Information Visualization I

School of Information, University of Michigan

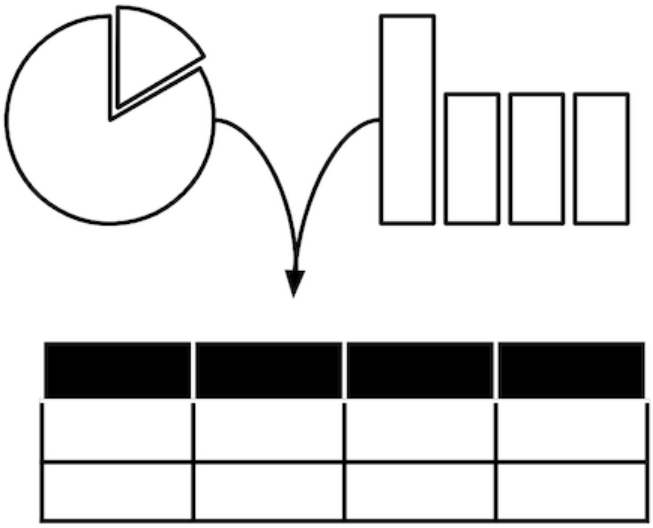
Week 2:

- 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 (https://altair-viz.github.io/)

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 (https://fivethirtyeight.com) available online (https://projects.fivethirtyeight.com/next-bechdel/) (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 (./assets)
  - The original dataset can be found on FiveThirtyEight Next Bechdel Dataset (https://github.com/fivethirtyeight/data/tree/master/next-bechdel)

Important notes:

1) Grading for this assignment is entirely done by a human grader. They will be running tests on the functions we ask you to create. This means there is no autograding (submitting through the autograder will result in an error). You are expected to test and validate your own code.

2) 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.

3) Keep your notebooks clean and readable. If your code is highly messy or inefficient you will get a deduction.

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

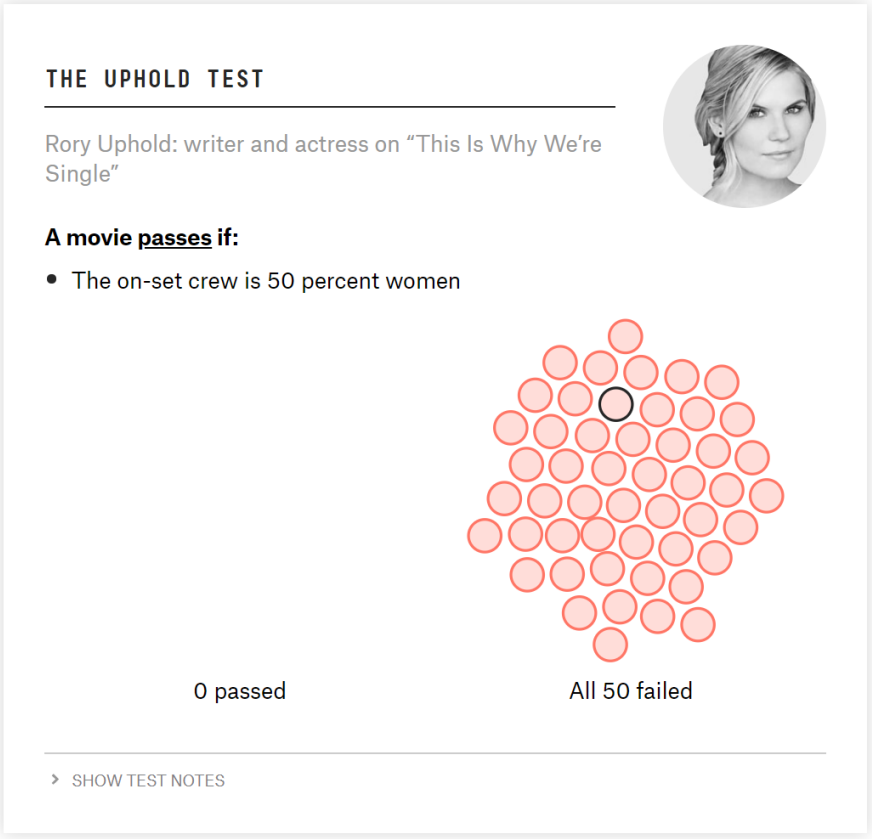
5) Follow the instructions for submission on Coursera. You will be providing us a generated link to a read-only version of your notebook and a PDF. When turning in your PDF, please use the File -> Print -> Save as PDF option ***from your browser***. Do ***not*** use the File->Download as->PDF option. Complete instructions for this are under Resources in the Coursera page for this class. If you're having trouble with printing, take a look at this video (https://youtu.be/PiO-K7AoWjk).

Part 1. Expressiveness and Effectiveness (30 points)

Read the following article *Creating the next Bechdel Test* (https://projects.fivethirtyeight.com/next-bechdel/) 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 (./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)`

1.1 Answer

B12 | *fx*

	A	B	C
1	**THE UPHOLD TEST**		
2			
3	**Movie Title**	**On-Set Crew Female Count**	**On-Set Crew Count Total**
4	Bad Moms	8	28
5	Hidden Figures	12	32
6	Finding Dory	9	22
7	Arrival	4	19
8	The Conjuring 2	2	25
9			

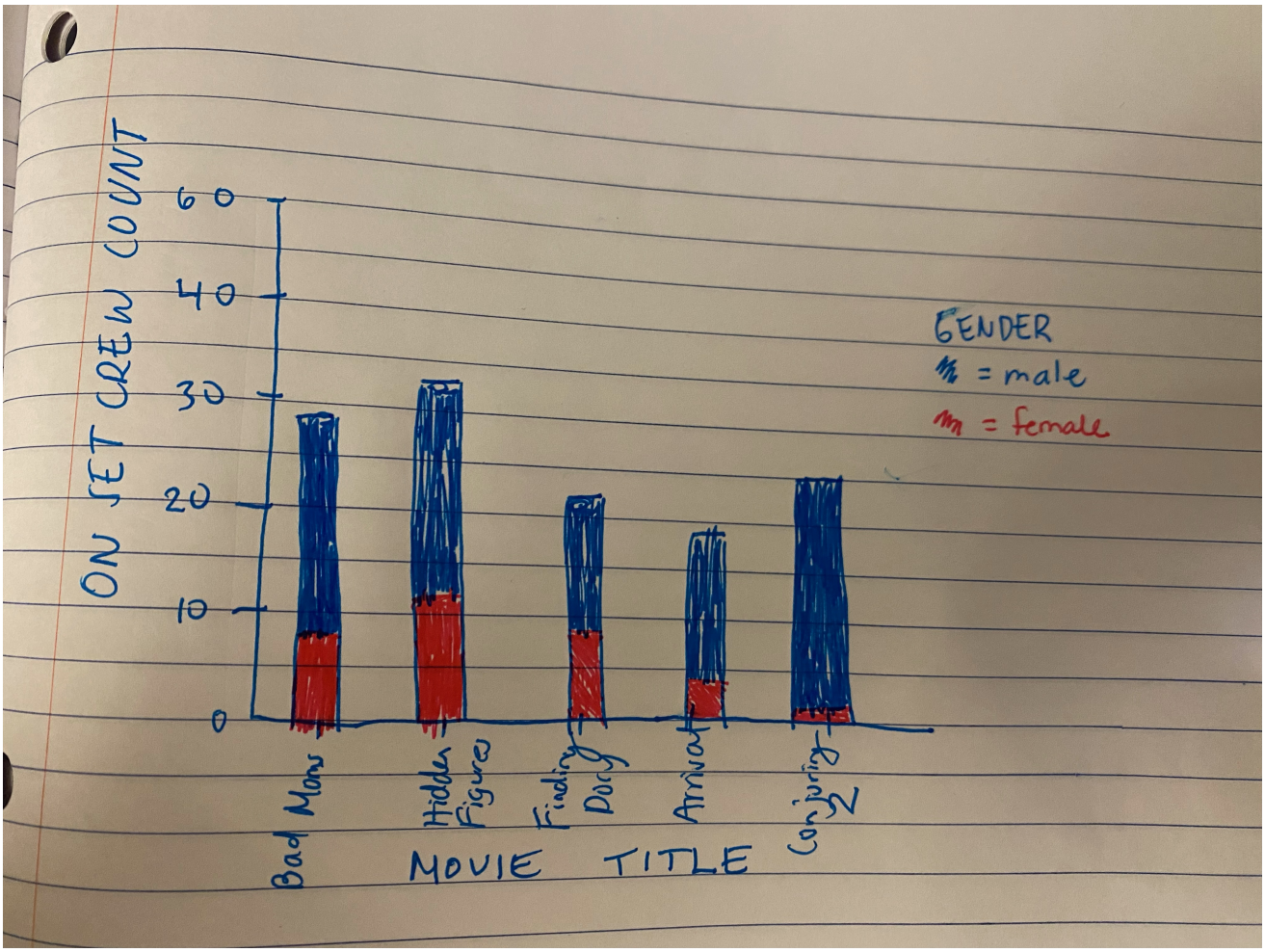
Note about 1.1

- Do not need a passed or failed column. Would just set a condition to say if 'On-Set Crew Female Count' divided by 'On-set Crew Count Total' is not greater than or equal to 0.5, movie failed.

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.

1.2 Answer



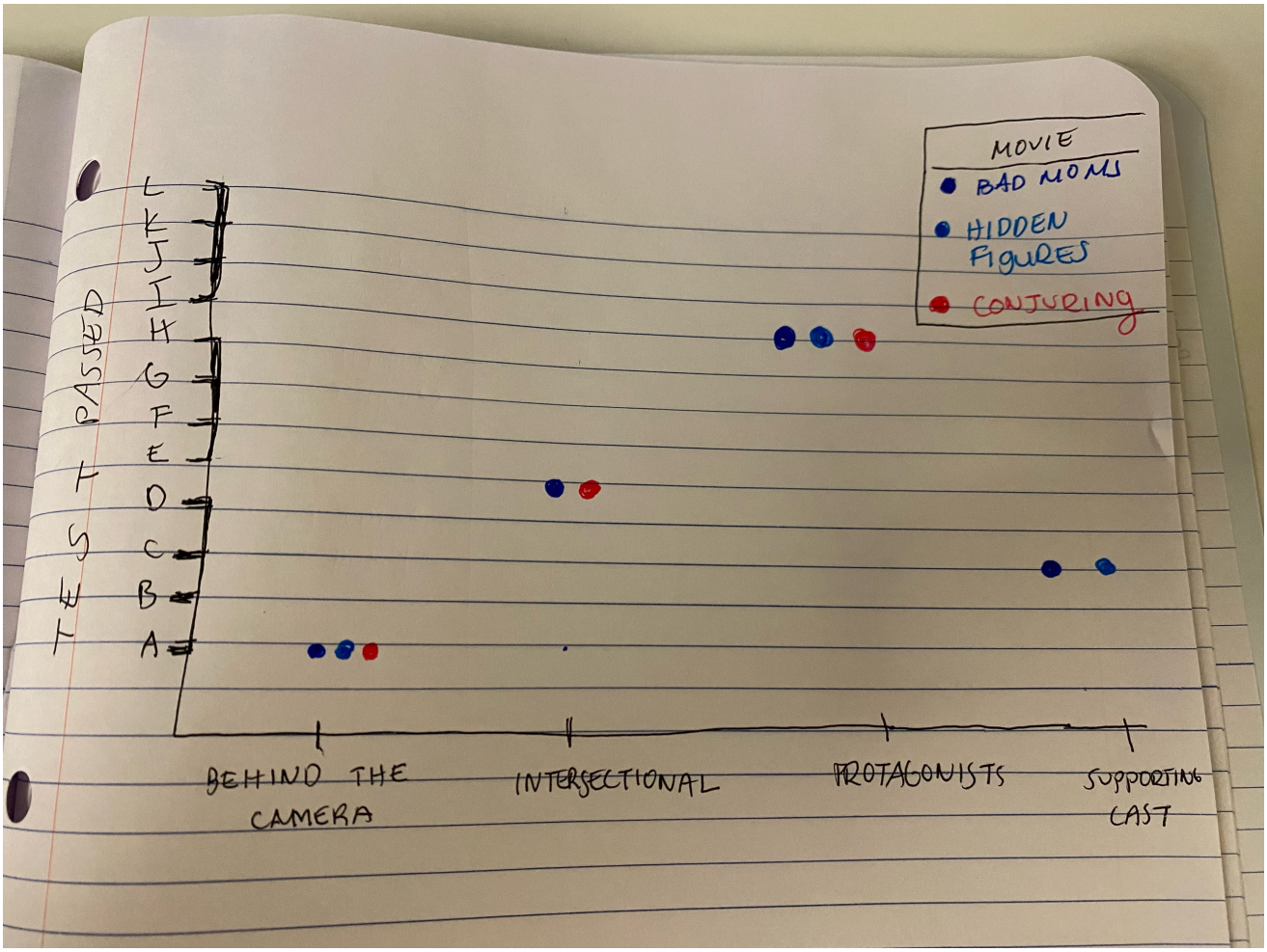
- my image lacks a clear indication of the fact that there is a test.
- I would add an icon at the top of each stacked bar with a P or F and I would add that to the legend

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

	TEST												
	BEHIND THE CAMERA				INTERSECTIONAL			PROTAGONISTS			SUPPORTING CAST		
MOVIES													
Bad Moms	✓				✓	✓		✓	✓	✓	✓	✓	
Hidden Figures	✓				✓	✓		✓	✓	✓			✓
Independence Day: Resurgence	✓				✓	✓		✓	✓			✓	✓
Finding Dory	✓		✓					✓	✓		✓		✓
Ghostbusters	✓					✓		✓	✓	✓			✓
Allegiant	✓					✓		✓	✓				✓
Arrival	✓							✓	✓			✓	✓
Ice Age: Collision Course	✓		✓			✓				✓		✓	

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.

1.3 Answer



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.

1.4 Answer

The original image in 1.3 is more effective for the following reasons:

1) Ease of determining if the test was passed or not (check mark) 2) Color coding per 'Behind the Camera', 'Intersectional', 'Protagonist', and 'Supporting Cast'

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 films 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]:
```
def load_bechdel_data(alldata='assets/nextBechdel_allTests.csv',
                      castgenderdata='assets/nextBechdel_castGender.csv',
                      top2016data='assets/top_2016.csv'):
    # read all the tables
    all_tests_df = pd.read_csv(alldata)
    cast_gender = pd.read_csv(castgenderdata)
    top_2016 = pd.read_csv(top2016data)

    # set up the tables for use
    act_movies = top_2016.set_index('Movie').join(cast_gender.set_index('MOVIE')).join(all_tests_df.set_i
    mov_order = top_2016.sort_values(by=['Rank'])['Movie'].tolist()
    return(act_movies,mov_order)
```

In [5]:
```
actors_movies,movies_order = load_bechdel_data()
```

2.1 Variables encoded (5 points)

Warmup: how many variables are encoded in the following visualization?

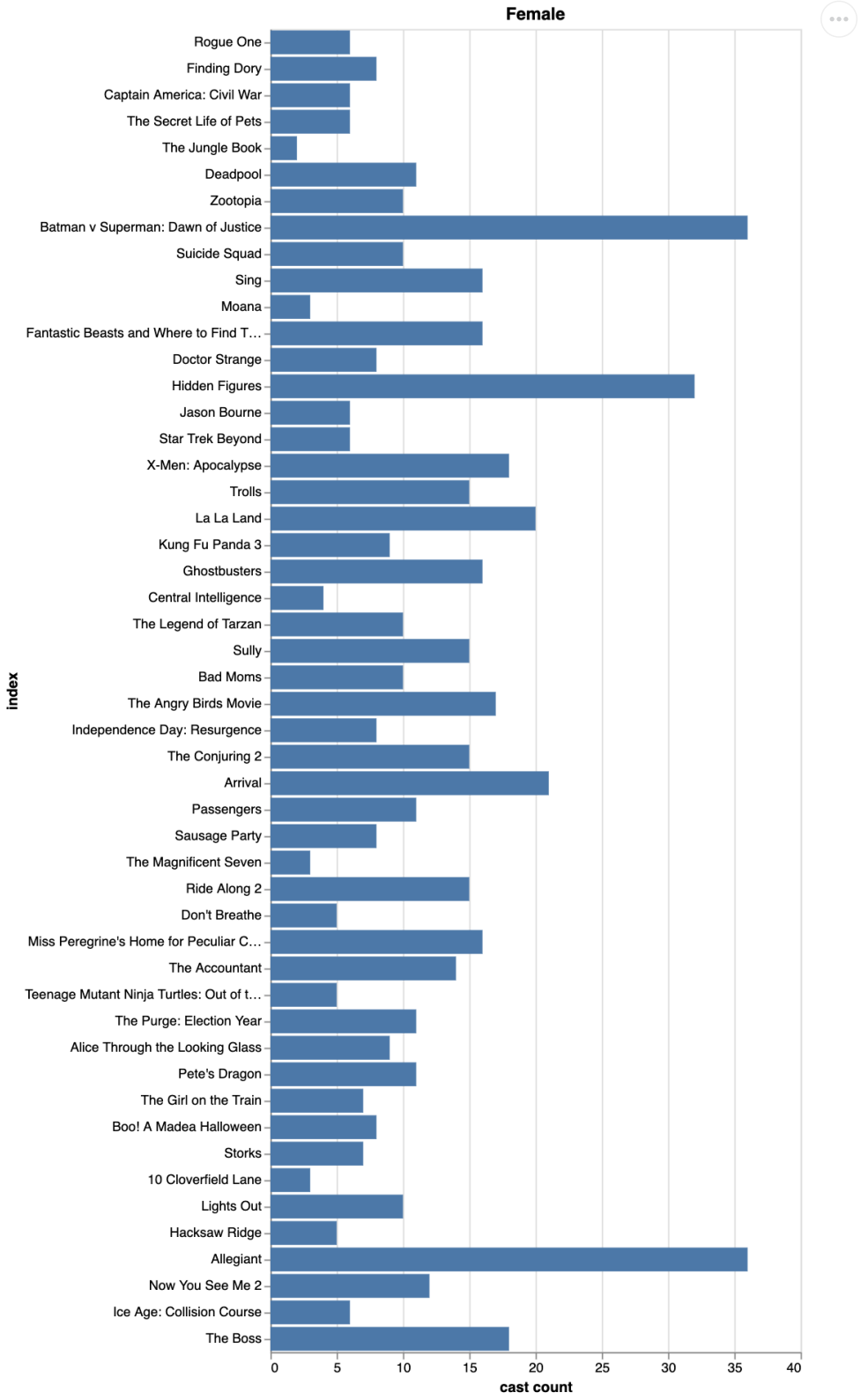
We also suggest you look closely this code and make sure you understand what we're doing. Most of the other problems below will follow this structure.

In [6]:
```
def get_base_vis(indf):
    #input: indf, the base chart dataset (i.e., actors_movies)

    # this is a "base" chart -- on its own, it will not display anything because we haven't defined a ma
    # but we can do operations that we'll want to do multiple times, in this case filtering some data
    return alt.Chart(indf).transform_filter(
        (alt.datum.TYPE != 'Unknown') & (alt.datum.GENDER != 'Unknown') & (alt.datum.GENDER != 'null')
    )

base_vis = get_base_vis(actors_movies)
```

In [7]:
```
# this is where we build the actual chart
def gen_f_bar_vis(base, indf):
    # input: base is the 'base' vis as produced by get_base_vis
    # indf: a dataframe like actors_movies -- we *could* use this, but it's easier to
    #       use the 'base' chart so we're not repeating work

    # we will "add" to the base chart with an additional transform filter
    encoding = base.transform_filter(
        alt.datum.GENDER == 'Female'
    ).encode(
        # adding all the encoding stuff
        y= alt.Y(
            'index:N',
            sort= movies_order
        ),
        x=alt.X('count(index):Q',
                title='cast count'),
    ).mark_bar( # use a "bar" symbol
    ).properties(
        # set the title
        title='Female'
    )


    # The less efficient way (where we don't use "base" but repeat work):

    # encoding = alt.Chart(indf).mark_bar().transform_filter(
    #     (alt.datum.TYPE != 'Unknown') & (alt.datum.GENDER != 'Unknown') & (alt.datum.GENDER != 'null')
    # ).transform_filter(
    #     alt.datum.GENDER == 'Female'
    # ).encode(
    #     y= alt.Y(
    #         'index:N',
    #         sort= movies_order
    #     ),
    #     x=alt.X('count(index):Q',
    #             title='cast count'),
    # ).properties(title='Female')
    #

    return encoding

female_bar = gen_f_bar_vis(base_vis,actors_movies)
```

In [8]:
```
female_bar
```

Out[8]:

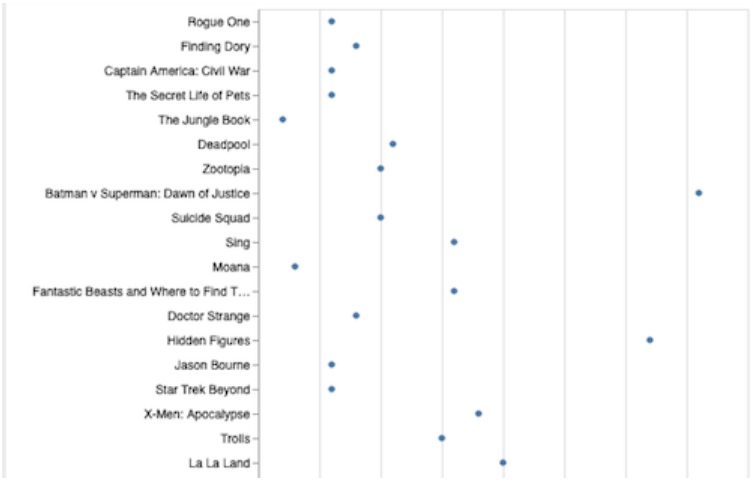


How many variables are encoded in the visualization above? Add your answer below.

2

2.2 Alternative encoding (5 points)

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



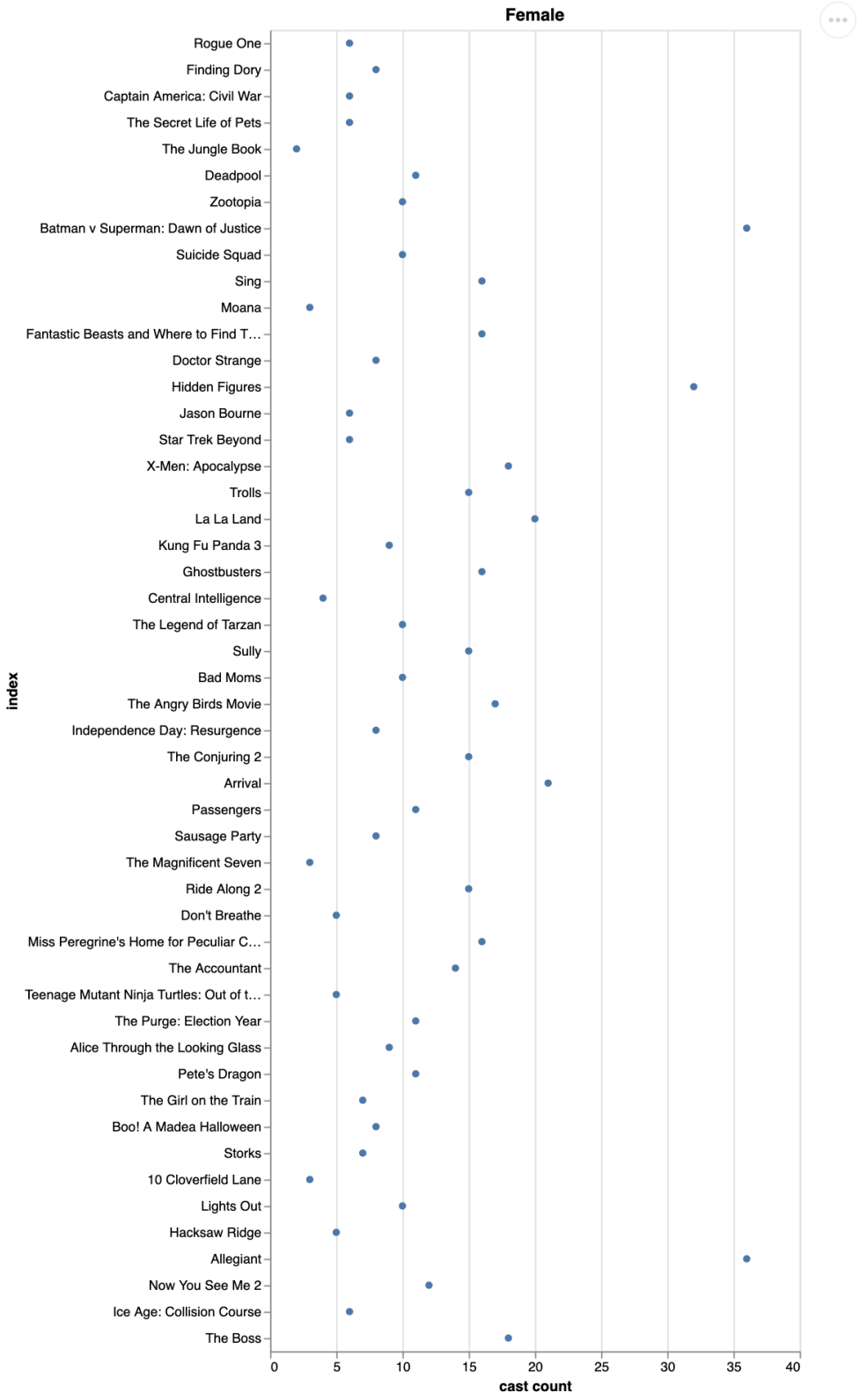
click here (assets/alt_enc_1_resized_fullImage.png) to see the full-sized image.

In [9]:
```
def get_base_vis(indf):
    #input: indf, the base chart dataset (i.e., actors_movies)

    # this is a "base" chart -- on its own, it will not display anything because we haven't defined a ma
    # but we can do operations that we'll want to do multiple times, in this case filtering some data
    return alt.Chart(indf).transform_filter(
        (alt.datum.TYPE != 'Unknown') & (alt.datum.GENDER != 'Unknown') & (alt.datum.GENDER != 'null')
    )

base_vis = get_base_vis(actors_movies)
```

In [10]:
```
def gen_f_circle_vis(base, indf):
    """
    input: base -- 'base' chart as defined above
    input: indf -- a dataframe like actors_movies (you can use this or 'base' to achieve the desired wor

    #return the call to altair function that uses the circle mark for the variables encoded in the previ
    """
    encoding = base.transform_filter(
        alt.datum.GENDER == 'Female'
    ).encode(
        # adding all the encoding stuff
        y= alt.Y(
            'index:N',
            sort= movies_order
        ),
        x=alt.X('count(index):Q',
                title='cast count'),
    ).mark_circle( # use a "circle" symbol
    ).properties(
        # set the title
        title='Female'
    )

    return encoding

circle = gen_f_circle_vis(base_vis,actors_movies)
```

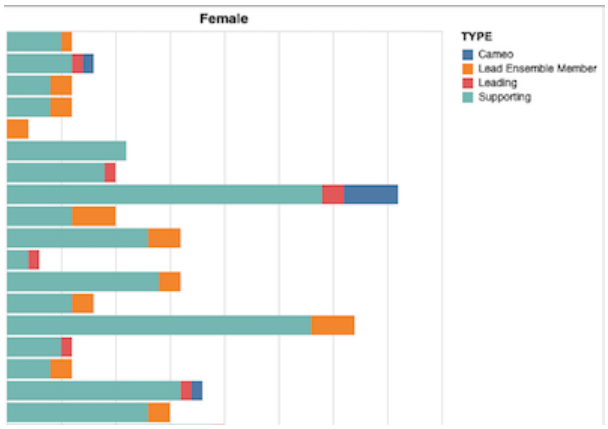
In [11]:
```
# generate and display the vis
circle = gen_f_circle_vis(base_vis, actors_movies)
circle
```

Out[11]:



2.3 Increase variables encoded (5 points)

Complete the `gen_f_stacked_vis` 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):



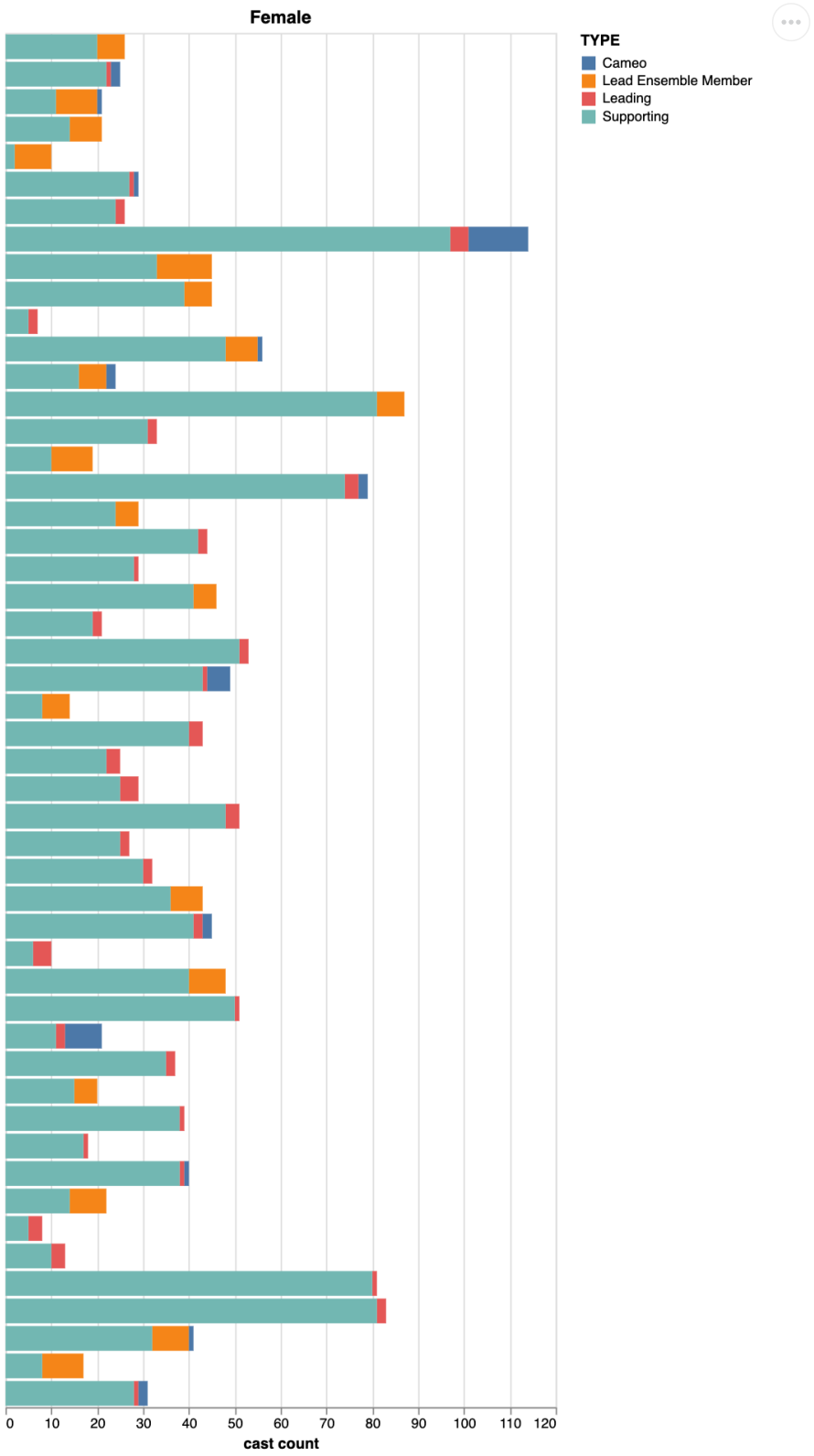
click here (assets/alt_enc_2_fullSize.png) to see the full-sized image.

- *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 [12]:
```
def gen_f_stacked_vis(base, indf):
    """
    input: base -- 'base' chart as defined above
    input: indf -- a dataframe like actors_movies (you can use this or 'base' to achieve the desired worl

    return the call to Altair function that uses the bar mark for the variables and the color for the TYI
    """

    # this is the original chart, you should replace this code
    encoding = base.encode(
            y= alt.Y(
            'index:N',
                axis =None,
            sort= movies_order
        ),
        x=alt.X('count(index):Q',
                title='cast count'),
        color = 'TYPE',
    ).mark_bar().properties(title='Female')


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

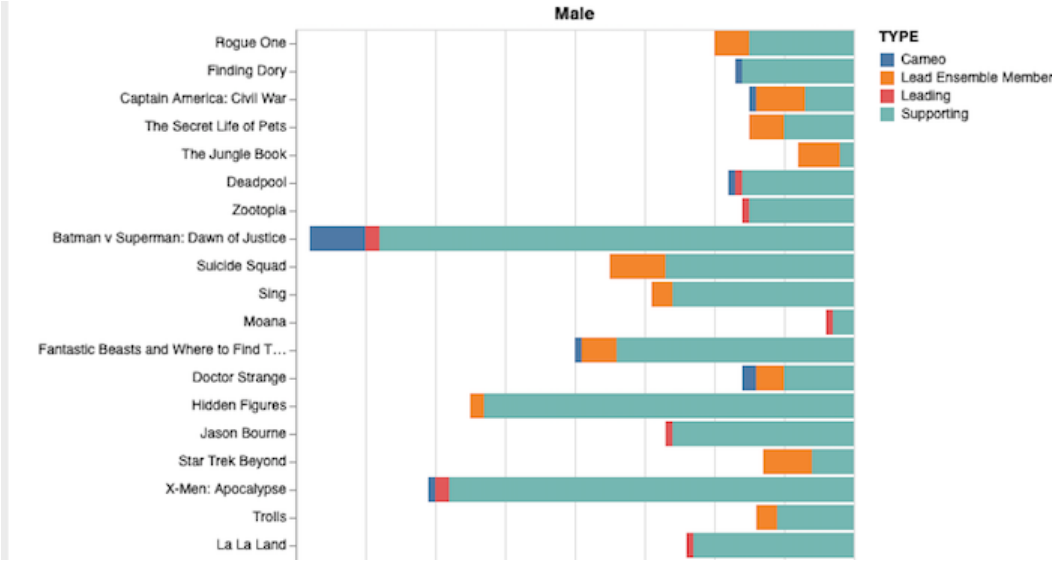
In [13]:
```
female = gen_f_stacked_vis(base_vis, actors_movies)
female
```

Out[13]:



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.



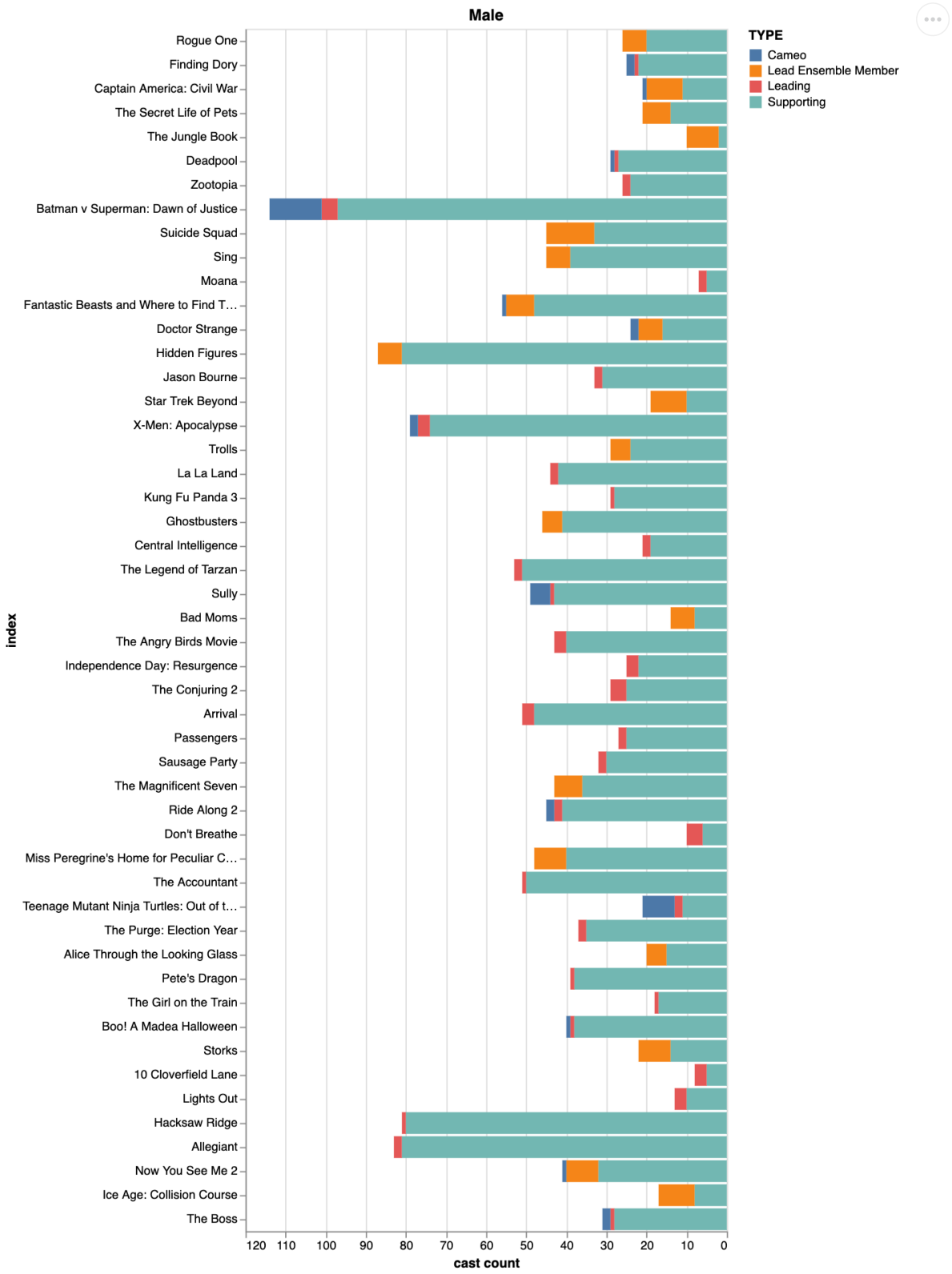
click here (assets/alt_enc_3_fullSize.png) to see the full-sized image.

- *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 [14]:
```
def gen_m_stacked_vis(base, indf):
    """
    input: base -- 'base' chart as defined above
    input: indf -- a dataframe like actors_movies (you can use this or 'base' to achieve the desired worl

    return the call to Altair function that uses the bar mark for the variables and the color for the TYI
    """
    #add filter transform

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

    # YOUR CODE HERE
    #raise NotImplementedError()

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

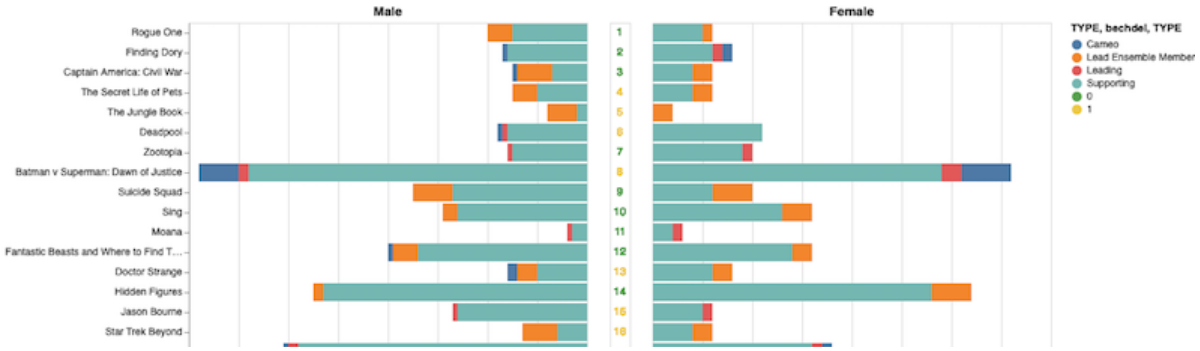
In [15]:
```
male = gen_m_stacked_vis(base_vis, actors_movies)
male
```

Out[15]:



2.5 Variables encoded 2 (5 points)

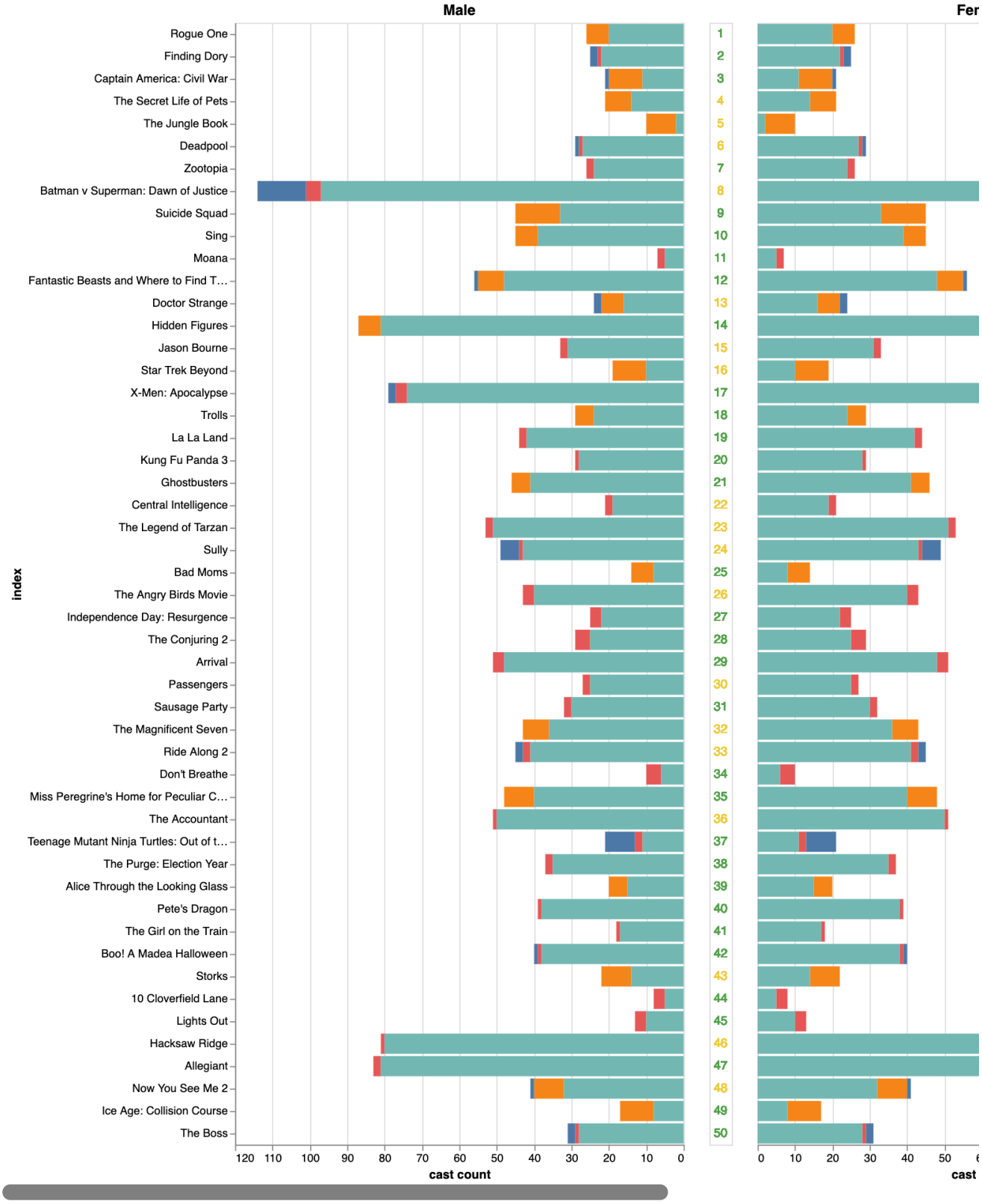
Execute the following cell and determine how many variables are being encoded in the combined visualization. If you have been able to complete the previous examples, the plot should look like this



click here (assets/visualization_fullSize.png) to see the full-sized image.

In [16]:
```
def get_middle_vis(base, indf):
    """
    input: base -- 'base' chart as defined above
    input: indf -- a dataframe like actors_movies (you can use this or 'base' to achieve the desired worl

    return the "middle" column -- a text visualization
    """
    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)
    return(middle)


# merge together the three charts, male, middle, female
middle = get_middle_vis(base_vis, actors_movies)
male | middle | female
```

Out[16]:



How many variables are encoded in the visualization above? (this should be an integer)

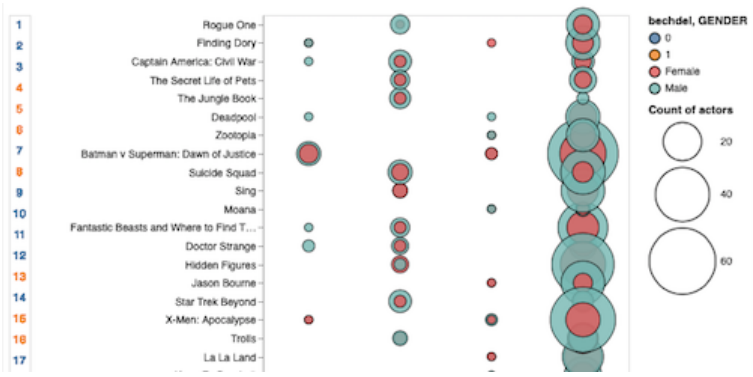
3

2.6 Alternative encoding 1 (20 Points)

Create a new visualization within the `gen_bubble_vis` 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 (it doesn't need to be pixel perfect, but should be close)
- you don't need to generate the leftmost column (it's the same as "middle" above and we'll add it at the end.



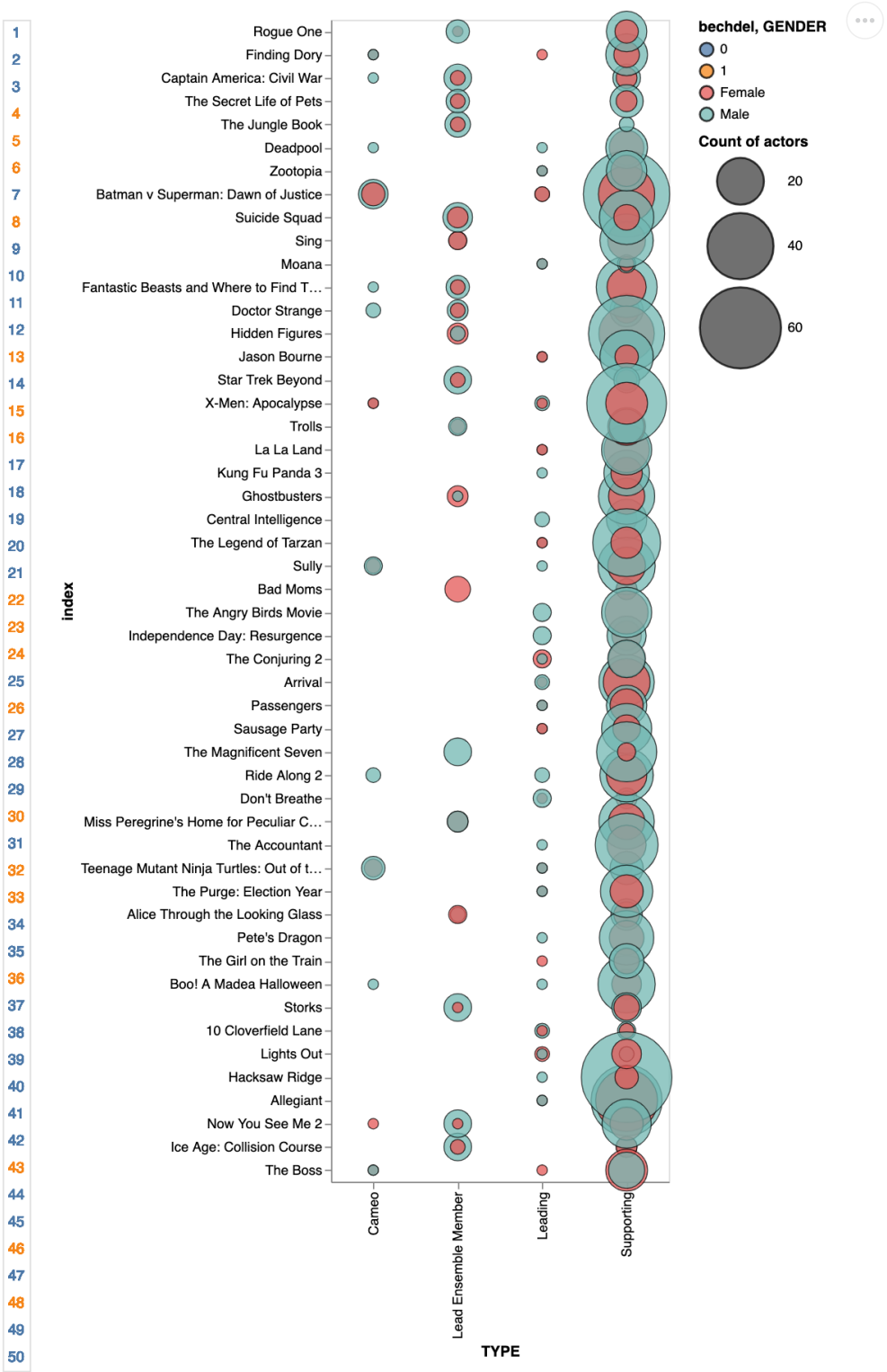
click here (assets/visualization_2_fullSize.png) to see the full-sized image.

- *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 [17]:
```
def gen_bubble_vis(base, indf):
    """
    input: base -- 'base' chart as defined above
    input: indf -- a dataframe like actors_movies (you can use this or 'base' to achieve the desired worl

    return an altair chart per the specification above
    """
    plot = base.mark_circle(
       opacity=0.75,
       stroke='black',
       strokeWidth=0.7
    ).encode(
        alt.Y('index:N',
              sort= movies_order),
        alt.X('TYPE:N'),
        alt.Color('GENDER:N'),
        #color ='GENDER:N',
        alt.Size('count(TYPE):Q',
                         legend = alt.Legend(title='Count of actors'),
                         scale=alt.Scale(range=[0, 4500]))
    ).properties(
        width=250,
        height=860,)


    return plot
```

In [18]:
```
# let's create the bubble chart
al_enc_one = gen_bubble_vis(base_vis, actors_movies)

# add middle to the left edge and display
middle | al_enc_one
```

Out[18]:



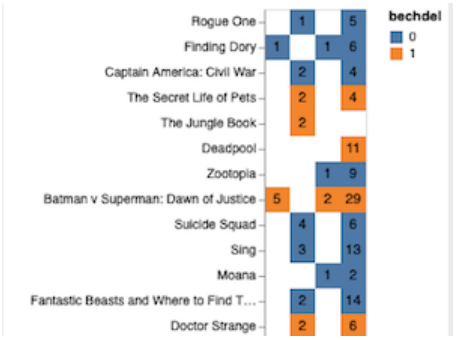
2.7 Alternative encoding 2 (25 Points)

We will be completing the "heat map" style vis as specificed below, but it will be easier to create a male and female subchart in two separate functions (the code will get long otherwise) and then put them together. Complete `gen_f_hm_vis()` and `gen_m_hm_vis()` functions to create a new visualization with the following encoding:

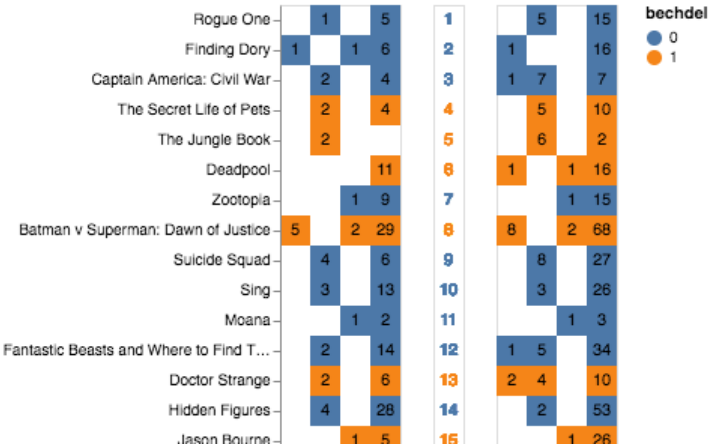
- The left and right plot should filter male and female actors respectively

- 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)
- 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):



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).

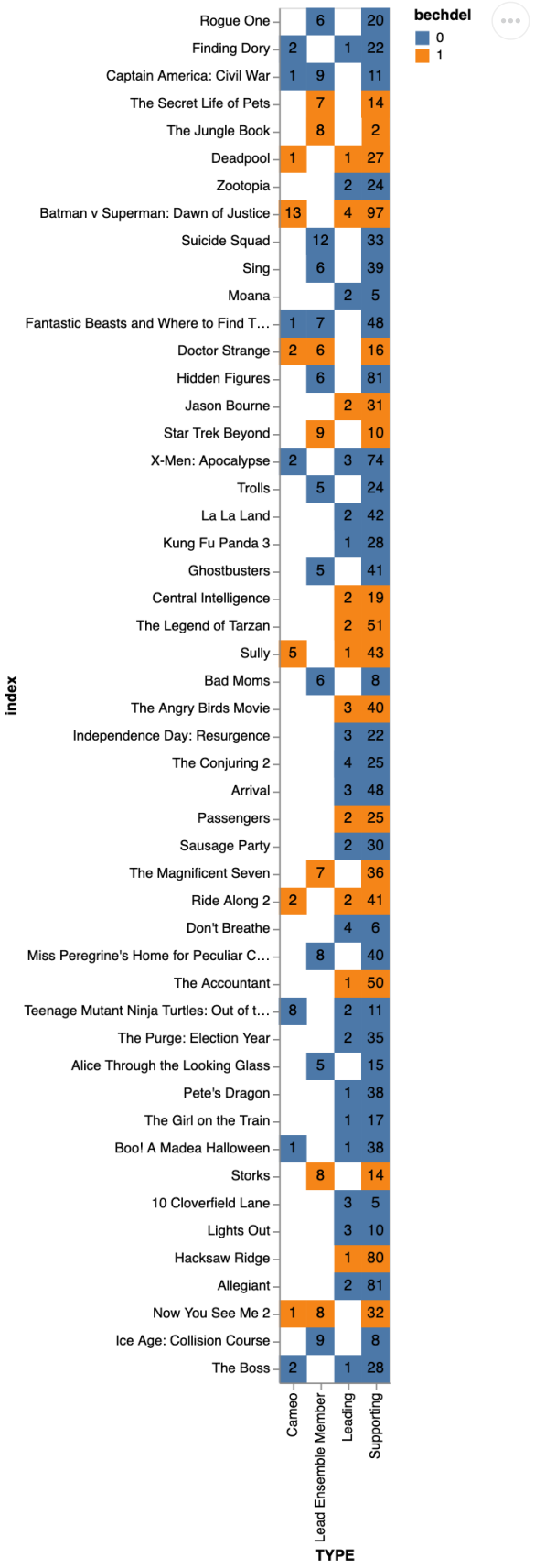


- *Partial credit can be granted for each visualization (up to 4 points for each function) if you provide a description of what the missing piece of the function is supposed to do without need for an Altair working version*

In [19]:
```
def gen_f_hm_vis(base, indf):
    """
    input: base -- 'base' chart as defined above
    input: indf -- a dataframe like actors_movies (you can use this or 'base' to achieve the desired worl

    return an altair chart per the specification above

    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 text inside each rectangle to encode the count of actors one each category (gender, type and
    """
    # modify to add filter transform
    plot = base.mark_rect().encode(
        alt.X('TYPE:N'),
        alt.Y('index:N',
              sort= movies_order),
        # add text and color
        alt.Color('bechdel:N'),

    )
    #modify to add filter transform
    text = base.mark_text(baseline='middle').encode(
        x='TYPE:N',
        y= alt.Y(
            'index:O',
            sort= movies_order,
             #axis=None
        ),
        #change this
        #text='TYPE:Q'
        text = 'count(GENDER):Q',

    )
    # YOUR CODE HERE
    #raise NotImplementedError()
    return plot + text
```

In [20]:
```
# create the female side
f_a_1 = gen_f_hm_vis(base_vis, actors_movies)

# display it to check
f_a_1
```

Out[20]:



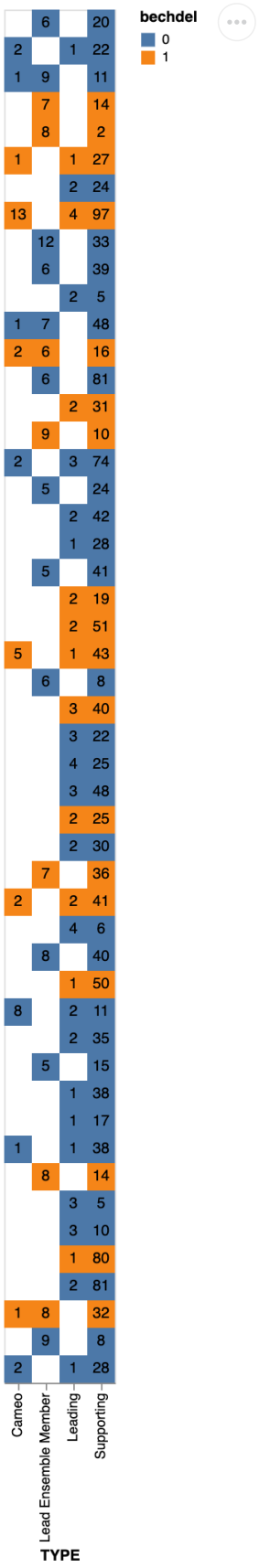
MISSING: Correct number inside of rectangles (count of actors one in each category (gender, type and movie) -- this mistake is carried

through the rest of the assignment

In [21]:
```
def gen_m_hm_vis(base, indf):
    """
    input: base -- 'base' chart as defined above
    input: indf -- a dataframe like actors_movies (you can use this or 'base' to achieve the desired wor

    return an altair chart per the specification above
    """
    #add filter transform
    plot = base.mark_rect().encode(
        alt.X('TYPE:N'),
        alt.Y('index:N',
              sort= movies_order),
        alt.Color('bechdel:N'),
        # 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(TYPE):Q',
    )
    # YOUR CODE HERE
    #raise NotImplementedError()
    return plot + text
```

In [22]:
```
m_a_1 = gen_m_hm_vis(base_vis, actors_movies)
m_a_1
```

Out[22]:



In [23]:
```
# create the visualization
f_a_1 | middle | m_a_1
```

Out[23]:

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?

2.8 Answer

- The 2.7 visualization is a lot cleaner and more readable than those of 2.5 and 2.6.
- The 2.7 visualization allows onlookers to get the exact number of actors per movie, where 2.5 and 2.6 allow for an estimate.
- 2.7 is not more effective in distinguishing gender. Lacking M/F lables, where 2.5 and 2.6 have gender by color or label.

In [ ]: