Information Visualization II

School of Information, University of Michigan

Week 2:

- Functions of interactivity

Assignment Overview

The objectives for this week are for you to:

- Understand the role of interaction in visualization
- Identify and understand various types of interaction in visualization
- Learn to implement interactive visualizations using Altair

The total score of this assignment will be

- Case study reflection: (30 points)
- Altair programming exercise (70 points)
- Extra credit (up to 10 points)

Resources:

- This article by FiveThirtyEight (https://fivethirtyeight.com). Available online (https://fivethirtyeight.com/features/women-in-comic-books/) (Hichey, 2014)
- 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 Comic Characters (https://github.com/fivethirtyeight/data/tree/master/comic-characters)

Important notes:

1) Grading for this assignment is entirely done by manual inspection.

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.

If you're having trouble with printing, take a look at this video (https://youtu.be/PiO-K7AoWjk).

Part 1. Interactive visualization assesment (30 points)

Read the following article Timeless Songs (https://pudding.cool/2017/03/timeless/) on the Pudding's site, and answer the following questions:

1.1 Identify various interactions (15 points)

For the five visualizations:

- What's Remembered from the 90s
- Biggie or Tupac
- Present-day Popularity of Five Decades of Music
- XXXX Tracks: Historic Billboard Performance vs. 2014 Spotify Plays
- The Long-term Future of Hits from 2013

Identify which of the 7 interaction types are implemented and how. You don't need a long description. A short sentence will do.

- What's Remembered from the 90s
  - **select** is implemented by displaying the name of an artist, the song in question, the number of times users played the song, and the year it was released when the user hovers over a picture of the artist. Here selection supports learning detailed information about a hit song without cluttering the visualization, which helps the user process the chart and makes it more appealing.
  - **filter** is implemented by using a search bar where the user can search for the name of an artist or song, and it will dynamically run the query and shade all non-matching bubbles in a lighter shade to highlight items that match the query. Here filtering supports comparisons of interest by bringing selected items to attention for easy comparison.
- Biggie or Tupac
  - **select** is implemented by displaying the name of an artist, the song in question, the number of times users played the song, and the year it was released when the user hovers over a bubble.
  - **filter** is implemented by using a search bar where the user can search for the name of an artist or song, and it will dynamically run the query and color all bubbles that match the query in red. There is also another option to filter by predefined filters ("All Rappers," "Just Biggie and Tupac," and "Just Jay-Z"), which similarly colors bubbles that match the queries.
- Present-day Popularity of Five Decades of Music
  - **reconfigure** is implemented by changing the bars corresponding to each track once a filter is applied. The opacity of colors also varies depending on the Spotify play counts as the filter changes; this provides a visual cue for the song's popularity relative to other songs in the filtered set.
  - **filter** is implemented by using a search bar where the user can search for the name of an artist or song, and it will dynamically run the query and remove all tracks from the table that do not match.
- XXXX Tracks: Historic Billboard Performance vs. 2014 Spotify Plays
  - **filter** is implemented by using a search bar where the user can search for the name of an artist or song, and it will dynamically run the query and shade all non-matching bubbles in a lighter shade to highlight items that match the query.

    - **selection** is implemented by displaying the name of an artist, the song in question, the number of times the song was played, and the year it was released when the user hovers over a bubble.
    - **reconfigure** is implemented by rearranging the bubbles corresponding to each track once a filter is applied.
- The Long-term Future of Hits from 2013
    - **filter** is implemented by selecting boxes for tracks to display on the time series chart. The user can choose songs they would like to compare, and all other time-series plots (for all other songs) will not be visible. Filtering is also implemented through a dynamic search bar where the user can search for the name of an artist or song, and it will dynamically run the query and limit the check box selection options.

1.2 Critique (15 points)

For one of the five visualizations, critique the use of interaction. What works well? What could be better? You can add your own images here if it helps.

XXXX Tracks: Historic Billboard Performance vs. 2014 Spotify Plays

What works well?

- The implementation of filtering is thorough and supports lots of interesting queries relevant to the purpose of the visualization.
- The tooltip that appears when hovering over bubbles provides as many details as are necessary for a selected track and nothing extra.

What could be better?

- The implementation of reconfigure could be improved. Specifically, I do not like the animation that occurs when you add a filter or selection. The bubbles jump around and do so quickly, so it is hard to gather any insight from the animation. It appears cartoonish and pointless. It would be best to use a still frame; to reconfigure the data points without any animation.

Part 2. Programming exercise (70 points)

Start by reading the 538 article here (https://fivethirtyeight.com/features/women-in-comic-books/). What you should know is that there are two major comic book companies: DC (Batman, Superman, Wonder Woman, etc.) and Marvel (Black Widow, Iron Man, Hulk, etc.).

We have a dataset of characters, their sex, when they were introduced, if their identify is secret, their eye and hair color, the number of appearances, etc. Lots of dimensions on which to build our visualizations.

In [1]:
```
# start with the setup
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')

# use the 538 theme
alt.themes.enable('fivethirtyeight')
```

Out[3]:
```
ThemeRegistry.enable('fivethirtyeight')
```

In [4]:
```
# load up the two datasets, one for Marvel and one for DC
dc = pd.read_csv('assets/dc-wikia-data.csv')
marvel = pd.read_csv('assets/marvel-wikia-data.csv')
```

In [5]:
```
dc.head(3) # shape = (6896, 13)
```

Out[5]:

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	SEX
0	1422	Batman (Bruce Wayne)	\/wiki\/Batman_(Bruce_Wayne)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Male Characters
1	23387	Superman (Clark Kent)	\/wiki\/Superman_(Clark_Kent)	Secret Identity	Good Characters	Blue Eyes	Black Hair	Male Characters
2	1458	Green Lantern (Hal Jordan)	\/wiki\/Green_Lantern_(Hal_Jordan)	Secret Identity	Good Characters	Brown Eyes	Brown Hair	Male Characters

In [6]:
```
marvel.head(3) # shape = (16376, 13)
```

Out[6]:

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	
0	1678	Spider-Man (Peter Parker)	\/Spider-Man_(Peter_Parker)	Secret Identity	Good Characters	Hazel Eyes	Brown Hair	C
1	7139	Captain America (Steven Rogers)	\/Captain_America_(Steven_Rogers)	Public Identity	Good Characters	Blue Eyes	White Hair	C
2	64786	Wolverine (James \"Logan\" Howlett)	\/Wolverine_(James_%22Logan%22_Howlett)	Public Identity	Neutral Characters	Blue Eyes	Black Hair	C

In [7]:
```
# create publisher column
dc['publisher'] = 'DC'
marvel['publisher'] = 'Marvel'
```

In [8]:
```
# rename some columns
marvel.rename(columns={'Year': 'YEAR'}, inplace=True)
```

In [9]:
```
# create the table with everything
comic = pd.concat([dc, marvel])

# drop years with na values
comic.dropna(subset=['YEAR'], inplace=True)
```

In [10]:
```
# let's look inside
comic.sample(5)

# this next line sub-samples the data if you want to experiment with
# a smaller dataset. This should only be used for testing. and should
# be commented back in after (otherwise your results won't match the
# images)
# comic = comic[comic.index % 5 == 0]
```

Out[10]:

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	
237	9907	Gunner MacKay (New Earth)	\/wiki\/Gunner_MacKay_(New_Earth)	Public Identity	Good Characters	Blue Eyes	Blond Hair	Char
9059	1988	Aelfyre Whitemane (Earth-616)	\/Aelfyre_Whitemane_(Earth-616)	Secret Identity	Good Characters	Pink Eyes	White Hair	Char
11033	485418	Herr Ekker (Earth-616)	\/Herr_Ekker_(Earth-616)	Public Identity	Bad Characters	NaN	White Hair	Char
15751	91896	Stuart Sarris (Earth-616)	\/Stuart_Sarris_(Earth-616)	Secret Identity	Neutral Characters	NaN	NaN	Char
5722	169312	DeHalle (Earth-616)	\/DeHalle_(Earth-616)	Secret Identity	Bad Characters	Grey Eyes	White Hair	F Char

Comic Books Are Still Made By Men, For Men And About Men

Original article available at [FiveThirtyEight (https://fivethirtyeight.com/features/women-in-comic-books/)](https://fivethirtyeight.com/features/women-in-comic-books/)

By [Walt Hickey (https://fivethirtyeight.com/contributors/walt-hickey/)](https://fivethirtyeight.com/contributors/walt-hickey/)

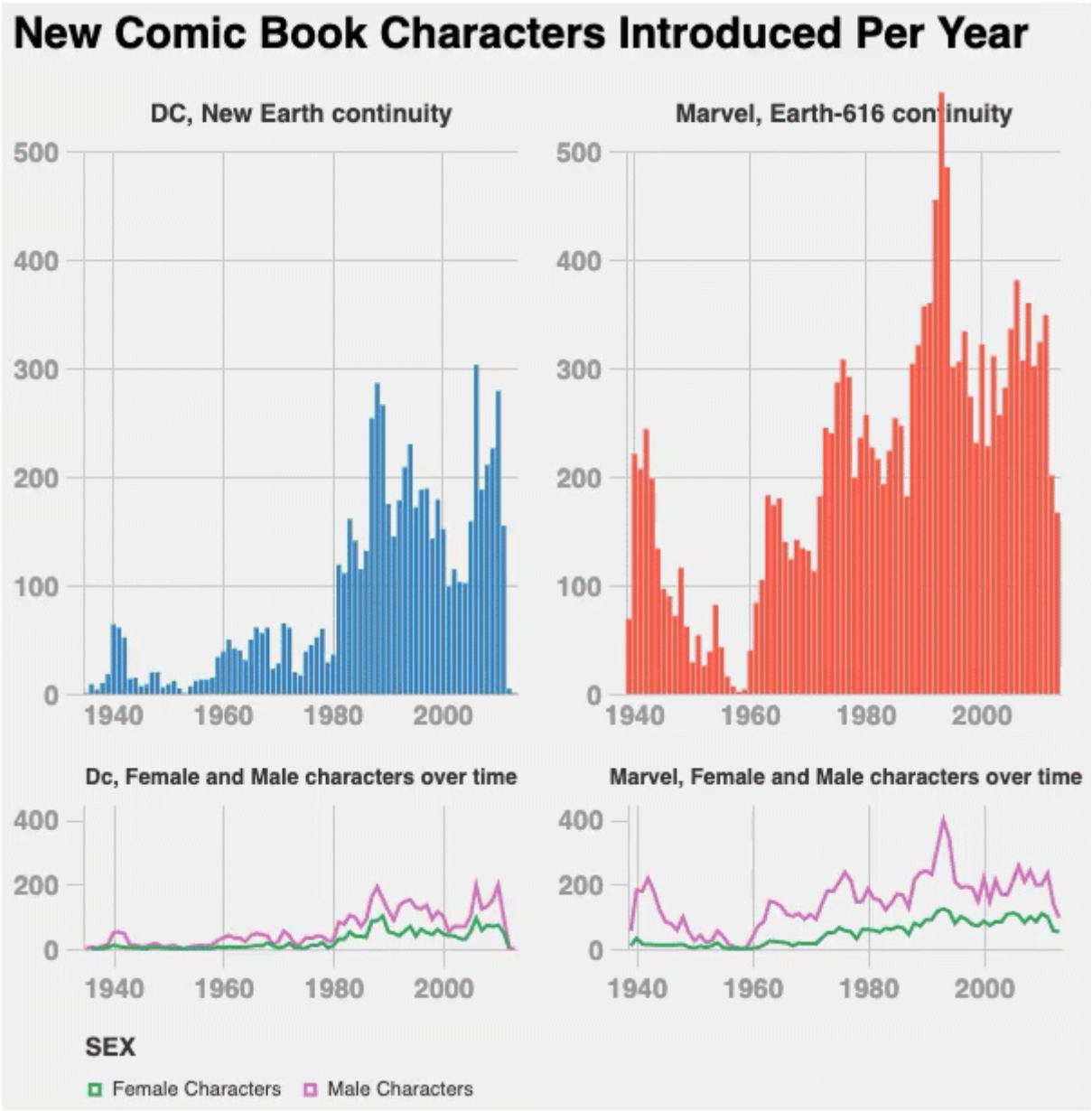
Get the data on [GitHub (https://github.com/fivethirtyeight/data/tree/master/comic-characters)](https://github.com/fivethirtyeight/data/tree/master/comic-characters)

We are going to be revising and adding to the visualizations for this article. While they're nice, we think we can do better by adding some interactivity.

Because many of these visualizations are interactive, we will be recording short clips to demonstrate the desired behavior of the systems. Unlike some of your previous assignments, we will give you portions of the Altair code and ask you to complete the interactive elements.

Problem 2.1 (35 Points)

We'd like to build an interactive visualization that allows us to compare the distributions of characters over time as well. The top two charts will represent the total characters over time (as bar charts). The bottom two will be a line chart with separate lines for female and male characters.



As ranges are selected or moved in the top charts, the bottom charts will automatically update (and the selection will be visible). Selection in one of the top two will also cause the appropriate data to be selected in the other (i.e., select on the left, and the right gets changed (highlighting); select on the right, and the left changes).

You can watch a narrated version of the visualization here (https://www.youtube.com/watch?v=NKCk97yBlJM)

In [11]:
```
gcmic['SEX'].unique()
```

Out[11]:
```
array(['Male Characters', 'Female Characters', nan,
       'Genderless Characters', 'Transgender Characters',
       'Genderfluid Characters', 'Agender Characters'], dtype=object)
```

In [12]:
```
# let's pre-process the data. We're going to focus on just Female and just
gcmic_gb1_df = gcmic[(gcmic['YEAR'] >= 1940) & (gcmic['SEX'].isin(['Female
```

In [13]:
```
# check what's inside
gcmic_gb1_df.sample(5)
```

Out[13]:

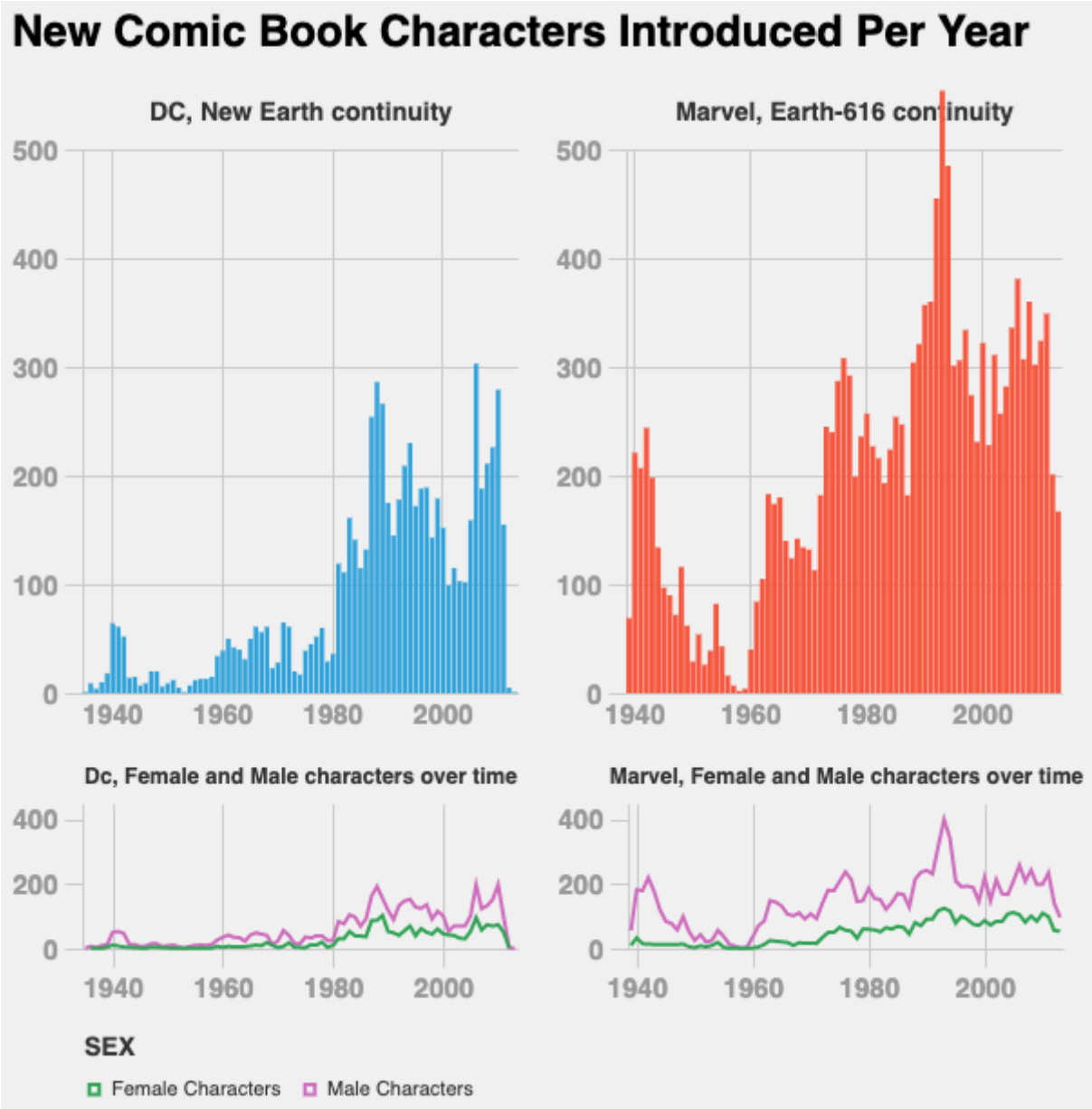
	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	
5532	115456	Gabriel Lan (Heroes Reborn) (Earth-616)	\/Gabriel_Lan_(Heroes_Reborn)_(Earth-616)	NaN	Bad Characters	NaN	NaN	Char
878	125189	Steven Smith (New Earth)	\/wiki\/Steven_Smith_(New_Earth)	Public Identity	Good Characters	Blue Eyes	Blond Hair	Char
2222	80982	Iona Vane (New Earth)	\/wiki\/Iona_Vane_(New_Earth)	Public Identity	Good Characters	Blue Eyes	Red Hair	F Char
432	1555	Amazo (New Earth)	\/wiki\/Amazo_(New_Earth)	Public Identity	Bad Characters	Red Eyes	NaN	Char
4152	364530	Robert Gadling (New Earth)	\/wiki\/Robert_Gadling_(New_Earth)	Public Identity	NaN	Brown Eyes	Brown Hair	Char

In [30]:
```
# we're largely going to use the same "base" visualization here for the bar
# chart and then change the details. The Y axis will be the count()
p1_bar_base = alt.Chart(comic).mark_bar(size=2.5).encode(
    alt.Y('count():Q',
          axis=alt.Axis(
              values=[0, 100, 200, 300, 400, 500],
              title=None,
              labelFontWeight="bold",
              labelFontSize=15
          ),
          scale=alt.Scale(domain=[0, 500]))).properties(
                width=240,
                height=300
)



# let's create the bar chart for DC. We'll take the "base" chart
bar_dc = p1_bar_base.encode(
    alt.X('YEAR:N',  # create the X axis based on year and fix the look of
    axis=alt.Axis(
        values=[1940, 1960, 1980, 2000],
        labels=True,
        ticks=False,
        grid=True,
        title="DC, New Earth continuity",
        titlePadding=-347,
        labelAngle=360,
        labelFontWeight="bold",
        labelFontSize=15,
    )
          ),
).transform_filter(
    # we will use Altair's filter to only keep DC for this chart
    alt.datum.publisher == 'DC'
        )

# let's do the same thing for marvel
bar_marvel = p1_bar_base.mark_bar(color='#f6573f').encode(
    alt.X('YEAR:N', # create the X axis based on year
    # fix the look of the axes
    axis=alt.Axis(
        values=[1940, 1960, 1980, 2000],
        labels=True,
        ticks=False,
        grid=True,
        title="Marvel, Earth-616 continuity",
        titlePadding=-347,
        labelAngle=360,
        labelFontWeight="bold",
        labelFontSize=15
    )
          ),
        ).transform_filter(
            # we will use Altair's filter to only keep DC for this chart
            alt.datum.publisher == 'Marvel'
```

```
        )


# let's create a new "base" chart for the two line charts. We'll take the b
# and modify it to use a line chart
p1_line_base = p1_bar_base.mark_line().encode(
     # the X axis will be year
     alt.X('YEAR:N'),
     # the Y axis will be the count (the number of points that year)
     alt.Y('count():Q',
           axis=alt.Axis(
               grid=False,
               labelFontWeight="bold",
               labelFontSize=15,
               title=None)
          ),
     # let's split the data and color by SEX
     alt.Color('SEX',
          scale = alt.Scale(
              domain=['Female Characters', 'Male Characters'],
              range=['#31a354', '#ce6dbd']
          ),
          legend=alt.Legend(
              orient="bottom")
          )
    ).properties(
        width=240, height=80
     )


line_dc = p1_line_base.encode(
    alt.X(
        'YEAR:N',
        axis=alt.Axis(values=[1940, 1960, 1980, 2000],
        grid=True,
        labelAngle=360,
        labelFontWeight="bold",
        labelFontSize=15,
        title = 'Dc, Female and Male characters over time',
        titlePadding=-130,
        titleFontSize = 12
        )
    )
    ).transform_filter(
        # this is the DC line chart, so we only want DC
        alt.datum.publisher == 'DC'
    )



line_marvel = p1_line_base.encode(
    alt.X(
        'YEAR:N',
        axis=alt.Axis(
            values=[1940, 1960, 1980, 2000],
            grid=True,
            labelAngle=360,
```

```
            labelFontWeight="bold",
            labelFontSize=15,
            title = 'Marvel, Female and Male characters over time',
            titlePadding=-130,
            titleFontSize = 12
            )
        )
    ).transform_filter(
        # this is the Marvel line chart, so we only want Marvel
        alt.datum.publisher == 'Marvel'
    )



# let's put everything together
# top piece
top_charts = alt.hconcat(bar_dc,bar_marvel).resolve_scale(y='shared'
          ).properties(
                   title='New Comic Book Characters Introduced Per Year'
          )

# bottom piece
bottom_charts = alt.hconcat(line_dc,line_marvel).resolve_scale(y='shared')

alt.vconcat(top_charts,bottom_charts).configure_view(
    strokeWidth=0
)
```

Out[30]:



Now we have the chart we need, but here is where you have to start doing some work. For this problem, we'll do this a little bit at a time.

Problem 2.1.1

First, modify the code below to create a "brush" object (a "selection" in Altair speak) that will let us select a time range. For all these, you should take a look at the examples on [this page (https://altair-viz.github.io/user_guide/interactions.html)](https://altair-viz.github.io/user_guide/interactions.html) to identify the right (and the lab).

In [15]:
```
# modify this cell to create the brush object
brush = alt.selection_interval()
```

Problem 2.1.2

The next step is to create the condition for the DC chart. Look at the documentation for the condition. We specifically want things selected by the "brush" object to stay the same color (#2182bd) and the unselected content to turn gray.

In [16]:
```
# modify this cell to create the brush object
colorConditionDC = alt.condition(brush, alt.value('#2182bd'), alt.value('ligh
```

Problem 2.1.3

Finally, we need to add both the condition and selection to the `bar_dc` chart. We'll call this new chart `i_bar_dc` (i for interactive). Remember that you can "override" or modify a chart by simply taking the original chart and adding an encode or some other function to it. For example the line:

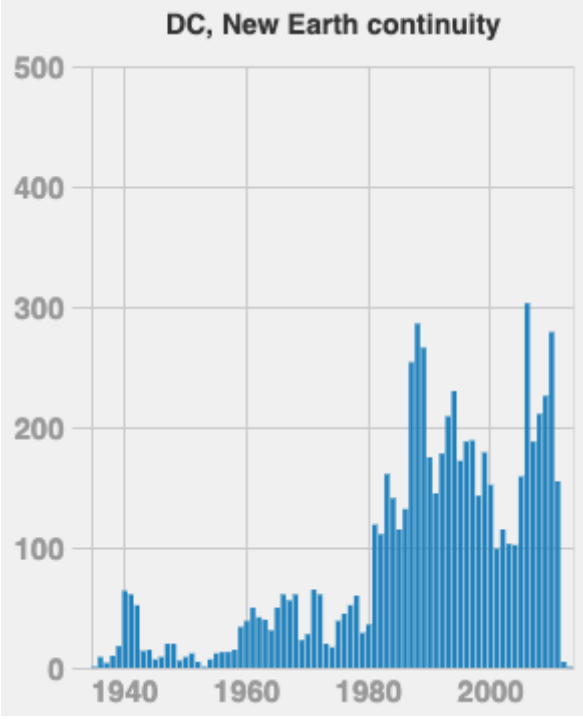
```
i_bar_dc = bar_dc.encode(color = 'TEST')
```

will take the original chart with all its original settings and make the color encoding based on the TEST column (which doesn't exist in this case). If there was a color encoding in `bar_dc`, it will be overridden by TEST. If there wasn't one, it will be added.

In [17]:
```
# modify this cell to create the brush object
i_bar_dc = bar_dc.encode(
    color=colorConditionDC
).add_selection(brush)
```

In [18]:
```
# if you did the last step correctly, you should be able to see the selecti
i_bar_dc
```

Out[18]:

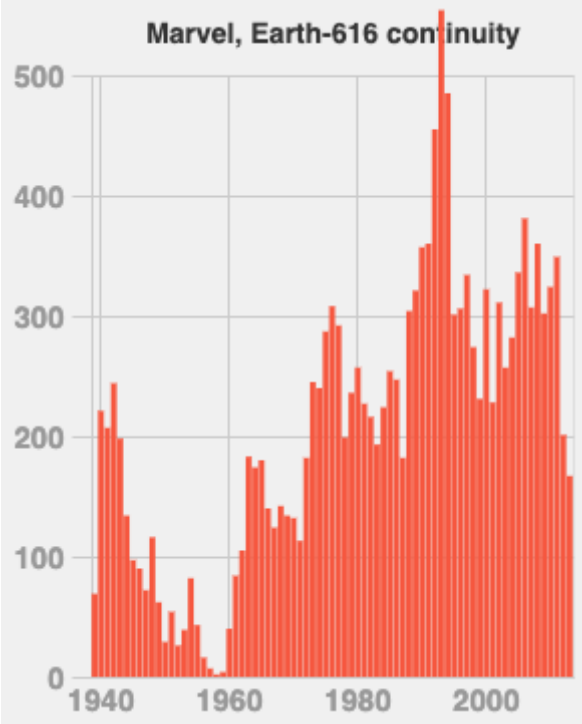


Problem 2.1.4

Do the same thing for the marvel chart. Create the color condition for marvel (selected should be #f6573f, unselected should be gray). Then add the brush and condition to the `bar_marvel` to create `i_bar_marvel`

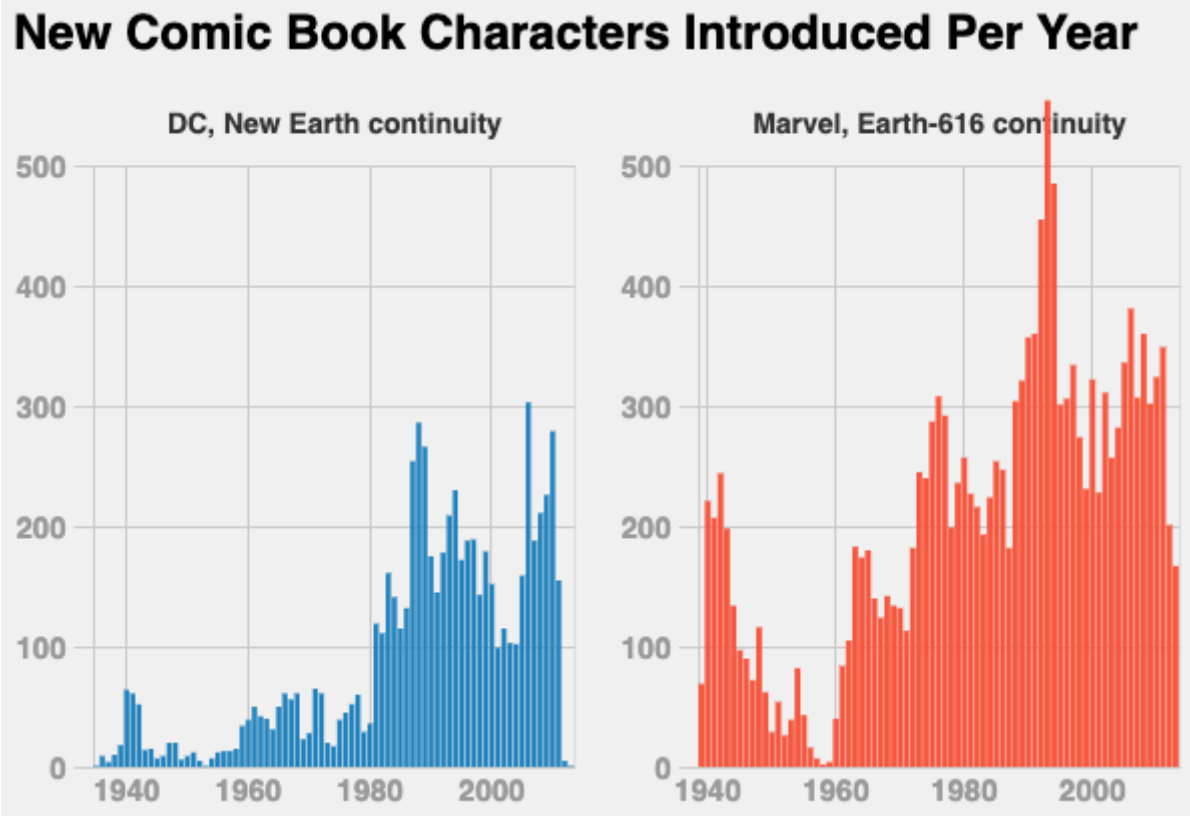
In [19]:
```
# modify the following two lines
colorConditionMarvel = alt.condition(brush,alt.value('#f6573f'),alt.value('
i_bar_marvel = bar_marvel.encode(
    color=colorConditionMarvel
).add_selection(brush)

i_bar_marvel
```

Out[19]:



In [20]:
```
# top piece
top_charts = alt.hconcat(i_bar_dc,i_bar_marvel).resolve_scale(y='shared'
           ).properties(
                    title='New Comic Book Characters Introduced Per Year'
           )

# if you did the two bar charts correctly, you should now be able to intera
# (and the selections should be linked)
```

Out[20]:



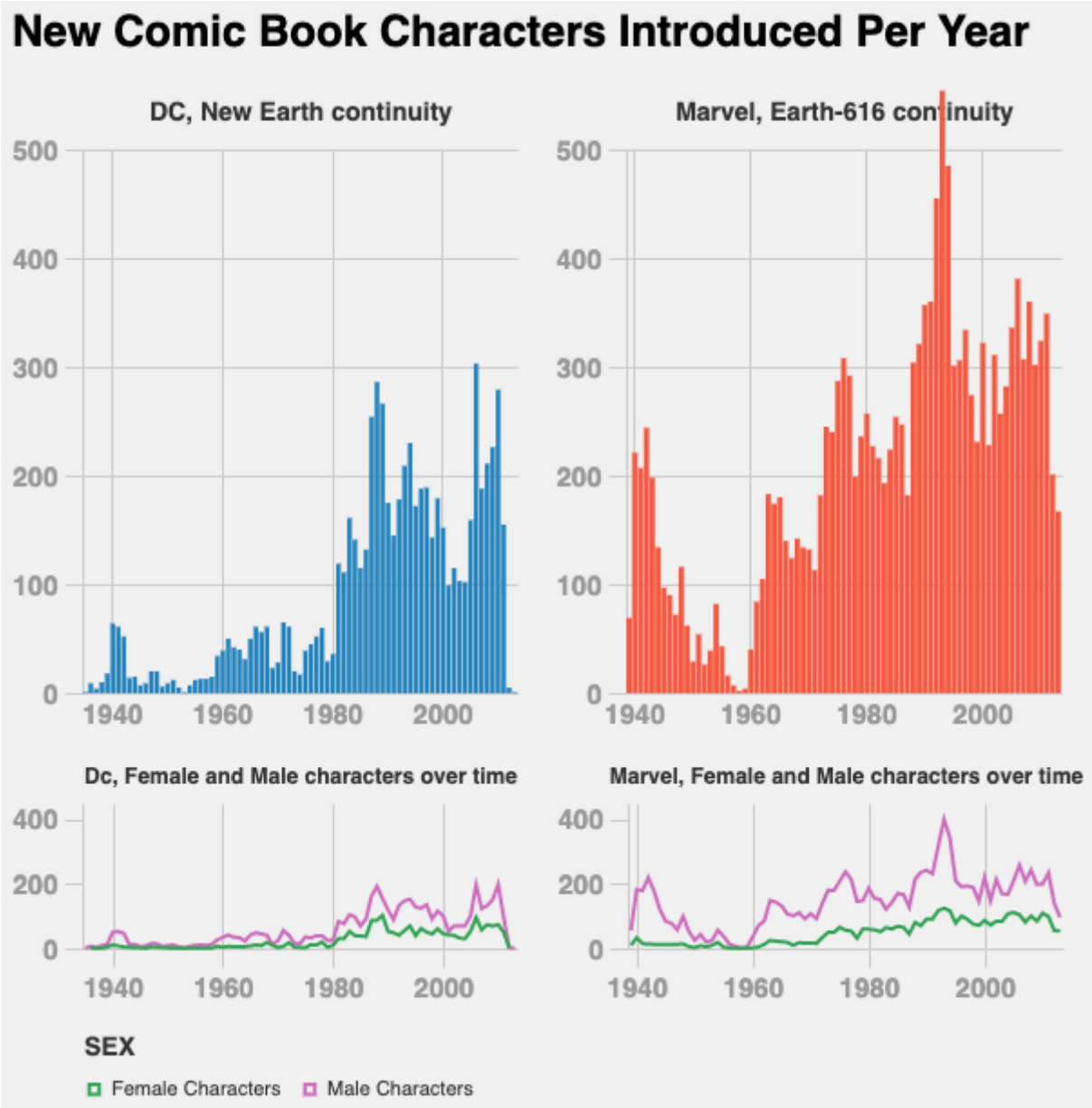
Problem 2.1.5

The last step is to modify the two line charts. Again, you'll want to start with `line_dc` and `line_marvel` to create the new charts.

In [21]:
```
# modify the code below
i_line_dc = line_dc.add_selection(brush).transform_filter(brush)
i_line_marvel = line_marvel.add_selection(brush).transform_filter(brush)
```

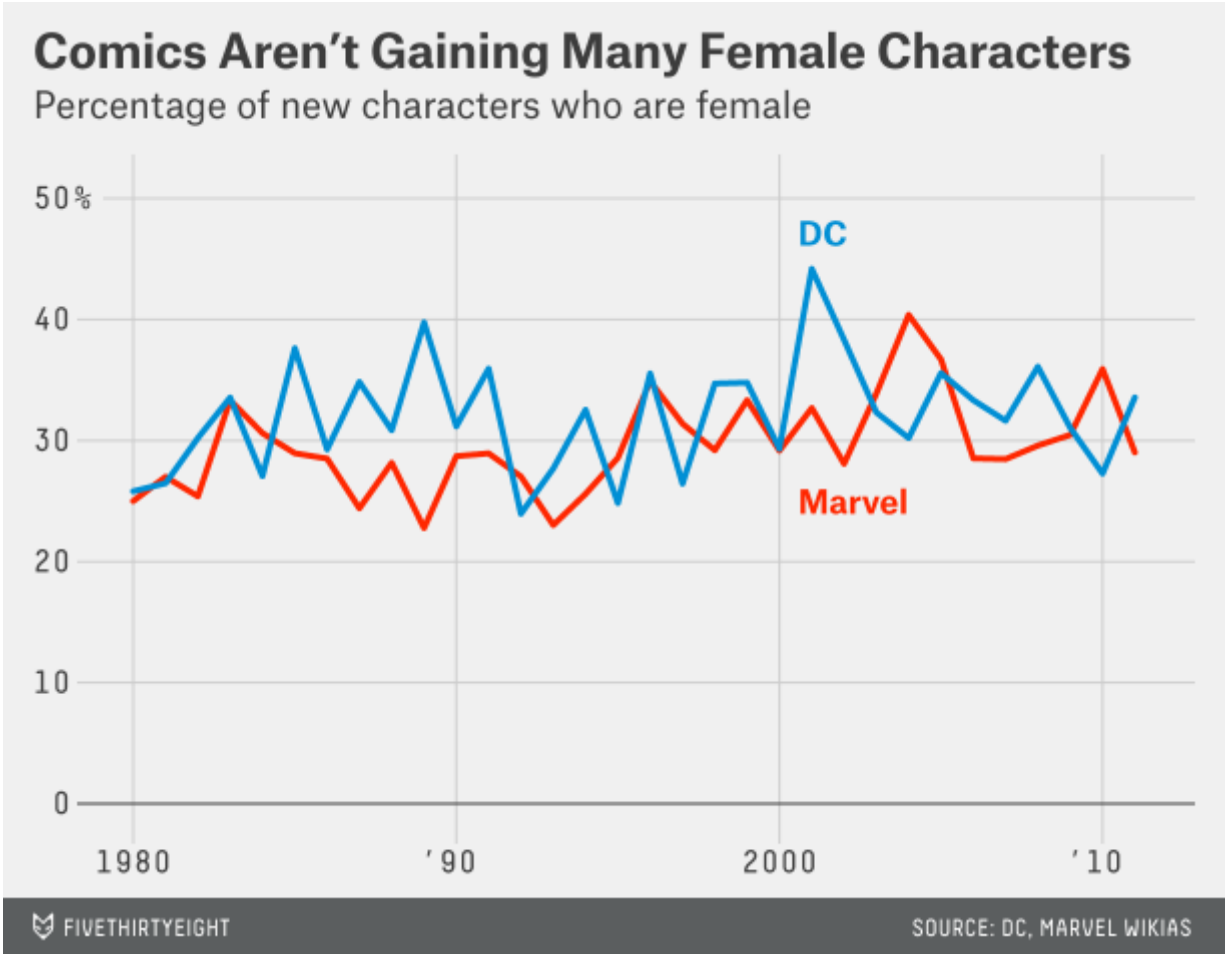
In [22]:
```
# let's put everything together with your new interactive charts. If you di
# should generate the visualizations we want

# bottom piece
bottom_charts = alt.hconcat(i_line_dc,i_line_marvel).resolve_scale(y='share

alt.vconcat(top_charts,bottom_charts).configure_view(
    strokeWidth=0
)
```

Out[22]:



Problem 2.2 (35 Points)

One of the issues discussed in the article is that the comics aren't gaining many female characters.



This visualization is ok, but we can enhance it with some interactivity. Let's start by dealing with the fact that the chart only presents one interesting: Percent female in any given year. It might help us understand the claim that there's a relatively trending change in this percent by plotting year-over-year percent changes. Also, it's possible that there are more characters being introduced in later years. So even one or two good years in the 2000's may make up for lots of bad years in the past (it turns out that this is not the case, but it is a question we might ask).

We're going to create the table with all the necessary statistics for you next:

In [23]:
```
def generatePercentTable(publisher):
    _df = comic[comic.publisher == publisher]
    _df = _df[['SEX','YEAR']]
    _df = pd.get_dummies(_df)
    _df.YEAR = _df.YEAR.astype('int')
    _df = _df.groupby(['YEAR']).sum()

    _df['total'] = 0
    _df['total'] = _df['total'].astype('int')
    for col in list(comic[comic.publisher == publisher].SEX.unique()):
        col = str(col)
        if (col != 'nan'):
            _df['total'] = _df['total'].astype('int') + _df["SEX_"+col].ast

    _df['% Female'] = _df['SEX_Female Characters'] / _df.total
    _df = _df.reset_index()
    _df = _df[['YEAR','% Female','SEX_Female Characters','SEX_Male Characte
    _df['publisher'] = publisher
    _df = _df[(_df.YEAR >= 1979)]
    _df['Year-over-year change in % Female'] = _df['% Female'].pct_change()
    toret = _df[(_df.YEAR > 1980) & (_df.YEAR < 2013)].copy()
    t2 = toret.cumsum()
    toret['% Female characters to date'] = list(t2['SEX_Female Characters']
    return(toret)

changedata = pd.concat([generatePercentTable("Marvel"),generatePercentTable

changedata = pd.melt(changedata,id_vars=['YEAR','publisher'],value_vars=['%
                                                                         'Year-over-yea
                                                                         '% Female char
```

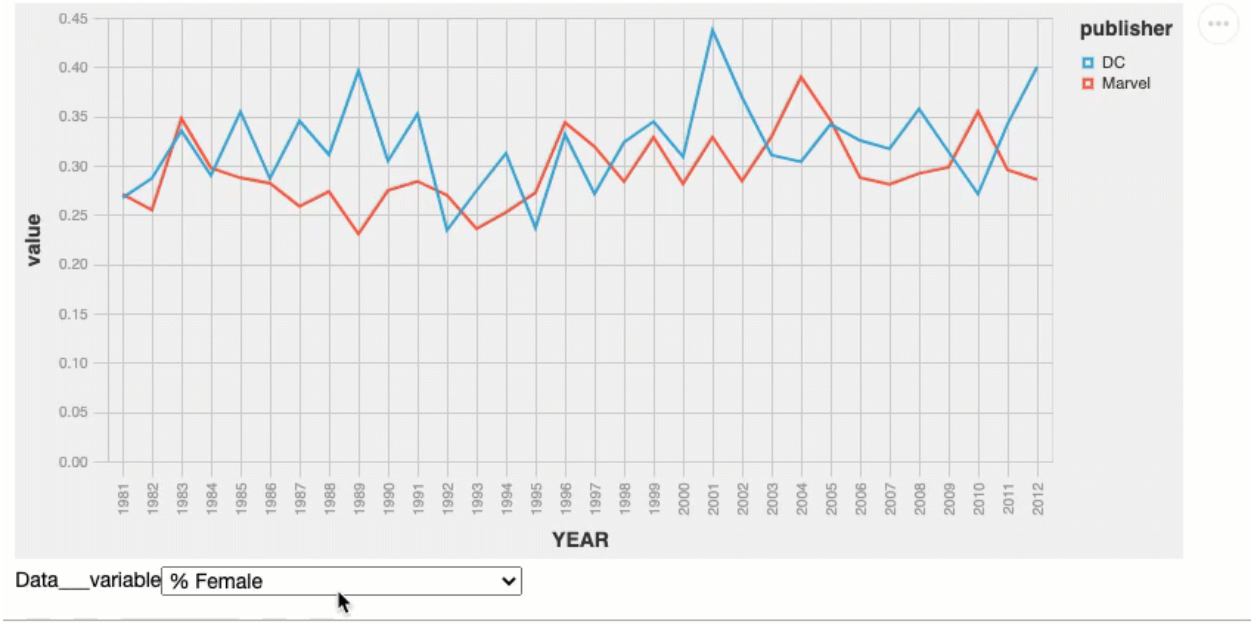
In [24]:
```
# let's see what's inside
changedata.sample(5)
```

Out[24]:

	YEAR	publisher	variable	value
92	2009	Marvel	Year-over-year change in % Female	0.021713
114	1999	DC	Year-over-year change in % Female	0.064532
40	1989	DC	% Female	0.396154
130	1983	Marvel	% Female characters to date	0.289256
172	1993	DC	% Female characters to date	0.315227

Problem 2.2.1

Your first job will be to create an interactive chart that has a drop-down box that allows us to select the variable of interest. Here's our target in action:

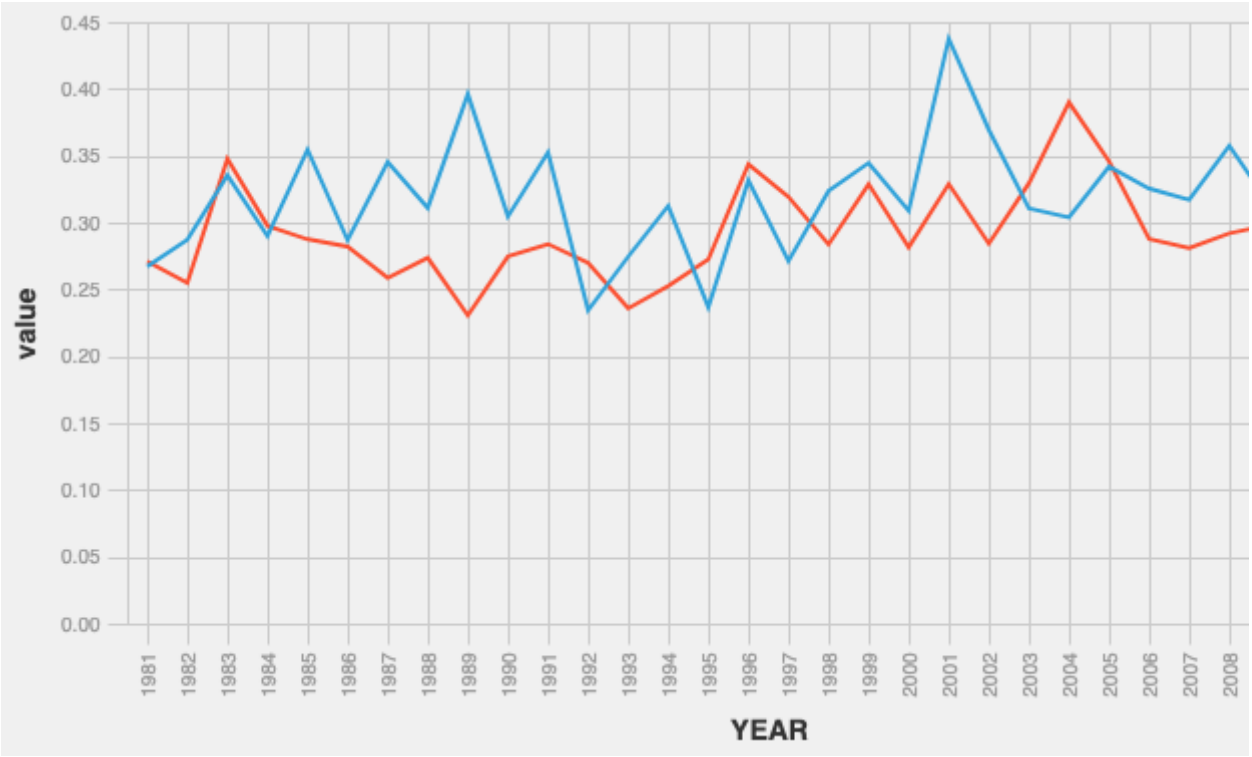


Modify `generateLineChartP21` below to generate this chart. If you haven't already, you'll want to take a look at the binding_select examples. Make sure you can get the chart working without interactivity first (hint: see if you can figure out how to filter to specific variables of interest).

In [25]:
```
def generateLineChartP21():

    metricOptions = ['% Female','Year-over-year change in % Female','% Fema

    selectMetric = alt.selection_single(
        fields=['variable'],
        init={'variable': '% Female' },
        bind=alt.binding_select(options=metricOptions,name='Data__variable
    )

    base = alt.Chart(changedata).encode(
        x='YEAR:O',
        y='value:Q',
        color='publisher:N'

    )
    line = base.mark_line().add_selection(selectMetric).transform_filter(se


    return line
```

In [26]:
```
# if you did everything correctly, this should generate the visualization:
```

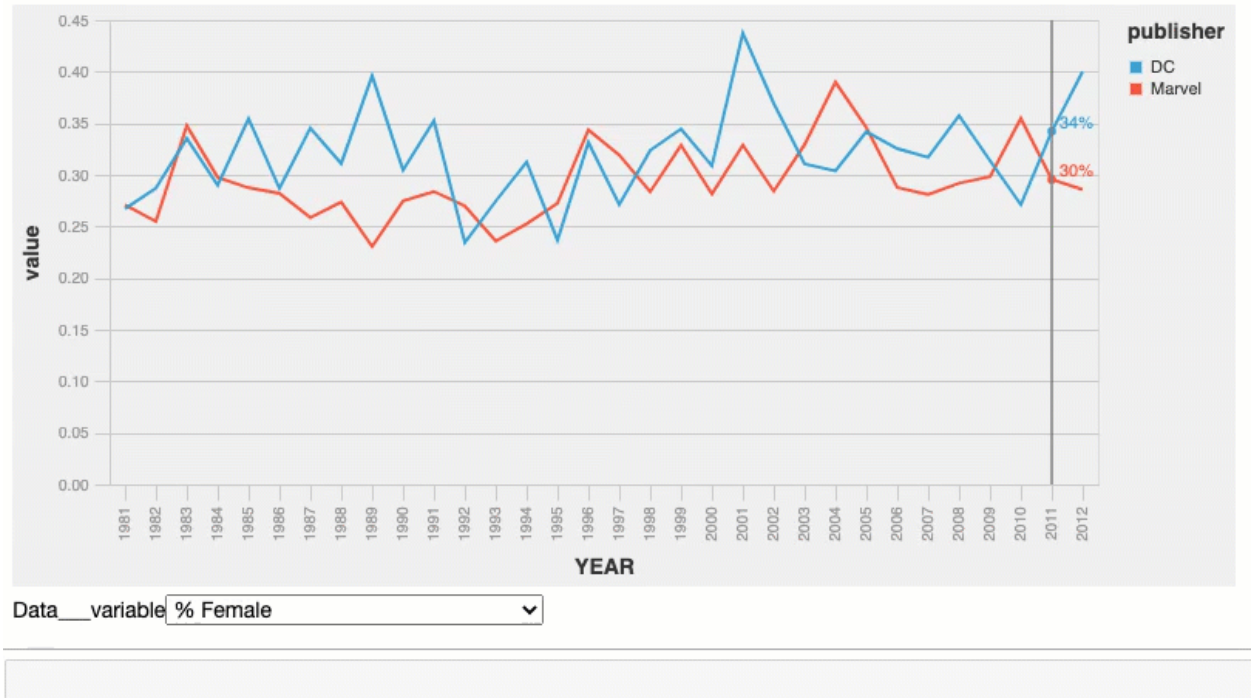
Out[26]:



Data__variable % Female

Problem 2.2.2

The next thing we're going to do is modify this example to give us a useful line that gives us the actual values (an effectiveness boost if we want to know the numbers). Here's an example:



Notice that the dropdown functionality still works. Your task is to build `generateLineChartP22` below to return this modified line chart. The good news is there's an example that's really close to what you need (https://altair-viz.github.io/gallery/multiline_tooltip.html). But you'll need to understand what's going on and modify it.

Some hints:

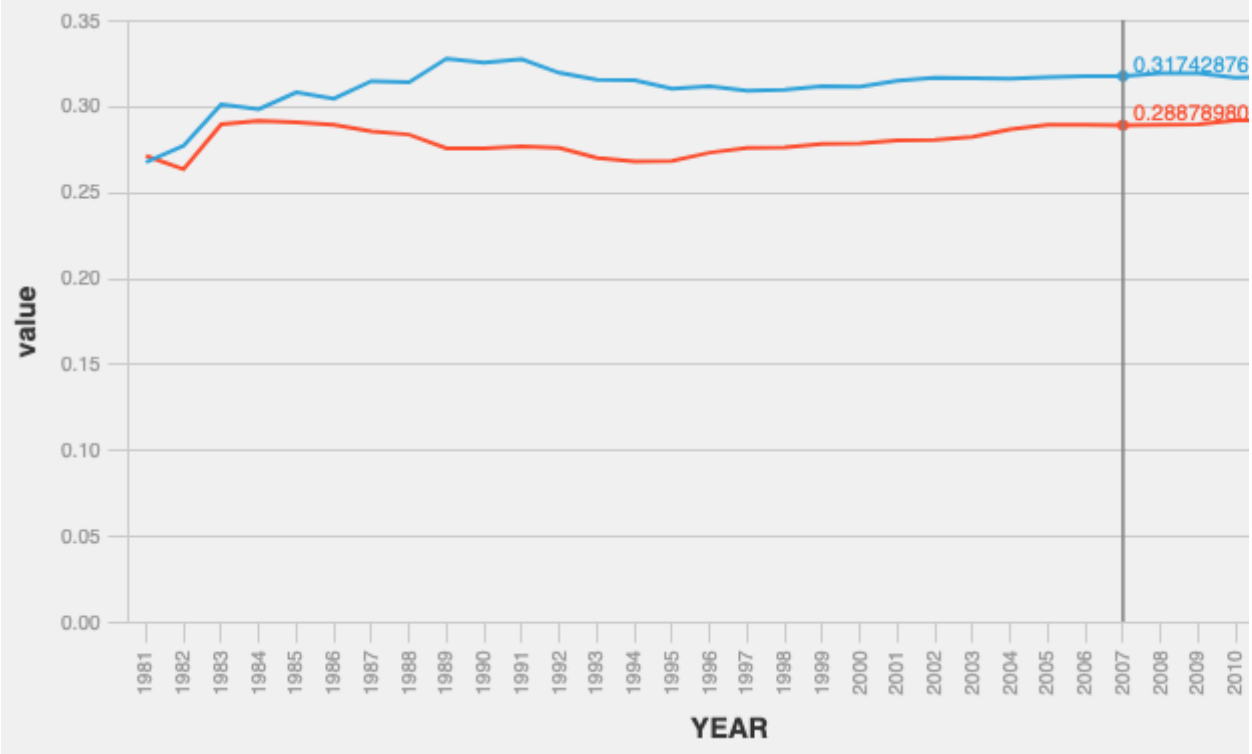
- You probably want to copy your code for `generateLineChartP21` into the new function. There are pieces of code you defined (e.g., the selection) that you'll need to use again.
- The example relies a lot on overloading Altair charts from a common base (e.g., `line = alt.Chart(...` and then `newline = line.encode...` so newline overloads/extends line). Our experience is that it's easy to get errors when doing this here because you'll be using multiple selections and conditions (another hint). We recommend defining the Altair charts (selectors, points, etc.) from scratch. It's more repeated code, but it'll save you the same headaches.

In [27]:
```
def generateLineChartP22():

    metricOptions = ['% Female','Year-over-year change in % Female','% Fema
    selectMetric = alt.selection_single(
            fields=['variable'],
            init={'variable': '% Female' },
            bind=alt.binding_select(options=metricOptions,name='Data__varia
        )

    # ---- interactive line chart ----
    line = alt.Chart(changedata).mark_line().encode(
        x=alt.X('YEAR:O'),
        y=alt.Y('value:Q'),
        color=alt.Color('publisher:N')
    )

    nearest = alt.selection(type='single', nearest=True, on='mouseover',
                            fields=['YEAR'], empty='none')


    selectors = alt.Chart(changedata).mark_point().encode(
        x=alt.X('YEAR:O'),
        opacity=alt.value(0),
    ).add_selection(
        nearest
    )

    points = alt.Chart(changedata).mark_point().encode(
        x=alt.X('YEAR:O'),
        y='value:Q',
        color='publisher:N',
        opacity=alt.condition(nearest, alt.value(1), alt.value(0))
    )

    # Draw text labels near the points, and highlight based on selection
    text = alt.Chart(changedata).mark_text(align='left', dx=5, dy=-5).encod
        x=alt.X('YEAR:O'),
        y=alt.Y('value:Q'),
        color=alt.Color('publisher:N'),
        text=alt.condition(nearest, 'value:Q', alt.value(' '))
    )

    # Draw a rule at the location of the selection
    rules = alt.Chart(changedata).mark_rule(color='gray').encode(
        x='YEAR:O',
    ).transform_filter(
        nearest
    )

    # Put the five layers into a chart and bind the data
    chart = alt.layer(
        line, selectors, points, rules, text
    ).properties(
        width=600, height=300
    )
```

```
    chart = chart.add_selection(selectMetric).transform_filter(selectMetric
    return chart
```

In [28]:
```
generateLineChartR22()
```

Out[28]:



Extra Credit (up to 10 points)

As an extra credit exercise, you can create a new interactive visualization that either replaces/extends one of the 538 examples OR invent a new one that fits with the article.

The interaction should be well thought out and appropriate (so just turning on ```.interactive()``` on a static chart won't really cut it). Please give us 1-2 sentences about what your interactivity adds.

In [ ]: