Information Visualization I

School of Information, University of Michigan

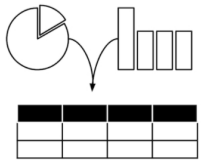
Week 2:

- Expressiveness and Effectiveness
- Grammar of Graphics

Assignment Overview

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

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)

Resources:

- This article by FiveThirtyEight available online (Hickey, Kosze, 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

Important notes:

1) Grading for this assignment is largely done by manual inspection. The autograder checks the basic details but is imperfect. Sometimes it will pass the test but not look right, sometimes it will look right and fail the test. That's fine! You should guide your answer by the look of our examples. It doesn't need to be pixel perfect (e.g., you may not always know what our example is scaled by), but it should be pretty close.

2) When turning in your PDF, please use the File -> Print -> Save as PDF option ***from your browser***. Do ***not*** use the File->Download as->PDF option. Complete instructions for this are under Resources in the Coursera page for this class.

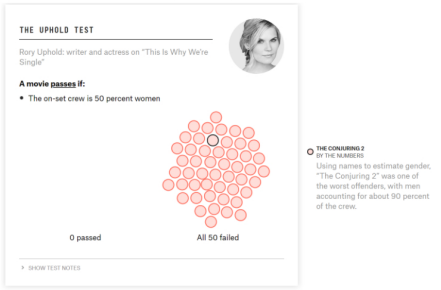
3) Pay attention to the return types of your functions.

Part 1. Expressiveness and Effectiveness (30 points)

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

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

You *should* consider the interactive parts of the visualization in your answer. 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:

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

ID	Title	Women Count	Men Count	Pass
1	Sausage Party	30	50	Fail
2	Don't Breathe	10	60	Fail
3	Arrival	25	50	Fail
4	Moana	35	60	Fail

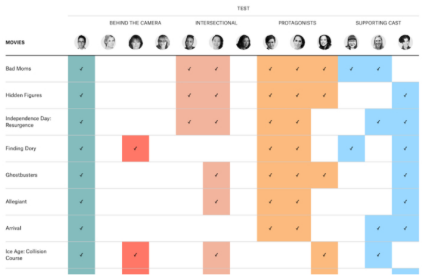
The pass and fail categories here would be created by computing the ratio of women and men in the workplace and deciding if at least half of the crew is female. If so, they will be binned into the pass category otherwise they fall into the fail category.

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

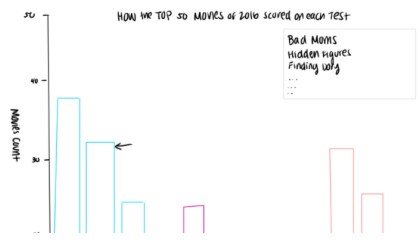
By hand is fine, but you can also use a tool. This is a sketch, the data need not be perfectly accurate or to scale. Again, upload a picture or screenshot below. Make sure there is enough annotation so it's clear why your picture has the same expressiveness.

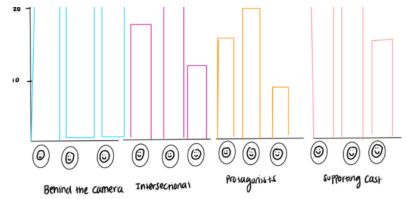
```
!(answer1.2)(assets/1.2.png)

This picture has the same expressiveness because it maintains the key information from the original figure: the
number of movies that passed/failed and the names of the movies that fall into each category. Note that the arrow
in the image represents a mouse, such that if you hover over each section of the bar chart you will see the name
of a movie that falls within that pass/fail category.
```

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.





This picture has the same expressiveness because it maintains the critical information from the original figure: who's test is represented, what category that test falls under (i.e., behind the camera, intersectional, protagonists, or supporting cast), and the movies that pass or fail. Note that the arrow in the image represents a mouse hovering. When a mouse hovers over a bar, a popup will display a list of all the movies that fall into that category ordered by box office ranking.

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

You are comparing the original figure in 1.3 and the one you created.

The visualization from FiveThirtyEight is more effective because its tabular structure makes it easy to compare how each movie ranks according to each person's test and across the different test categories (e.g., behind the camera, intersectional, protagonists, supporting cast).

The visualization by FiveThirtyEight puts the movies at the forefront of the visualization. In contrast, my visualization includes the movie titles as secondary information that the user can search for upon further investigation (i.e., with the hover of a mouse). Given that the main point is to compare how various movies (especially top-ranked movies) do across the multiple tests of female inclusivity, the visualization by FiveThirtyEight is better.

```
## Part 2. Altair programming exercise (70 points)
We have provided some code to create visualizations based on the following datasets:

1. [all_tests](assets/nextBechdel_allTests.csv) Is a collection of different Bechdel test results for the 50 top-grossing filme [at the domestic box office in 2016](https://www.the-numbers.com/box-office-records/domestic/all-movies/cumulative/released-in-2016)
2. [cast_gender](assets/nextBechdel_castGender.csv) Is the gender for all the cast member of each movie in the Bechdel rankings
3. [top_2016](assets/top_2016.csv) 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
```

In [1]:
```
import pandas as pd
import numpy as np
import altair as alt
```

In [2]:
```
# enable correct rendering
alt.renderers.enable('default')
```

Out[2]: RendererRegistry.enable('default')

In [3]:
```
# uses intermediate json files to speed things up
alt.data_transformers.enable('json')
```

Out[3]: DataTransformerRegistry.enable('json')

In [4]:
```
# 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')
```

Preview the Data

In [5]:
```
print(all_tests_df.columns)
all_tests_df.head(3)
```

```
Index(['movie', 'bechdel', 'peirce', 'landau', 'feldman', 'villareal', 'hagen',
       'ko', 'villarroboa', 'waithe', 'koeze_dottle', 'uphold', 'white',
       'rees-davies'],
      dtype='object')
```

Out[5]:

	movie	bechdel	peirce	landau	feldman	villareal	hagen	ko	villarroboa	waithe	koeze_dottle	uphold	white	rees-davies
0	Bad Moms	0	0	0	1	0	0	1	0	0	1	1	1
1	Hidden Figures	0	0	0	0	1	0	1	0	1	1	1	1
2	Independence Day: Resurgence	0	0	1	0	0	1	0	1	0	0	1	1	1

In [6]:
```
print(cast_gender.columns)
cast_gender.head(3)
```

```
Index(['MOVIE', 'ACTOR', 'CHARACTER_NAME', 'TYPE', 'BILLING', 'GENDER'], dtype='object')
```

Out[6]:

	MOVIE	ACTOR	CHARACTER_NAME	TYPE	BILLING	GENDER
0	Boo! A Madea Halloween	Tyler Perry	Madea/Joe/Brian	Leading	1	Male
1	Boo! A Madea Halloween	Cassi Davis	Aunt Bam	Supporting	2	Female
2	Boo! A Madea Halloween	Patrice Lovely	Hattie	Supporting	3	Female

In [7]:
```
print(top_2016.columns)
top_2016.head(3)
```

```
Index(['Rank', 'Movie', 'Domestic Box Office', 'Opening Theater Count',
       'Opening Weekend Box Office', 'Max Theater Count'],
      dtype='object')
```

Out[7]:

	Rank	Movie	Domestic Box Office	Opening Theater Count	Opening Weekend Box Office	Max Theater Count
0	1	Rogue One	$532,177,324	4,157	$155,081,681	4,157
1	2	Finding Dory	$486,295,561	4,305	$135,060,273	4,305
2	3	Captain America: Civil War	$408,084,349	4,226	$179,139,142	4,226

In [8]:
```
# set up the tables for use
ctors_movies = top_2016.set_index('Movie').join(cast_gender.set_index('MOVIE')).join(all_tests_df.set_index('movie'
ovies_order = top_2016.sort_values(by=['Rank'])['Movie'].tolist()
```

In [9]:
```
print(actors_movies.columns)
actors_movies
```

```
Index(['index', 'Rank', 'Domestic Box Office', 'Opening Theater Count',
       'Opening Weekend Box Office', 'Max Theater Count', 'ACTOR',
       'CHARACTER_NAME', 'TYPE', 'BILLING', 'GENDER', 'bechdel', 'peirce',
       'landau', 'feldman', 'villareal', 'hagen', 'ko', 'villarroboa', 'waithe',
       'koeze_dottle', 'uphold', 'white', 'rees-davies'],
      dtype='object')
```

Out[9]:

	index	Rank	Domestic Box Office	Opening Theater Count	Opening Weekend Box Office	Max Theater Count	ACTOR	CHARACTER_NAME	TYPE	BILLING	...	feldman	villareal	hagen	ko	villa
0	10 Cloverfield Lane	44	$72,082,999	3,391	$24,727,437	3,427	John Goodman	Howard	Leading	1	...	1	0	1	1	
1	10 Cloverfield Lane	44	$72,082,999	3,391	$24,727,437	3,427	Mary Elizabeth Winstead	Michelle	Leading	2	...	1	0	1	1	
2	10 Cloverfield Lane	44	$72,082,999	3,391	$24,727,437	3,427	John Gallagher, Jr.	Emmitt	Leading	3	...	1	0	1	1	
3	10 Cloverfield Lane	44	$72,082,999	3,391	$24,727,437	3,427	Mat Vairo	Jeremy	Supporting	4	...	1	0	1	1	
4	10 Cloverfield Lane	44	$72,082,999	3,391	$24,727,437	3,427	Maya Erskine	Darcy	Supporting	5	...	1	0	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2070	Zootopia	7	$341,268,248	3,827	$75,063,401	3,959	Byron Howard	Bucky Oryx-Antlerson	Supporting	28	...	1	0	1	1	
2071	Zootopia	7	$341,268,248	3,827	$75,063,401	3,959	Mark 'Rhino' Smith	Officer McHorn	Supporting	29	...	1	0	1	1	
2072	Zootopia	7	$341,268,248	3,827	$75,063,401	3,959	Jared Bush	Priank Oryx-Antlerson	Supporting	30	...	1	0	1	1	
2073	Zootopia	7	$341,268,248	3,827	$75,063,401	3,959	John Lasseter	Mouse Foreman	Supporting	31	...	1	0	1	1	
2074	Zootopia	7	$341,268,248	3,827	$75,063,401	3,959	Josie Trinidad	Landlady	Supporting	32	...	1	0	1	1	

2075 rows × 24 columns

In [10]:
```
print(movies_order)
```

```
['Rogue One', 'Finding Dory', 'Captain America: Civil War', 'The Secret Life of Pets', 'The Jungle Book', 'Deadpool', 'Zootopia', 'Batman v Superman: Dawn of Justice', 'Suicide Squad', 'Sing', 'Moana', 'Fantastic Beasts and Where to Find Them', 'Doctor Strange', 'Hidden Figures', 'Jason Bourne', 'Star Trek Beyond', 'X-Men: Apocalypse', 'Trolls', 'La La Land', 'Kung Fu Panda 3', 'Ghostbusters', 'Central Intelligence', 'The Legend of Tarzan', 'Sully', 'Bad Moms', 'The Angry Birds Movie', 'Independence Day: Resurgence', 'The Conjuring 2', 'Arrival', 'Passengers', 'Sausage Party', 'The Magnificent Seven', 'Ride Along 2', 'Don't Breathe', 'Miss Peregrine's Home for Peculiar Children', 'The Accountant', 'Teenage Mutant Ninja Turtles: Out of the Shadows', 'The Purge: Election Year', 'Alice Through the Looking Glass', 'Pete's Dragon', 'The Girl on the Train', 'Boo! A Madea Halloween', 'Storks', '10 Cloverfield Lane', 'Lights Out', 'Hacksaw Ridge', 'Allegiant', 'Now You See Me 2', 'Ice Age: Collision Course', 'The Boss']
```

2.1 Variables encoded (5 points)

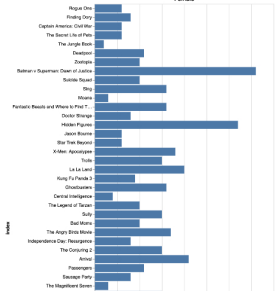
Warmup: complete the `variables_encoded` function. Return the number of variables encoded in the following example

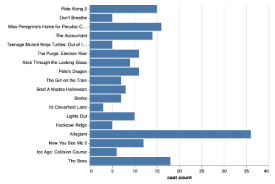
In [11]:
```
base = alt.Chart(actors_movies).transform_filter(
    (alt.datum.TYPE != 'Unknown') & (alt.datum.GENDER != 'Unknown') & (alt.datum.GENDER != 'null')
)
```

In [12]:
```
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()
```

In [13]:
```
bar
```

Out[13]:





In [14]:
```
def variables_encoded(vis):
    """
    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 len(vis.encoding.to_dict())
```

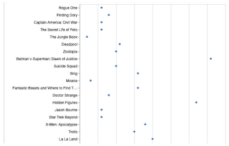
In [15]:
```
# test function on bar
variables_encoded(bar)
```

Out[15]: 2

In [16]:
```
#hidden tests are within this cell
```

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



click here to see the full-sized image.

In [17]:
```
def m_circle():
    """
    return the call to altair function that uses the circle mark for the variables encoded in the previous example
    """

    return encoding.mark_circle().properties(title='Female')
```

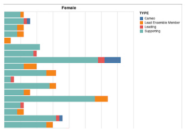
In [18]:
```
# test m_circle

# copy and pasted base and encoding as a reminder of the global var

base = alt.Chart(actors_movies).transform_filter(
    (alt.datum.TYPE != 'Unknown') & (alt.datum.GENDER != 'Unknown') & (alt.datum.GENDER != 'null')
)

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

m_circle()
```

Out[18]:

In [19]:
```
#hidden tests are within this cell
```

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):



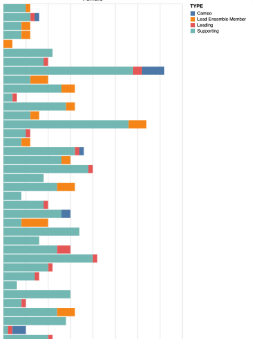
click here to see the full-sized image.

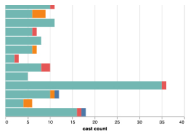
- *Partial credit can be granted for each visualization (up to 2 points) if you provide a description of what the missing piece of the function is supposed to do without need for an Altair working version*

In [20]:
```
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='TYPE'
    )

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

In [21]:
```
# testing solution
female = female_actors()
female
```

Out[21]:

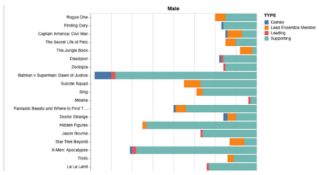




In [22]:
```
#hidden tests are within this cell
```

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.



click here to see the full-sized image.

- *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*

In [50]:
```
def male_actors():
    """
    return an altair-defined chart that uses the bar mark for the variables and the color for the TYPE
    this function filters the actors to be male
    """

    filter_encoding = base.transform_filter(
        (alt.datum.GENDER == 'Male')
    ).encode(
        y= alt.Y(
            'index:N',
            sort= movies_order,
            title='movies'
            ),
        x=alt.X(
            'count(index):Q',
            sort='descending',
            title='cast count'
            ),
        color='TYPE'
        )


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

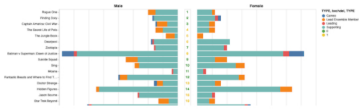
In [51]:
```
male = male_actors()
male
```

Out[51]:

In [25]:
```
#hidden tests are within this cell
```

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



click here to see the full-sized image.

In [52]:
```
middle = base.encode(
    y=alt.Y(
        'Rank:O',
        axis=None
        ),
    text=alt.Text(
        'Rank:Q'
        ),
    color=alt.Color(
        'backdel:N'
        )
    ).mark_text().properties(width=20)

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

Out[52]:

In [27]:
```
def variables_encoded_2():
    """
    return the number of variables encoded in the given example
    """
    return len(middle.encoding.to_dict())+len(male.encoding.to_dict())+len(female.encoding.to_dict())
```

In [28]:
```
variables_encoded_2()
```

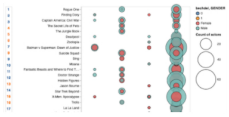
Out[28]:
```
9
```

In [29]:
```
#Hidden tests are within this cell
```

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



click here to see the full-sized image.

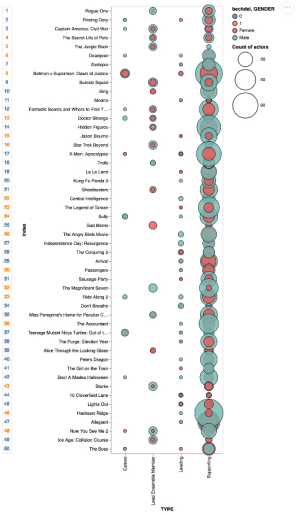
- *Partial credit can be granted for each visualization (up to 5 points) if you provide a description of what the missing piece of the function is supposed to do*

In [30]:
```
# a reminder of what base is
base = alt.Chart(actors_movies).transform_filter(
    (alt.datum.TYPE != 'Unknown') & (alt.datum.GENDER != 'Unknown') & (alt.datum.GENDER != 'null')
)
```

In [63]:
```
def alternative_encoding_one():
    """
    return call to altair function for the new visualization
    """

    # left part of the graph (box office ranking colored by bechdel test, pass y/n)
    left = base.mark_text().encode(
        y=alt.Y(
            'RankO',
            axis=None
            ),
        text=alt.Text(
            'RankO'
            ),
        color=alt.Color(
            'bechdel:N',
            )
        ).properties(width=20)

    # right part of the graph contains movie names, gender encoded by color scaled by size of the circles
    right = base.mark_circle(
                opacity=0.8,
                stroke='black',
                strokeWidth=1
            ).encode(
                x=alt.X(
                    'TYPE:N',
                    ),
                y=alt.Y(
                    'index:O',
                    sort= movies_order
                    ),
                color=alt.Color(
                    'GENDER:N'
                    ),
                size=alt.Size(
                    'count(actors):Q',
                    scale=alt.Scale(range=[0,4000]),
                    legend = alt.Legend(title='Count of actors',symbolFillColor='White')
                    )
                ).properties(width=250)
    return left | right
```

In [64]:
```
# Testing results
al_enc_one = alternative_encoding_one()

al_enc_one
```

Out[64]:



In [33]:
```
#Hidden tests are within this cell
```

2.7 Alternative encoding 2 (25 Points)

complete `female_actors_1()` and `male_actors_1()` functions to create a new visualization with the following encoding:

- [x] The left and right plot should filter male and female actors respectively
- [x] Use rectangles as the mark
- [x] Use the text inside each rectangle to encode the count of actors one each category (gender, type and movie)
- [x] Use the y position of the rectangle to encode the movie
- [x] Use the x position of the rectangle to encode the type of actor
- [x] Use the color of the rectangle to encode whether that movie passes the Bechdel test or not (bechdel variable)
- [x] Note that only the female vis has labels on the left, the Male vis does not

The top of the female plot would look like this (click [here](assets/visualization_3_fullSize.png) to see the full-sized image):

!["Drawing"](assets/visualization_3_resized.png)

If you've done everything correctly, your final visualization should look like the one below (click [here](assets/visualization_3_full.png) for the full plot).

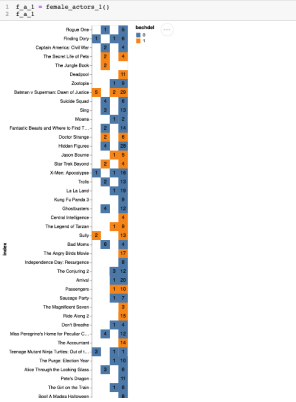
!["Drawing"](assets/visualization_3.png)

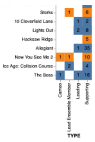
- *Partial credit can be granted for each visualization (up to 4 points for each function; if you provide a description of what the missing piece of the function is supposed to do without need for an Altair working version*

In [71]:
```
def female_actors_1():
    """
    return call to altair function for the new visualization
    """
    plot = base.mark_rect().transform_filter(
        alt.datum.GENDER == 'Female'
    ).encode(
        alt.X('TYPE:N'),
        alt.Y(
            'index:N',
            sort= movies_order),
        alt.Color('bechdel:N')
    )

    text = base.mark_text(baseline='middle').transform_filter(
        alt.datum.GENDER == 'Female'
        ).encode(
            x='TYPE:N',
            y= alt.Y(
                'index:N',
                sort= movies_order,
                axis=None
                ),
            text=alt.Text('count(actors):Q')
        )

    return plot + text
```

In [72]:
```
f_a_1 = female_actors_1()
f_a_1
```

Out[72]:

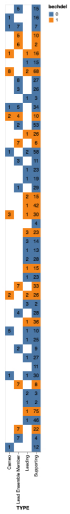




In [36]:
```
#Hidden tests are within this cell

```

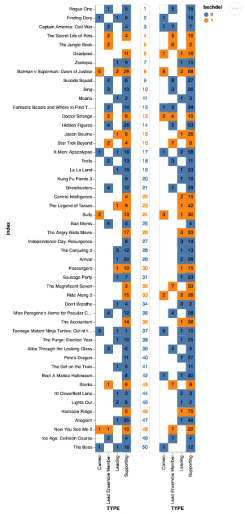
In [74]:
```
def male_actors_1():
    """
    return call to altair function for the new visualization
    """

    plot = base.mark_rect().transform_filter(
        alt.datum.GENDER == 'Male'
    ).encode(
        alt.X('TYPE:N'),
        alt.Y('index:N',
              sort= movies_order,
              axis=None
              ),
        alt.Color('bechdel:N')
    )

    text = base.mark_text(baseline='middle').transform_filter(
        alt.datum.GENDER == 'Male'
    ).encode(
        x='TYPE:N',
        y= alt.Y(
            'index:N',
              sort= movies_order,
              axis=None
        ),
        text=alt.Text('count(actors):Q')
    )
    return plot + text
```

In [75]:
```
m_a_1 = male_actors_1()
m_a_1
```

Out[75]:


In [39]:
```
#Hidden tests are within this cell

```

In [76]:
```
# create the visualization
f_a_1 | middle | m_a_1
```

Out[76]:


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?

> The visualization for question 2.7 is more effective because it is easier to perceive than visualizations 2.5 and 2.6. Visualization 2.7 is easier to perceive because it contains structures and encodings that are intuitive to the naive observer. For example, one doesn't need much prior knowledge about information visualization to understand that the checkered pattern conveys the presence or absence of actors of that type. The numbers can be easily understood (with annotations) as the count of movies that fall into that bin, and the color clearly describes the Bechdel categories.
>
> Meanwhile, stacked bar charts are hard to read, especially for novices. Not only are stacked bar charts hard to read, but it is also hard to compare across the different bars, especially if the bars of the movies of interest are far apart from each other.
>
> Additionally, the bubble chart's overlapping circles distract from the message. Because each bubble takes up so much space, it is hard to grasp the meaning and difference across the bubbles. Moreover, it is hard to differentiate between the sizes of the circles in the bubble chart, especially between counts of 40 and 60. Some people have a more challenging time perceiving the subtle differences in the size of the same shape, so size may not have been the best encoding option as information is lost when the perception of subtle differences is poor.