪names
names = [
    'Luke Skywalker', 'Han Solo', 'Princess Leia Organa',
    'Obi Wan Kenobi', 'Yoda', 'R2 D2', 'C-3PO',
    'Anakin Skywalker', 'Darth Vader', 'Lando Calrissian', 'Padme Amidala',␣
 ↪'Boba Fett', 'Emperor Palpatine', 'Jar Jar Binks'
]

# we're also going to use this order to sort, so names_l will now have our sort␣
 ↪order
names_l = names

print("sort order: ",names_l)
```

```
sort order:  ['Luke Skywalker', 'Han Solo', 'Princess Leia Organa', 'Obi Wan
Kenobi', 'Yoda', 'R2 D2', 'C-3PO', 'Anakin Skywalker', 'Darth Vader', 'Lando
Calrissian', 'Padme Amidala', 'Boba Fett', 'Emperor Palpatine', 'Jar Jar Binks']
```

[32]:
```
# let's do some data pre-processing... sw (star wars) has everything

# We want to only use those people who have seen at least one movie, let's get␣
 ↪the people, toss NAs
# and get the total count

# find people who have at least on of the columns (seen_*) not NaN
rank_of_characters = sw.dropna(subset=['Han Solo', 'Luke Skywalker', 'Princess␣
 ↪Leia Organa', 'Anakin Skywalker', 'Obi Wan Kenobi', 'Emperor Palpatine',␣
 ↪'Darth Vader', 'Lando Calrissian', 'Boba Fett', 'C-3PO', 'R2 D2', 'Jar Jar␣
 ↪Binks', 'Padme Amidala', 'Yoda' ],how='all')
total = len(rank_of_characters)

print("total who have seen at least one: ", total)
```

```
total who have seen at least one:  834
```

[33]:
```
#Get percentages of times each character was ranked as favorable
list1 = []
for name in names_l:
    total_1 = (rank_of_characters[name] == 'Very favorably').sum()
    total_2 = (rank_of_characters[name] == 'Somewhat favorably').sum()
    total = total_1 + total_2
    list1.append(total)


perc = []
```

```
for total in list1:
    percentage = total/(rank_of_characters[name].count())
    perc.append(percentage)

# at this point perc is holding our percentages

# now we're going use a trick to make tuples--pairing names with␣
 ↪percents--using "zip" and then make a dataframe
tuples = list(zip([name for name in names_l],perc))
favorable_df = pd.DataFrame(tuples, columns = ['Name', 'Percentage'])
favorable_df
```

[33]:

	Name	Percentage
0	Luke Skywalker	0.939099
1	Han Solo	0.926918
2	Princess Leia Organa	0.922046
3	Obi Wan Kenobi	0.913520
4	Yoda	0.912302
5	R2 D2	0.909866
6	C-3PO	0.856273
7	Anakin Skywalker	0.626066
8	Darth Vader	0.585871
9	Lando Calrissian	0.444580
10	Padme Amidala	0.427527
11	Boba Fett	0.354446
12	Emperor Palpatine	0.308161
13	Jar Jar Binks	0.294762

[34]:
```
# ok, time to make the chart... let's make a bar chart (use mark_bar)
bars = alt.Chart(favorable_df).mark_bar(size=20, color='#228B22',␣
 ↪strokeWidth=0).encode(
    # encode x as the percent, and hide the axis
    x=alt.X(
        'Percentage',
        axis=None),
    y=alt.Y(
        # encode y using the name, use the movie name to label the axis, sort␣
 ↪using the names_l
        'Name:N',
         axis=alt.Axis(tickCount=5, title=''),
         # we give the sorting order to avoid alphabetical order
        sort=names_l
    )
)

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

```
bars
```

[34]:
```
alt.Chart(...)
```

[35]:
```
# we're going to overlay the text with the percentages, so let's make another␣
 ↪visualization
# that's just text labels

text = bars.mark_text(
    align='left',
    baseline='middle',
    dx=3  # Nudges text to right so it doesn't appear on top of the bar
).encode(
    # we'll use the percentage as the text
    text=alt.Text('Percentage:Q',format='.0%')
)

# finally, we're going to combine the bars and the text and do some styling
favorable = (text + bars).properties(
    # set the dimensions of the visualization
    width=75,
    height=300

).properties(
    # add a title
     title={
      "text": ["Favorable"]

    }
)

favorable
```

[35]:
```
alt.LayerChart(...)
```

[36]:
```
#Get percentages of times each character was ranked as neutral
list1 = []
for name in names_l:
    total = (rank_of_characters[name] == 'Neither favorably nor unfavorably␣
 ↪(neutral)').sum()
    list1.append(total)


perc = []

for total in list1:
    percentage = total/(rank_of_characters[name].count())
    perc.append(percentage)
```

```
# at this point perc is holding our percentages

# now we're going use a trick to make tuples--pairing names with␣
 ↪percents--using "zip" and then make a dataframe
tuples = list(zip([name for name in names_l],perc))
neutral_df = pd.DataFrame(tuples, columns = ['Name', 'Percentage'])
neutral_df
```

[36]:

	Name	Percentage
0	Luke Skywalker	0.046285
1	Han Solo	0.053593
2	Princess Leia Organa	0.058465
3	Obi Wan Kenobi	0.052375
4	Yoda	0.062119
5	R2 D2	0.069428
6	C-3P0	0.096224
7	Anakin Skywalker	0.164434
8	Darth Vader	0.102314
9	Lando Calrissian	0.287454
10	Padme Amidala	0.252132
11	Boba Fett	0.302071
12	Emperor Palpatine	0.259440
13	Jar Jar Binks	0.199756

[37]:
```
# ok, time to make the chart... let's make a bar chart (use mark_bar)
bars = alt.Chart(neutral_df).mark_bar(size=20, color='#008fd5', strokeWidth=0).
 ↪encode(
    # encode x as the percent, and hide the axis
    x=alt.X(
        'Percentage',
        axis=None),
    y=alt.Y(
        # encode y using the name, use the movie name to label the axis, sort␣
 ↪using the names_l
        'Name:N',
         axis=None,
         # we give the sorting order to avoid alphabetical order
        sort=names_l
    )
)

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

[37]:
```
alt.Chart(...)
```

[38]:
```
# we're going to overlay the text with the percentages, so let's make another␣
 ↪visualization
# that's just text labels

text = bars.mark_text(
    align='left',
    baseline='middle',
    dx=3  # Nudges text to right so it doesn't appear on top of the bar
).encode(
    # we'll use the percentage as the text
    text=alt.Text('Percentage:Q',format='.0%')
)

# finally, we're going to combine the bars and the text and do some styling
neutral = (text + bars).properties(
    # set the dimensions of the visualization
    width=75,
    height=300

).properties(
    # add a title
     title={
      "text": ["Neutral"]

    }
)

neutral
```

[38]:
```
alt.LayerChart(...)
```

[39]:
```
#Get percentages of times each character was ranked as unfavorable
list1 = []
for name in names_l:
    total_1 = (rank_of_characters[name] == 'Somewhat unfavorably').sum()
    total_2 = (rank_of_characters[name] == 'Very unfavorably').sum()
    total = total_1 + total_2
    list1.append(total)
perc = []

for total in list1:
    percentage = total/(rank_of_characters[name].count())
    perc.append(percentage)

# at this point perc is holding our percentages

# now we're going use a trick to make tuples--pairing names with␣
 ↪percents--using "zip" and then make a dataframe
```

```
tuples = list(zip([name for name in names_l],perc))
unfavorable_df = pd.DataFrame(tuples, columns = ['Name', 'Percentage'])
unfavorable_df
```

[39]:

	Name	Percentage
0	Luke Skywalker	0.019488
1	Han Solo	0.010962
2	Princess Leia Organa	0.021924
3	Obi Wan Kenobi	0.018270
4	Yoda	0.019488
5	R2 D2	0.019488
6	C-3P0	0.036541
7	Anakin Skywalker	0.148599
8	Darth Vader	0.305725
9	Lando Calrissian	0.086480
10	Padme Amidala	0.112058
11	Boba Fett	0.171742
12	Emperor Palpatine	0.233861
13	Jar Jar Binks	0.372716

[40]:
```
# ok, time to make the chart... let's make a bar chart (use mark_bar)
bars = alt.Chart(neutral_df).mark_bar(size=20, color='#ff0000', strokeWidth=0).
 ↪encode(
    # encode x as the percent, and hide the axis
    x=alt.X(
        'Percentage',
        axis=None),
    y=alt.Y(
        # encode y using the name, use the movie name to label the axis, sort␣
 ↪using the names_l
        'Name:N',
         axis=None,
         # we give the sorting order to avoid alphabetical order
        sort=names_l
    )
)

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

[40]:
```
alt.Chart(...)
```

[41]:
```
# we're going to overlay the text with the percentages, so let's make another␣
 ↪visualization
# that's just text labels

text = bars.mark_text(
    align='left',
```

```
    baseline='middle',
    dx=3  # Nudges text to right so it doesn't appear on top of the bar
).encode(
    # we'll use the percentage as the text
    text=alt.Text('Percentage:Q',format='.0%')
)

# finally, we're going to combine the bars and the text and do some styling
unfavorable = (text + bars).properties(
    # set the dimensions of the visualization
    width=75,
    height=300

).properties(
    # add a title
     title={
      "text": ["Unfavorable"]

    }
)
unfavorable
```

[41]:
```
alt.LayerChart(...)
```

[42]:
```
#Get percentages of times each character was ranked as unfamiliar
list1 = []
for name in names_l:
    total = (rank_of_characters[name] == 'Unfamiliar (N/A)').sum()

    list1.append(total)
perc = []

for total in list1:
    percentage = total/(rank_of_characters[name].count())
    perc.append(percentage)

# at this point perc is holding our percentages

# now we're going use a trick to make tuples--pairing names with␣
 ↪percents--using "zip" and then make a dataframe
tuples = list(zip([name for name in names_l],perc))
unfamiliar_df = pd.DataFrame(tuples, columns = ['Name', 'Percentage'])
unfamiliar_df
```

[42]:

	Name	Percentage
0	Luke Skywalker	0.007308
1	Han Solo	0.018270
2	Princess Leia Organa	0.009744

```
3          Obi Wan Kenobi    0.020706
4                    Yoda    0.012180
5                   R2 D2    0.012180
6                   C-3PO    0.018270
7        Anakin Skywalker    0.063337
8             Darth Vader    0.012180
9        Lando Calrissian    0.180268
10          Padme Amidala    0.199756
11              Boba Fett    0.160780
12     Emperor Palpatine    0.190012
13          Jar Jar Binks    0.132765
```

[43]:
```
# ok, time to make the chart... let's make a bar chart (use mark_bar)
bars = alt.Chart(unfamiliar_df).mark_bar(size=20, color='#D3D3D3',␣
 ↪strokeWidth=0).encode(
    # encode x as the percent, and hide the axis
    x=alt.X(
        'Percentage',
        axis=None),
    y=alt.Y(
        # encode y using the name, use the movie name to label the axis, sort␣
 ↪using the names_l
        'Name:N',
         axis=None,
         # we give the sorting order to avoid alphabetical order
        sort=names_l
    )
)

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

[43]:
```
alt.Chart(...)
```

[44]:
```
# we're going to overlay the text with the percentages, so let's make another␣
 ↪visualization
# that's just text labels

text = bars.mark_text(
    align='left',
    baseline='middle',
    dx=3  # Nudges text to right so it doesn't appear on top of the bar
).encode(
    # we'll use the percentage as the text
    text=alt.Text('Percentage:Q',format='.0%')
)
```

```
# finally, we're going to combine the bars and the text and do some styling
unfamiliar = (text + bars).properties(
    # set the dimensions of the visualization
    width=75,
    height=300

).properties(
    # add a title
     title={
      "text": ["Unfamiliar"]

    }
)

unfamiliar
```

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

[45]:
```
# Concatenate the three charts and add dark background style
combined_chart = alt.hconcat(favorable, neutral, unfavorable, unfamiliar).
 ↪properties(
     title={
      "text": ["Star Wars Charactar Favorability Ratings"],
      "subtitle": ["By 834 respondents"],
      "subtitleColor": "grey"
    }).configure_view(
    # we don't want a stroke around the bars
    strokeWidth=0
).configure_scale(
    # add some padding
    bandPaddingInner=0.2).configure(
    background='#DCDCDC')
combined_chart




# YOUR CODE HERE
#raise NotImplementedError()
```

[45]: `alt.HConcatChart(...)`

You read that correctly. Jar Jar Binks has a lower favorability rating than the actual personification of evil in the galaxy.

And for those of you who want to know the impact that historical revisionism can have on a society:

** Homework note: Click here to see a version of this plot generated in Altair.

[46]:
```
copy_df = sw
#Create Han, Greedo, and I don't understand this question columns
d = ['Han']
copy_df['Pre-Han'] =  np.where(copy_df['Which character shot first?'].isin(d),␣
 ↪1, 0)
copy_df['Han'] = copy_df['Pre-Han'].replace(1, 'Han')
copy_df['Han'] = copy_df['Han'].replace(0, np.NaN)

e = ['Greedo']
copy_df['Pre-Greedo'] =  np.where(copy_df['Which character shot first?'].
 ↪isin(e), 1, 0)
copy_df['Greedo'] = copy_df['Pre-Greedo'].replace(1, 'Greedo')
copy_df['Greedo'] = copy_df['Greedo'].replace(0, np.NaN)

f = ["I don't understand this question"]

copy_df['Pre-Dont'] =  np.where(copy_df['Which character shot first?'].isin(f),␣
 ↪1, 0)
copy_df['Dont'] = copy_df['Pre-Dont'].replace(1, "I don't understand this␣
 ↪question")
copy_df['Dont'] = copy_df['Dont'].replace(0, np.NaN)
```

[47]:
```
# We're going to fix the labels a bit so will create a mapping to the full␣
 ↪names
names = [
    'Han', 'Greedo', "I don't understand this question"
]

# we're also going to use this order to sort, so names_l will now have our sort␣
 ↪order
names_l = names

print("sort order: ",names_l)
```

```
sort order:  ['Han', 'Greedo', "I don't understand this question"]
```

[48]:
```
# let's do some data pre-processing... sw (star wars) has everything

# We want to only use those people who have seen at least one movie, let's get␣
 ↪the people, toss NAs
# and get the total count

# find people who have at least on of the columns (seen_*) not NaN
who_shot_first = sw.dropna(subset=['Han', 'Greedo', 'Dont' ],how='all')
total = len(who_shot_first)

print("total who have seen at least one: ", total)
```

```
total who have seen at least one:  828
```

[49]:
```
#Get percentages of times each character was ranked as favorable
grand_total = 828
list1 = []
total_1 = who_shot_first['Han'].count()
total_2 = who_shot_first['Greedo'].count()
total_3 = who_shot_first['Dont'].count()

list1 = [total_1, total_2, total_3]


perc = []
for total in list1:
    percentage = total/grand_total
    perc.append(percentage)


# at this point perc is holding our percentages

# now we're going use a trick to make tuples--pairing names with␣
 ↪percents--using "zip" and then make a dataframe
tuples = list(zip([name for name in names_1],perc))
who_shot_df = pd.DataFrame(tuples, columns = ['Name', 'Percentage'])
who_shot_df
```

[49]:

	Name	Percentage
0	Han	0.392512
1	Greedo	0.237923
2	I don't understand this question	0.369565

[50]:
```
# ok, time to make the chart... let's make a bar chart (use mark_bar)
bars = alt.Chart(who_shot_df).mark_bar(size=20, color='#008fd5', strokeWidth=0).
 ↪encode(
    # encode x as the percent, and hide the axis
    x=alt.X(
        'Percentage',
        axis=None),
    y=alt.Y(
        # encode y using the name, use the movie name to label the axis, sort␣
 ↪using the names_l
        'Name:N',
         axis=alt.Axis(tickCount=5, title=''),
         # we give the sorting order to avoid alphabetical order
        sort=names_1
    )
)
```

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

[50]:
```
alt.Chart(...)
```

[51]:
```
# we're going to overlay the text with the percentages, so let's make another␣
 ↪visualization
# that's just text labels

text = bars.mark_text(
    align='left',
    baseline='middle',
    dx=3  # Nudges text to right so it doesn't appear on top of the bar
).encode(
    # we'll use the percentage as the text
    text=alt.Text('Percentage:Q',format='.0%')
)

# finally, we're going to combine the bars and the text and do some styling
who_shot_first = (text + bars).properties(
    # set the dimensions of the visualization
    width=75,
    height=300

).properties(
    # add a title
     title={
      "text": ["Who Shot First?"],
      "subtitle": ["According to 834 respondents"],
      "subtitleColor": "grey"}).configure_view(
    strokeWidth=0
).configure_scale(
    bandPaddingInner=0.2
).properties(
    width=500,
    height=180
).configure(
    background='#DCDCDC')




who_shot_first
```

[51]:
```
alt.LayerChart(...)
```

3 Make your own (part 1)

Propose and code an alternative visualization for one of the visualizations *already in the article*. Add a short paragraph describing why your visualization is more (or less) *effective* based on principles of perception/cognition. (15 points/ 10 points plot + 5 justification)

[52]:
```
episodes = ['EI', 'EII', 'EIII', 'EIV', 'EV', 'EVI']
names = {
    'EI' : 'The Phantom Meanance', 'EII' : 'Attack of the clones', 'EIII' :␣
 ↪'Revenge of the Sith',
    'EIV': 'A New Hope', 'EV': 'The Empire Strikes Back', 'EVI' : 'The Return␣
 ↪of the Jedi'
}


names_l = [names[ep] for ep in episodes]
```

[53]:
```
seen_at_least_one = sw.dropna(subset=['seen_' + ep for ep in␣
 ↪episodes],how='all')
total = len(seen_at_least_one)
```

[54]:
```
# for each movie, we're going to calculate the percents and generate a new data␣
 ↪frame
percs = []

# loop over each column and calculate the number of people who have seen the␣
 ↪movie
# specifically, filter out the people who are *NaN* for a specific episode (e.g.
 ↪, ep_EII), count them
# and divide by the percent
for seen_ep in ['seen_' + ep for ep in episodes]:
    perc = len(seen_at_least_one[~ pd.isna(seen_at_least_one[seen_ep])]) / total
    percs.append(perc)



# at this point percs is holding our percentages

# now we're going use a trick to make tuples--pairing names with␣
 ↪percents--using "zip" and then make a dataframe
tuples = list(zip([names[ep] for ep in episodes],percs))
seen_per_df = pd.DataFrame(tuples, columns = ['Name', 'Percentage'])
seen_per_df
seen_per_df.rename(columns = {'Name':'Film'}, inplace = True)
```

[55]:
```
alt.Chart(seen_per_df).mark_point(filled=True, color = '#008fd5').encode(
  x=alt.X(
        'Film:N',
        axis=alt.Axis(title = 'Film')),
```

```
    y=alt.Y(
        # encode y using the name, use the movie name to label the axis, sort␣
 ↪using the names_l
        'Percentage',
         axis=alt.Axis( title='Percentage'))).properties(
    # set the dimensions of the visualization
    width=300,
    height=300).properties(
    # add a title
  title={
      "text": ["Which 'Star Wars' Movies Have You Seen?"],
      "subtitle": ["Of 835 respondents who have seen any film"],
      "subtitleColor": "grey"}).configure(
    background='#DCDCDC').configure_axis(
    labelFontSize=15,
    titleFontSize=15
)
```

[55]:
```
alt.Chart(...)
```

Recreation of initial example visualization

I somewhat contend with 538's decision to formulate this visualization as a horizontal row chart, since horizontal direction - intuitively - does not traditionally convey differences in length or height. The small number of films being compared, and thus, the small number of labels for 'film name' necessary for the visualization, eliminates any need for rotating the plot and placing qualitative values on the y-axis for spacial reasons, which is, generally speaking, not the norm in the first place. It would have been perfectly viable, then, for 538's visualizers to refrain from rotating the bar plot, thereby allowing the plot to remain directionally intuitive.

In choosing the purposefully simple scatterplot shown above, I was primarily motivated by the desire to easily allow the viewer to compare percentage values between the six films. While differences in height between multiple objects, if significantly close enough, can be rather difficult to discern, and is most certainly a deficiency with the potential to disadvantage bar plots, which utilizes height as its primary medium of conveyance, differences in elevation amongst two dimensional objects can be discerned immediately, and requires little to no non-preattentive cognitive processing on the part of the viewer. If the film name and the percentage of respondents who answered in the affirmative for each movie are the only variables that need to be encoded, then a generic scatter plot would certainly suffice, and would arguably convey the differences between each movie with more salience through its use of elevation, as opposed to height.

4 Make your own (part 2)

Propose and code a *new visualization* to complement a part of the article. Add a short paragraph justifying your decisions in terms of Perception/Cognition processes. (15 points/ 10 points plot + 5 justification)

[56]:
```
##Question: What percentage of respondents who have seen all star wars films␣
 ↪are from each geographic region of the US?
copy_df = sw
```

```
##filter the dataframe into only those respondents who saw all six films



episodes = ['EI', 'EII', 'EIII', 'EIV', 'EV', 'EVI']
names = {
    'EI' : 'The Phantom Meanance', 'EII' : 'Attack of the clones', 'EIII' :␣
 ↪'Revenge of the Sith',
    'EIV': 'A New Hope', 'EV': 'The Empire Strikes Back', 'EVI' : 'The Return␣
 ↪of the Jedi'
}




copy_df = copy_df.dropna(subset=['seen_' + ep for ep in episodes],how='all')
total = len(seen_at_least_one)

print("total who have seen at least one: ", total)
```

```
total who have seen at least one:  835
```

[57]:
```
#Create columns for all the regions in the country
d = ['South Atlantic']

copy_df['Pre-South Atlantic'] =  np.where(copy_df['Location (Census Region)'].
 ↪isin(d), 1, 0)
copy_df['South Atlantic'] = copy_df['Pre-South Atlantic'].replace(1, 'South␣
 ↪Atlantic')
copy_df['South Atlantic'] = copy_df['South Atlantic'].replace(0, np.NaN)

e = ['West South Central']

copy_df['Pre-West South Central'] =  np.where(copy_df['Location (Census␣
 ↪Region)'].isin(e), 1, 0)
copy_df['West South Central'] = copy_df['Pre-West South Central'].replace(1,␣
 ↪'West South Central')
copy_df['West South Central'] = copy_df['West South Central'].replace(0, np.NaN)

f = ['West North Central']

copy_df['Pre-West North Central'] =  np.where(copy_df['Location (Census␣
 ↪Region)'].isin(f), 1, 0)
```

```
copy_df['West North Central'] = copy_df['Pre-West North Central'].replace(1,␣
 ↪'West North Central')
copy_df['West North Central'] = copy_df['West North Central'].replace(0, np.NaN)

g = ['Middle Atlantic']

copy_df['Pre-Middle Atlantic'] =  np.where(copy_df['Location (Census Region)'].
 ↪isin(g), 1, 0)
copy_df['Middle Atlantic'] = copy_df['Pre-Middle Atlantic'].replace(1, 'Middle␣
 ↪Atlantic')
copy_df['Middle Atlantic'] = copy_df['Middle Atlantic'].replace(0, np.NaN)

h = ['East North Central']

copy_df['Pre-East North Central'] =  np.where(copy_df['Location (Census␣
 ↪Region)'].isin(h), 1, 0)
copy_df['East North Central'] = copy_df['Pre-East North Central'].replace(1,␣
 ↪'East North Central')
copy_df['East North Central'] = copy_df['East North Central'].replace(0, np.NaN)

i = ['Pacific']

copy_df['Pre-Pacific'] =  np.where(copy_df['Location (Census Region)'].isin(i),␣
 ↪1, 0)
copy_df['Pacific'] = copy_df['Pre-Pacific'].replace(1, 'Pacific')
copy_df['Pacific'] = copy_df['Pacific'].replace(0, np.NaN)

j = ['Mountain']

copy_df['Pre-Mountain'] =  np.where(copy_df['Location (Census Region)'].
 ↪isin(j), 1, 0)
copy_df['Mountain'] = copy_df['Pre-Mountain'].replace(1, 'Mountain')
copy_df['Mountain'] = copy_df['Mountain'].replace(0, np.NaN)

k = ['New England']

copy_df['Pre-New England'] =  np.where(copy_df['Location (Census Region)'].
 ↪isin(k), 1, 0)
copy_df['New England'] = copy_df['Pre-New England'].replace(1, 'New England')
copy_df['New England'] = copy_df['New England'].replace(0, np.NaN)

l = ['East South Central']

copy_df['Pre-East South Central'] =  np.where(copy_df['Location (Census␣
 ↪Region)'].isin(l), 1, 0)
```

```
copy_df['East South Central'] = copy_df['Pre-East South Central'].replace(1,␣
 ↪'East South Central')
copy_df['East South Central'] = copy_df['East South Central'].replace(0, np.NaN)
```

[58]:
```
#Let's make an arbitrary sort order


names = ['South Atlantic', 'West South Central', 'West North Central', 'Middle␣
 ↪Atlantic', 'East North Central', 'Pacific', 'Mountain', 'New England', 'East␣
 ↪South Central']


names_l = names

print("sort order: ",names_l)
```

```
sort order:  ['South Atlantic', 'West South Central', 'West North Central',
'Middle Atlantic', 'East North Central', 'Pacific', 'Mountain', 'New England',
'East South Central']
```

[59]:
```
#Get percentages of times each character was ranked as favorable
grand_total = 835
list1 = []
total_1 = copy_df['South Atlantic'].count()
total_2 = copy_df['West South Central'].count()
total_3 = copy_df['West North Central'].count()
total_4 = copy_df['Middle Atlantic'].count()
total_5 = copy_df['East North Central'].count()
total_6 = copy_df['Pacific'].count()
total_7 = copy_df['Mountain'].count()
total_8 = copy_df['New England'].count()
total_9 = copy_df['East South Central'].count()


list1 = [total_1, total_2, total_3, total_4, total_5, total_6, total_7,␣
 ↪total_8, total_9]


perc = []
for total in list1:
    percentage = total/grand_total
    perc.append(percentage)


# at this point perc is holding our percentages
```

```
# now we're going use a trick to make tuples--pairing names with␣
 ↪percents--using "zip" and then make a dataframe
tuples = list(zip([name for name in names_l],perc))
where_from_df = pd.DataFrame(tuples, columns = ['Name', 'Percentage'])
where_from_df
```

[59]:

	Name	Percentage
0	South Atlantic	0.159281
1	West South Central	0.091018
2	West North Central	0.091018
3	Middle Atlantic	0.111377
4	East North Central	0.159281
5	Pacific	0.174850
6	Mountain	0.081437
7	New England	0.071856
8	East South Central	0.038323

[60]:
```
# ok, time to make the chart... let's make a bar chart (use mark_bar)
bars = alt.Chart(where_from_df).mark_bar(size=15, color='#008fd5',␣
 ↪strokeWidth=0).encode(
    # encode x as the percent, and hide the axis
    x=alt.X(
        'Percentage',
        axis=None),
    y=alt.Y(
        # encode y using the name, use the movie name to label the axis, sort␣
 ↪using the names_l
        'Name:N',
         axis=alt.Axis(tickCount=5, title=''),
         # we give the sorting order to avoid alphabetical order
        sort=names_l
    )
)

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

[60]:
```
alt.Chart(...)
```

[61]:
```
text = bars.mark_text(
    align='left',
    baseline='middle',
    dx=3
).encode(

    text=alt.Text('Percentage:Q',format='.0%')
)
```

```
where_from = (text + bars).properties(

    width=75,
    height=300

).properties(

     title={
      "text": ["Where Are You From?"],
      "subtitle": ["According to 835 respondents who've seen all six 'Star␣
 ↪Wars' films (3% of respondents no location)."],
      "subtitleColor": "grey"}).configure_view(
    strokeWidth=0
).configure_scale(
    bandPaddingInner=0.2
).properties(
    width=500,
    height=180
).configure(
    background='#DCDCDC')




where_from
```

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

With respondent location information readily available, I felt that 538's visualizers overlooked a potential opportunity to glean, at the very least, a basic understanding of their respondensts' demographic information. At a purfunctory glance, it appears that most of the respondents - all of whom have seen all six 'Star Wars' films - hail from the Pacific region of the United States - namely, Alaska, California, Hawaii, Oregon, and Washington. Given the films' classification as 'Hollywood' films, the inclusion of California in this region may have tipped the scales upward ever so slightly, though this is unsubstantiated. Conversely, the region that boasted the fewest respondents who saw all six films is the East South Central region, comprising of Alabama, Kentucky, Mississippi, and Tennessee.

From a perception/cognition perspective, I aimed to match the styling mold of 538's other basic bar charts. On account of the larger number of labels and the smaller width of the bars that this necessitated, I decided against rotating the chart vertically to circumvent issues regarding the readibility of the labels. While this is less directionally intuitive, in situations where labels are numerous, viewers can parse through the visualization from the top down, as opposed to the bottom up, which is arguably more intuitive from a purely physical standpoint. In general, we begin to dissect visualizations starting from the top, where the title is.