assignment2

February 16, 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"
```

---

1 Information Visualization I

1.1 School of Information, University of Michigan

1.2 Week 2:

- Expressiveness and Effectiveness
- Grammar of Graphics

1.3 Assignment Overview

1.3.1 Our objectives for this week:

- Review, reflect, and apply the concepts of encoding. Given a visualization recreate the data that was encoded.
- Review, reflect, and apply the concepts of Expressiveness and Effectiveness. Given a visualization, evaluate alternatives with the same expressiveness.

Two visualizations, same expressiveness

- Review and evaluate an implementation of Grammar of Graphics using Altair

1.3.2 The total score of this assignment will be 100 points consisting of:

- Case study reflection: Next Bechdel Test (30 points)
- Altair programming exercise (70 points)
- Bonus (5 points)

1.3.3 Resources:

- This article by FiveThirtyEight available online (Hickey, Koeze, Dottle, Wezerek 2017)
- Datasets from FiveThirtyEight, we have downloaded a subset of these datasets available in the folder for your use into `./assets`
  - The original dataset can be found on FiveThirtyEight Next Bechdel Dataset

1.4 Part 1. Expressiveness and Effectiveness (30 points)

Read the following article *Creating the next Bechdel Test* and answer the following questions:

1.4.1 1.1 Recreate the table (by hand or excel) needed to create the following visualization (7 points)

Take a picture or screenshot of your table and add it to the answer below

An easy way to upload images is to jump into the `./assets` directory (or use the Coursera notebook explorer and navigate to it) and then use the upload button to save your image:

upload

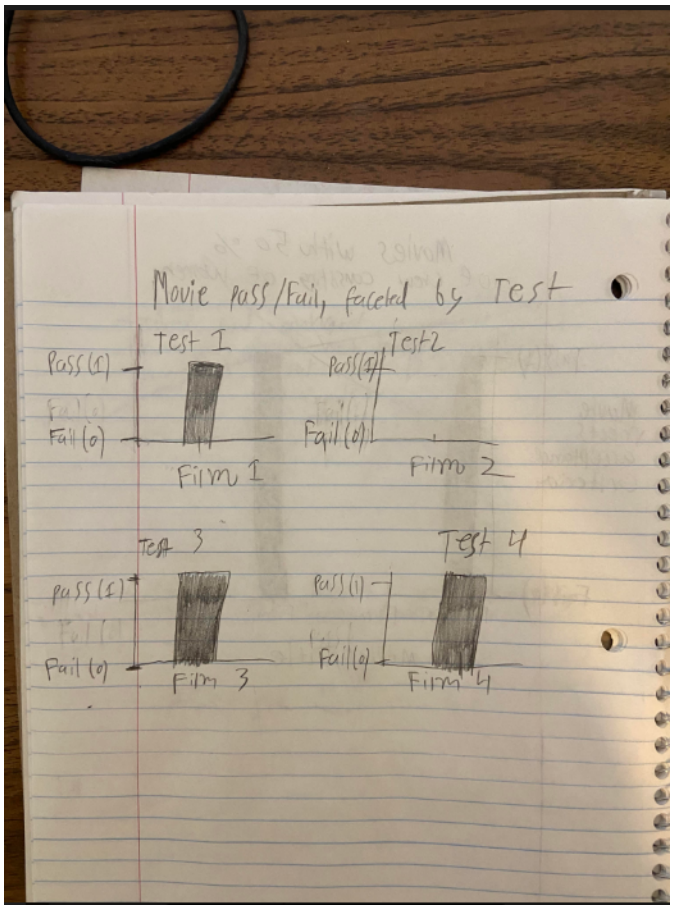
Once you have the image, you can link to it using the markdown command: `![answer1.2](assets/my_image_1.2.png)`

Movie Title	Total Crew Members(All Genders)	Cinematography Crew(F)	Costume Design Crew(F)	Screenwriting Crew(F)	Makeup Design Crew(F)	Lighting Crew(F)	Sum(F)	Percentage	Pass/Fail
Film 1	300	30	26	16	42	22	136	0.453333333	0
Film 2	250	50	40	45	30	30	195	0.78	1
Film 3	100	14	8	10	7	9	48	0.48	0
Film 4	558	67	45	80	90	56	338	0.605734767	1

answer1.1

1.4.2 1.2 Sketch an alternative visualization with the same expressiveness (7 points)

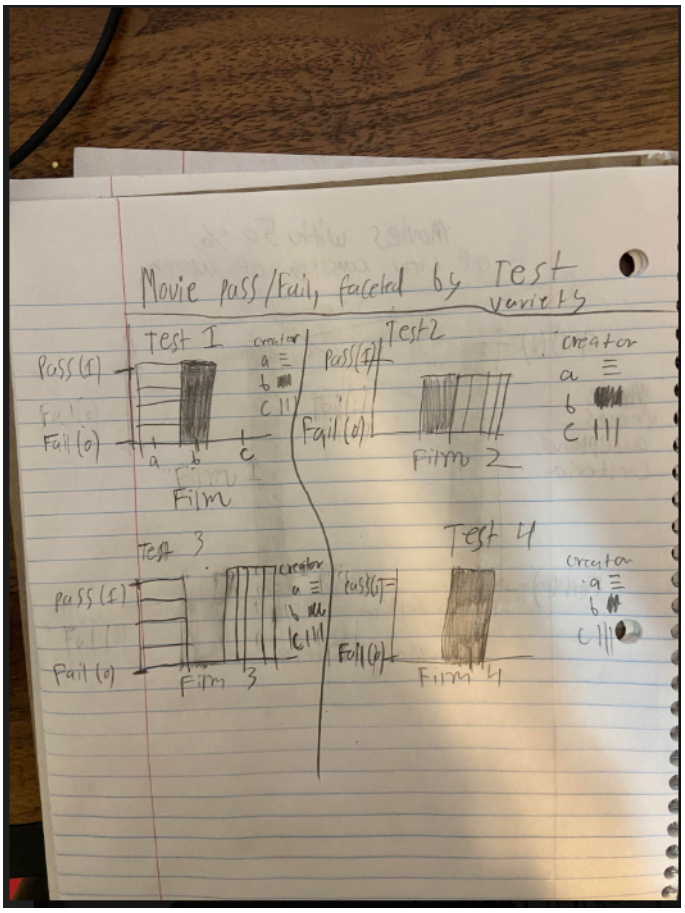
By hand is fine, but you can also use a tool. This is a sketch, the data need not be perfectly accurate or to scale. Again, upload a picture or screenshot below. Make sure there is enough annotation so it's clear why your picture has the same expressiveness.



answer1.2

1.4.3 1.3 Sketch an alternative visualization with the same expressiveness of the following visualization (10 points)

Same deal as last question: by hand or with a tool is fine. The data need not be perfectly accurate or to scale. Make sure there is enough annotation so it's clear why your picture has the same expressiveness. Again, upload a picture or screenshot below.



answer1.3

1.4.4 1.4 Reflect on which visualization you think is more *effective* and why? (6 points)

The above vizualization published by 538, and my hand drawn rough sketch, in general, express the same information - namely, whether a certain movie passed one of the various test varieties discussed in the article, while also encoding the bars by individual test creator in order to convey whether or not the film passed their specific version of the test, as 538's visualization also conveys. However, I can certainly say that the visualization published by 538 is inconvtrovertibly more effective than my rough version, in the sense that it is capable of displaying a much larger array of films within its limited space. Furthermore, it is much easier to decipher the encoding of 538's visualization. Blank squares can be disregarded entirely, and there is no need to display the name of each test creator, because their visages are embedded directly within the visualization. In the case of my own sketch, it would arguably take the viewer a few extra

1.5 Part 2. Altair programming exercise (70 points)

We have provided some code to create visualizations based on the following datasets:

1. all_tests Is a collection of different Bechdel test results for the 50 top-grossing films at the domestic box office in 2016
2. cast_gender Is the gender for all the cast member of each movie in the Bechdel rankings

3. top_2016 Is the date, box office and theater count for each top 2016 movie.

Complete each assignment function and run each cell to generate the final visualizations

[69]:
```
import pandas as pd
import numpy as np
import altair as alt

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

[18]:
```
# enable correct rendering
alt.renderers.enable('default')
```

[18]:
```
RendererRegistry.enable('default')
```

[19]:
```
# uses intermediate json files to speed things up
alt.data_transformers.enable('json')
```

[19]:
```
DataTransformerRegistry.enable('json')
```

[20]:
```
# read all the tables
all_tests_df = pd.read_csv('assets/nextBechdel_allTests.csv')
cast_gender = pd.read_csv('assets/nextBechdel_castGender.csv')
top_2016 = pd.read_csv('assets/top_2016.csv')
```

[21]:
```
# set up the tables for use
actors_movies = top_2016.set_index('Movie').join(cast_gender.
 ↪set_index('MOVIE')).join(all_tests_df.set_index('movie')).reset_index().
 ↪dropna()
movies_order = top_2016.sort_values(by=['Rank'])['Movie'].tolist()
```

1.5.1 2.1 Variables encoded (5 points)

Warmup: complete the `variables_encoded` function. Return the number of variables encoded in the following example

[22]:
```
base = alt.Chart(actors_movies).transform_filter(
    (alt.datum.TYPE != 'Unknown') & (alt.datum.GENDER != 'Unknown') & (alt.
 ↪datum.GENDER != 'null')
)
```

[70]:
```
actors_movies.head(10)
```

[70]:

	index	Rank	Domestic Box Office Opening	Theater Count	\
0	10 Cloverfield Lane	44	$72,082,999	3,391	
1	10 Cloverfield Lane	44	$72,082,999	3,391	
2	10 Cloverfield Lane	44	$72,082,999	3,391	
3	10 Cloverfield Lane	44	$72,082,999	3,391	
4	10 Cloverfield Lane	44	$72,082,999	3,391	
5	10 Cloverfield Lane	44	$72,082,999	3,391	
6	10 Cloverfield Lane	44	$72,082,999	3,391	
7	10 Cloverfield Lane	44	$72,082,999	3,391	

```
8  10 Cloverfield Lane    44        $72,082,999                  3,391
9  10 Cloverfield Lane    44        $72,082,999                  3,391

  Opening Weekend Box Office Max Theater Count                   ACTOR  \
0               $24,727,437                3,427             John Goodman
1               $24,727,437                3,427  Mary Elizabeth Winstead
2               $24,727,437                3,427      John Gallagher, Jr.
3               $24,727,437                3,427                Mat Vairo
4               $24,727,437                3,427             Maya Erskine
5               $24,727,437                3,427              Cindy Hogan
6               $24,727,437                3,427       Douglas M. Griffin
7               $24,727,437                3,427                Jamie Clay
8               $24,727,437                3,427          Sumalee Montano
9               $24,727,437                3,427             Frank Mottek

      CHARACTER_NAME        TYPE BILLING   GENDER  bechdel  peirce  landau  \
0             Howard     Leading       1     Male        0       0       1
1           Michelle     Leading       2   Female        0       0       1
2             Emmitt     Leading       3     Male        0       0       1
3             Jeremy  Supporting       4     Male        0       0       1
4              Darcy  Supporting       5   Female        0       0       1
5    Neighbor Woman  Supporting       6   Female        0       0       1
6             Driver  Supporting       7     Male        0       0       1
7   State Trooper 42  Supporting       8  Unknown        0       0       1
8     Voice on Radio  Supporting       9  Unknown        0       0       1
9  Radio Broadcaster  Supporting      10     Male        0       0       1

   feldman  villareal  hagen  ko  villarobos  waithe  koeze_dottle  uphold  \
0        1          0      1   1           1       1             1       1
1        1          0      1   1           1       1             1       1
2        1          0      1   1           1       1             1       1
3        1          0      1   1           1       1             1       1
4        1          0      1   1           1       1             1       1
5        1          0      1   1           1       1             1       1
6        1          0      1   1           1       1             1       1
7        1          0      1   1           1       1             1       1
8        1          0      1   1           1       1             1       1
9        1          0      1   1           1       1             1       1

   white  rees-davies
0      1            1
1      1            1
2      1            1
3      1            1
4      1            1
5      1            1
6      1            1
```

```
7      1            1
8      1            1
9      1            1
```

[23]:
```
encoding = base.transform_filter(
    alt.datum.GENDER == 'Female'
).encode(
    y= alt.Y(
        'index:N',
        sort= movies_order
    ),
    x=alt.X('count(index):Q',
            title='cast count'),
)

def m_bar():
    return encoding.mark_bar().properties(title='Female')

bar = m_bar()
```

[24]:
```
bar
```

[24]:
```
alt.Chart(...)
```

[71]:
```
def variables_encoded():
    """
    This is a helper function for automated grading, just return the number (e.g., 'return 50000;'). You don't
    need to do anything fancy here.
    """
    return 2
    #raise NotImplementedError()
```

[26]:
```
#hidden tests are within this cell
```

1.5.2 2.2 Alternative encoding (5 points)

Complete the `m_circle` function. Change the encoding used in the previous example from a bar to a circle. Your visualization should look like

[27]:
```
def m_circle():
    """
    return the call to altair function that uses the circle mark for the variables encoded in the previous example
    """
    # return encoding. (...)
    return encoding.mark_circle().properties(title='Female')
    #raise NotImplementedError()
```

[28]:
```
circle = m_circle()
circle
```

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

[29]:
```
#hidden tests are within this cell
```

1.5.3 2.3 Increase variables encoded (5 points)

Complete the `female_actors` function. Modify the first bar chart encoding the type of the actor with the color of the bar. Your visualization should look like the following (note that we don't have labels yet):

- *Partial credit can be granted for each visualization (up to 2 points) if you provide a description of what the missing piece of the function is supposed to do without need for an Altair working version*

[61]:
```
def female_actors():
    """
    return the call to Altair function that uses the bar mark for the variables␣
 ↪and the color for the TYPE
    """
    encoding = base.transform_filter(
    alt.datum.GENDER == 'Female'
    ).encode(
        y= alt.Y(
            'index:N',
            sort= movies_order,
            axis=None
        ),
        x=alt.X('count(index):Q',
                title='cast count'),
        color=alt.Color('TYPE:N')
    )
    # YOUR CODE HERE
    #raise NotImplementedError()
    return encoding.mark_bar().properties(title='Female')
```

[62]:
```
female = female_actors()
female
```

[62]:
```
alt.Chart(...)
```

[63]:
```
#hidden tests are within this cell
```

1.5.4 2.4 Change filter transform (5 points)

Complete the male_actors function, modify the previous visualization so that the actors visualized have Male gender. Use the Altair transform function for this, not Pandas

- *Partial credit can be granted for each visualization (up to 2 points) if you provide a description of what the missing piece of the function is supposed to do without need for an altair working version*

[64]:
```
def male_actors():
    """
    return the call to altair function that uses the bar mark for the variables␣
 ↪and the color for the TYPE
    this function filters the actors to be male
    """
    #add filter transform

    # This is the starting point. Again, modify or replace this code to get the␣
 ↪encoding we describe above
    encoding =  base.transform_filter(
    alt.datum.GENDER == 'Male'
    ).encode(
            y= alt.Y(
            'index:N',
            sort= movies_order
        ),
        x=alt.X('count(index):Q',
                sort='descending',
                title='cast count'),
          color=alt.Color('TYPE:N')
    ).mark_bar().properties(title='Male')

    # YOUR CODE HERE
    #raise NotImplementedError()

    return encoding.mark_bar().properties(title='Male')
```

[65]:
```
male = male_actors()
male
```

[65]:
```
alt.Chart(...)
```

[66]:
```
#hidden tests are within this cell
```

1.5.5 2.5 Variables encoded 2 (5 points)

Complete the `variables_encoded_2` function. Return the number of variables encoded in the following plot (again, something like `return 5000;...` we're using this for automated testing). If you have been able to complete the previous examples, the plot should look like this

[67]:
```
middle = base.encode(
    y=alt.Y('Rank:O', axis=None),
    text=alt.Text('Rank:Q'),
    color=alt.Color('bechdel:N')
).mark_text().properties(width=20)
```

```
# merge together the three charts, male, middle, female
male | middle | female
```

[67]:
```
alt.HConcatChart(...)
```

[37]:
```
def variables_encoded_2():
    """
    return the number of variables encoded in the given example
    """
    return 4;
    #raise NotImplementedError()
```

[38]:
```
#hidden tests are within this cell
```

1.5.6 2.6 Alternative encoding 1 (20 Points)

Create a new visualization within the `alternative_encoding_one` function with the following encoding: - Use circles as the mark - Use the scale of the circles to encode the number of actors on each category - Use the y position of the circle to encode the movie - Use the x position of the circle to encode the type of actor - Use the color of the circle to encode the gender of the actor - Match the styling of the example

- *Partial credit can be granted for each visualization (up to 5 points) if you provide a description of what the missing piece of the function is supposed to do*

[39]:
```
def alternative_encoding_one():
    """
    return call to altair function for the new visualization
    """


    plot = base.mark_circle(
       opacity=0.8,
       stroke='black',
       strokeWidth=1
    ).encode(
        y = alt.Y('index:N',
              sort= movies_order),

         x=alt.X('TYPE:N',
                #sort='descending',
                title='cast count'),
          color=alt.Color('GENDER:N'),

        size =alt.Size('count(index):Q',
        scale=alt.Scale(range=[0, 4000]),
        legend=alt.Legend(title='Count of actors', symbolFillColor = 'white'))).
↪properties(
```

```
        width=350,
        height=880
    )
    # YOUR CODE HERE
    #raise NotImplementedError()
    return plot
```

[40]:
```
al_enc_one = alternative_encoding_one()
middle | al_enc_one
```

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

[41]:
```
#hidden tests are within this cell
```

1.5.7 2.7 Alternative encoding 2 (25 Points)

complete `female_actors_1()` and `male_actors_2()` functions to create a new visualization with the following encoding: - The left and right plot should filter male and female actors - Use rectangles as the mark - Use the text inside each rectangle to encode the count of actors one each category (gender, type and movie) - Use the y position of the rectangle to encode the movie - Use the x position of the rectangle to encode the type of actor - Use the color of the rectangle to encode whether that movie passes the Bechdel test or not (bechdel variable)

- *Partial credit can be granted for each visualization (up to 4 points for each function) if you provide a description of what the missing piece of the function is supposed to do without need for an Altair working version*

[44]:
```
def female_actors_1():
    """
    return call to altair function for the new visualization
    """
    #add filter transform
    plot = base.mark_rect().encode(
        alt.X('TYPE:N'),
        alt.Y('index:N',
        sort= movies_order),
        color=alt.Color('bechdel:N'),
        text='count(index):Q'
    ).transform_filter(
    alt.datum.GENDER == 'Female')



    #add filter transform
    text = base.mark_text(baseline='middle').encode(
        x='TYPE:O',
        y= alt.Y(
            'index:O',
            sort= movies_order,
```

```
            axis=None
        ),
        #change this
        text='count(index):Q'
    ).transform_filter(
    alt.datum.GENDER == 'Female')
    # YOUR CODE HERE
    #raise NotImplementedError()
    return plot + text
```

[45]:
```
f_a_1 = female_actors_1()
f_a_1
```

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

[46]:
```
#hidden tests are within this cell
```

[57]:
```
def male_actors_1():
    """
    return call to altair function for the new visualization
    """
    #add filter transform
    plot = base.mark_rect().encode(
        alt.X('TYPE:N'),
        alt.Y('index:N',
        sort= movies_order, axis = None),
        color=alt.Color('bechdel:N'),
        text='count(index):Q').transform_filter(
        alt.datum.GENDER == 'Male')
        # add text and color



    #add filter transform
    text = base.mark_text(baseline='middle').encode(
        x='TYPE:O',
        y= alt.Y(
            'index:N',
            sort= movies_order,
             axis=None
        ),
        #change this
        text='count(index):Q'
    ).transform_filter(
    alt.datum.GENDER == 'Male')
    # YOUR CODE HERE
    #raise NotImplementedError()
    return plot + text
```

[58]:
```
m_a_1 = male_actors_1()
m_a_1
```

[58]:
```
alt.LayerChart(...)
```

[59]:
```
#hidden tests are within this cell
```

[60]:
```
# create the visualization
f_a_1 | middle | m_a_1
```

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

1.5.8 2.8 (Bonus) Compare effectiveness (5 points)

Look at the visualization for question 2.7. How does this visualization compare in terms of effectiveness to the visualizations in questions 2.5 and 2.6?

I believe that the visualization produced for question 2.7 is, arguably, more effective than the vizualizations produced for 2.5 and 2.6. Although they express the same information - namely, count of cast per movie in various roles, encoded by gender and whether or not they passed the bechdel test - it is much easier to glean the relevant quantitative information from 2.7(count of cast in each role, seperated by gender, per movie). While the encoding does somewhat help in terms of bringing out the quantitative information in 2.5 and 2.6, it would take much longer for a new viewer to parse it out. In regards to 2.5, the viewer has to scroll all the way down to the x-axis labels in order to obtain the cast count, and then guide their vision upwards to the relevant bar. In other words, they have to go through a two step process in order to obtain a single value. 2.6 is arguably even worse than 2.5 in this respect, in the sense that it is nearly impossible for the viewer to judge, with exact precision, the count of cast for any movie. While the legend does offer a rough estimate of size to value, it is incredibly easy to misjudge size, and thereby misjudge the value.

The only rather glaring issue with the combined visualization in 2.7, is that it's impossible to know which of the two visualizations correspond to which gender. Although it would result in potential visual clutter, if this is something the creator of the graph is attempting to convey, labels above each graph that indicate this information is essential.