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. Available online (Hichey, 2014)
- 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 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.

Part 1. Interactive visualization assesment (30 points)

Read the following article Timeless Songs 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.

YOUR ANSWER HERE

- What's Remembered from the 90s
    - Select is implemented by 'What's Remembered from the 90s' using the mouse-hover technique to support detailed views of individual data points. Filter is also used through a search bar, in order to highlight statistics pertaining to a single artist of interest(all other artists are greyed out so as not to detract from the viewer's attention).
- Biggie or Tupac
    - Select is implemented by 'Biggie or Tupac' using the mouse-hover technique to support detailed views of individual data points. Filter is implemented using a search bar and buttons, to show data conditionally(i.e. only show points for rapper x and exclude all other points). Encode is also used(red color is utilized to highlight rappers that have been specifically filtered for, otherwise, all points are red).
- Present-day Popularity of Five Decades of Music
    - Filter is implemented in 'Present-day Popularity of Five Decades of Music' through the search bar, in order to only display bars for the artist of interest, while the buttons can be used as a broader filter by decade. Encode is used through the coloration of the bars(different color for each decade), to support easier cross-decade analysis.
- XXXX Tracks: Historic Billboard Performance vs. 2014 Spotify Plays
    - Filter is implemented in 'XXXX Tracks: Historic Billboard Performance vs. 2014 Spotify Plays' through the search bar, to highlight statistics pertaining to a single artist of interest(all other artists are greyed out), while the buttons and year widget can be used to filter more broadly by decade or by year. Select is implemented using the mouse-hover technique, again, to support detailed views of individual data points, while encode is used by the won grammy/peaked at billboard #1 buttons to highlight specific points the author thought was of greater importance(encoded using the color red).
- The Long-term Future of Hits from 2013
    - Filter is implemented in 'The Long-term Future of Hits from 2013' through the search bar, to allow users to search for specific tracks without scanning the list of ones available for

visualization(only supports single selection). Selection is implemented through the list of tracks by check boxes, allowing users to select as many tracks as they would like to visualize. encode is utilized through the colors of the line graphs and the labels(if the viewer selects multiple tracks, each line will be encoded using a different color).

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.

I believe XXXX 'TRACKS: HISTORIC BILLBOARD PERFORMANCE VS. 2014 SPOTIFY PLAYS' is arguably the chart with the highest learning curve, in the sense that it's difficult to keep track of all the various encodings simultaneously. Intuitively, one might expect that the size of each circle point would indicate the size of the playcount for each respective track, but that is actually not the case. Rather, the size is indicative of billboard performance at the time of release, whereas the tickmarks are what are actually indicative of song playcount. Furthermore, the color encodings for both 'Won Grammy' and 'Peaked At Billboard #1 are the same(both bright red), and it is not possible to toggle both simultaneously. I believe emulating 'Present-day Popularity of Five Decades of Music' would have benefitted this graph. The filter search bar and year widget, the buttons, and other existing toggling could exist, and similarly, the bars corresponding to each decade could be encoded with a different color. The bars indicating a grammy win and #1 billboard toppers could have each been overlaid with some unique indicator(a checkmark symobl, a trophy symbol, etc). The bar chart format would have made magnitude comparisons much more intuitive, which was the intended goal of the original visualization anyhow, so effectiveness would not be sacrificed by converting formats, and would arguably be augmented.

Part 2. Programming exercise (70 points)

Start by reading the 538 article here. 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]:
```python
# start with the setup
import pandas as pd
import numpy as np
import altair as alt
```

In [2]:
```python
# 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['publisher'] = 'DC'
marvel['publisher'] = 'Marvel'
```

In [6]:
```
# rename some columns
marvel.rename(columns={'Year': 'YEAR'}, inplace=True)
```

In [7]:
```
# create the table with everything
comic = pd.concat([dc, marvel])

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

In [8]:
```
# let's look inside
comic.sample(5)
```

Out[8]:

	page_id	name	urlslug	ID	ALIGN	EYE	HA
247	1140	Wolfgang von Strucker (Earth-616)	\/Wolfgang_von_Strucker_(Earth-616)	Public Identity	Bad Characters	Blue Eyes	Ba
81	21110	Daniel Turpin (New Earth)	\/wiki\/Daniel_Turpin_(New_Earth)	Public Identity	NaN	Brown Eyes	Brov H
4972	37369	Social Butterfly (Earth-616)	\/Social_Butterfly_(Earth-616)	Secret Identity	Good Characters	NaN	Na
4459	191490	Red of the Rainbow Raiders (New Earth)	\/wiki\/Red_of_the_Rainbow_Raiders_(New_Earth)	Secret Identity	Bad Characters	NaN	R H

	page_id	name	urlslug	ID	ALIGN	EYE	HA
11925	306526	Jake Malloy (Earth-616)	\/Jake_Malloy_(Earth-616)	NaN	Bad Characters	NaN	Na

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

Original article available at FiveThirtyEight

By Walt Hickey

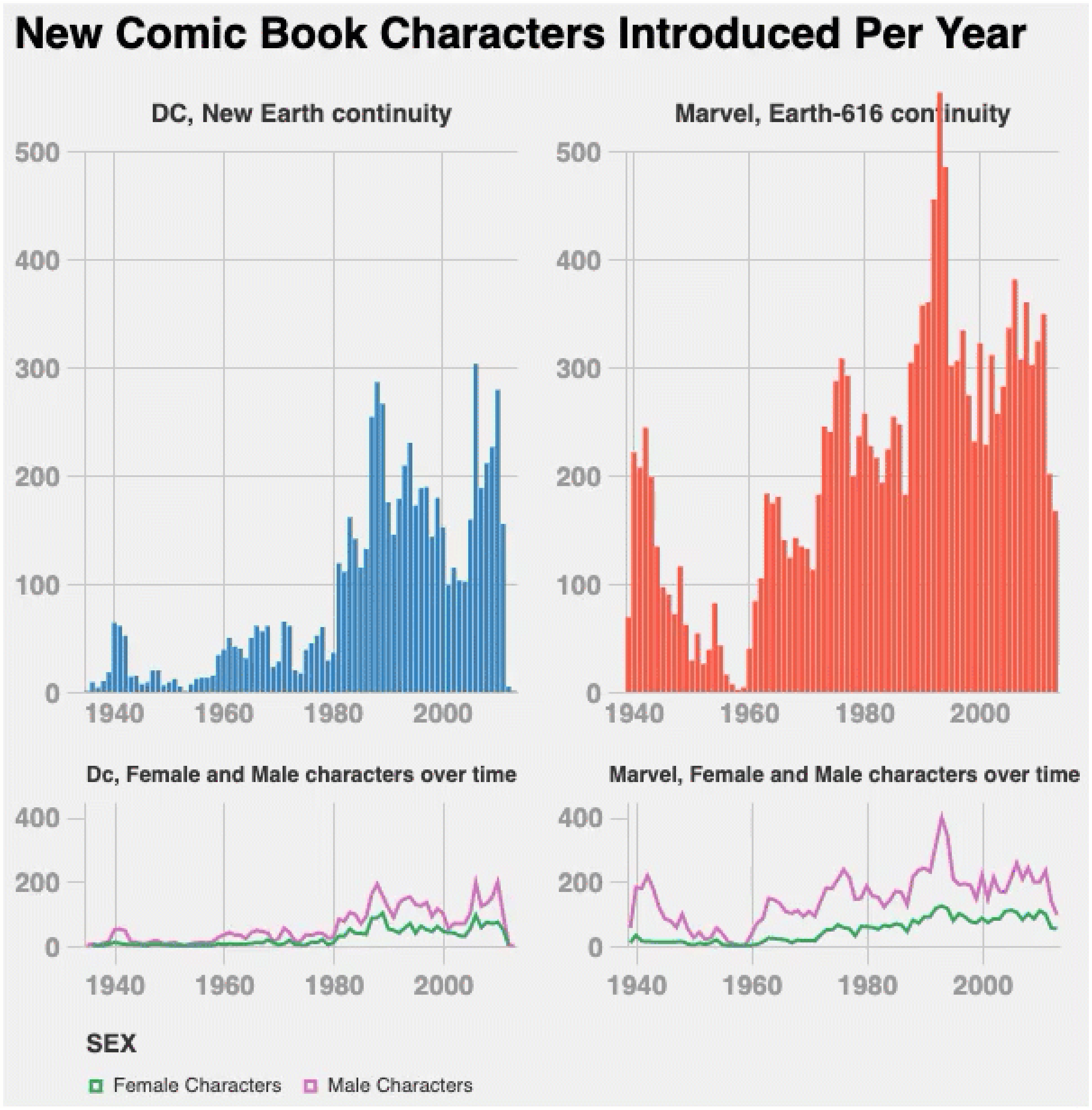
Get the data on GitHub

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).

In [9]:
```python
# let's pre-process the data. We're going to focus on just Female and just Male charact
# only consider those
comic_ch1_df = comic[(comic['YEAR'] >= 1940) & (comic['SEX'].isin(['Female Characters',
```

In [10]:
```python
# check what's inside
comic_ch1_df.sample(5)
```

Out[10]:

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	
1962	4916	Sebastian Blood IX (New Earth)	\/wiki\/Sebastian_Blood_IX_(New_Earth)	Secret Identity	Bad Characters	Red Eyes	White Hair	Charac

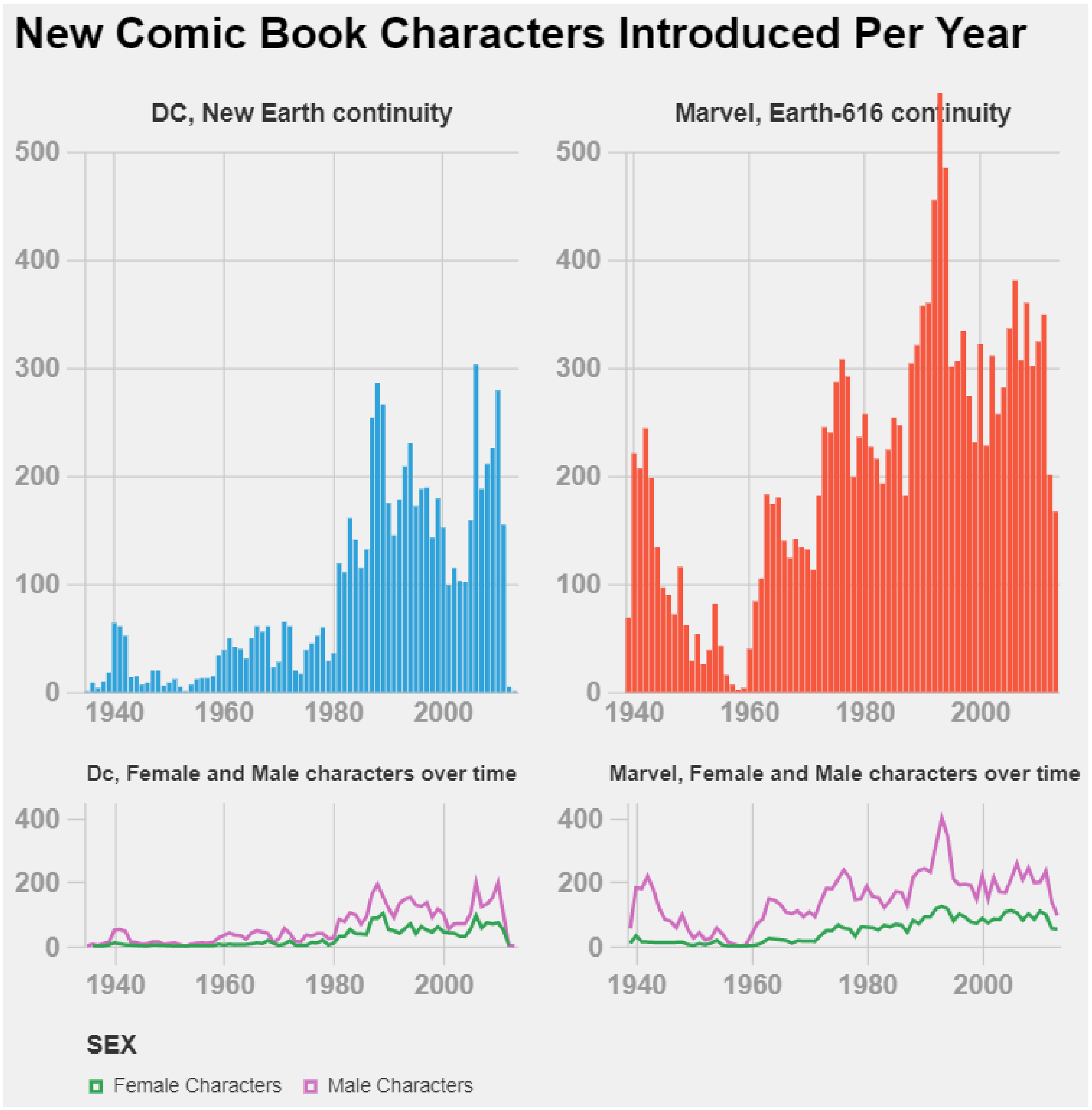
	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	
13606	560516	Midwife (Earth-616)	\/Midwife_(Earth-616)	Secret Identity	Bad Characters	NaN	Red Hair	Fe
9239	29097	Jerome Hamilton (Battleaxe) (Earth-616)	\/Jerome_Hamilton_(Battleaxe)_(Earth-616)	Secret Identity	NaN	NaN	Black Hair	Chara
9178	254295	Kartak (Earth-616)	\/Kartak_(Earth-616)	NaN	Bad Characters	NaN	NaN	Chara
6387	201677	Galte-Re (New Earth)	\/wiki\/Galte-Re_(New_Earth)	Public Identity	Good Characters	NaN	NaN	Chara

In [11]:

```python
# we're largely going to use the same "base" visualization here for the bar. We can sta
# 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
                              axis=alt.Axis(values=[1940, 1960, 1980, 2000], labels=Tr
                                            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
                            # fix the look of the axes
                           axis=alt.Axis(values=[1940, 1960, 1980, 2000], labels=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 bar chart bas
# 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=
               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 M
                                                  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 M
                                                      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[11]:



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 to identify the right (and the lab).

In [12]:

```
# 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 [13]:

```
# modify this cell to create the brush object
colorConditionDC = alt.condition(brush, alt.value('#008fd5'), alt.value('gray'), legend
```

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.

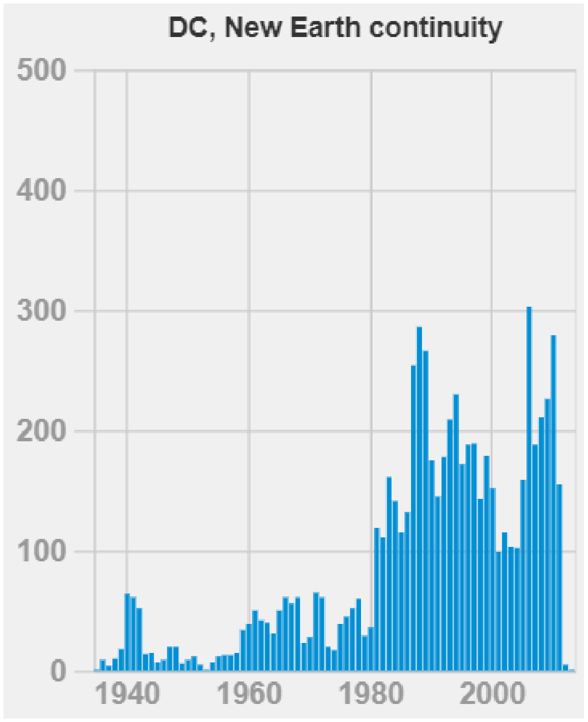
In [14]:

```
# modify this cell to create the brush object
i_bar_dc = bar_dc.add_selection(
    #step 3
    brush
).encode(
    #step 4,
    color=colorConditionDC
)
```

In [15]:

```
# if you did the last step correctly, you should be able to see the selection work for
i_bar_dc
```

Out[15]:



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`

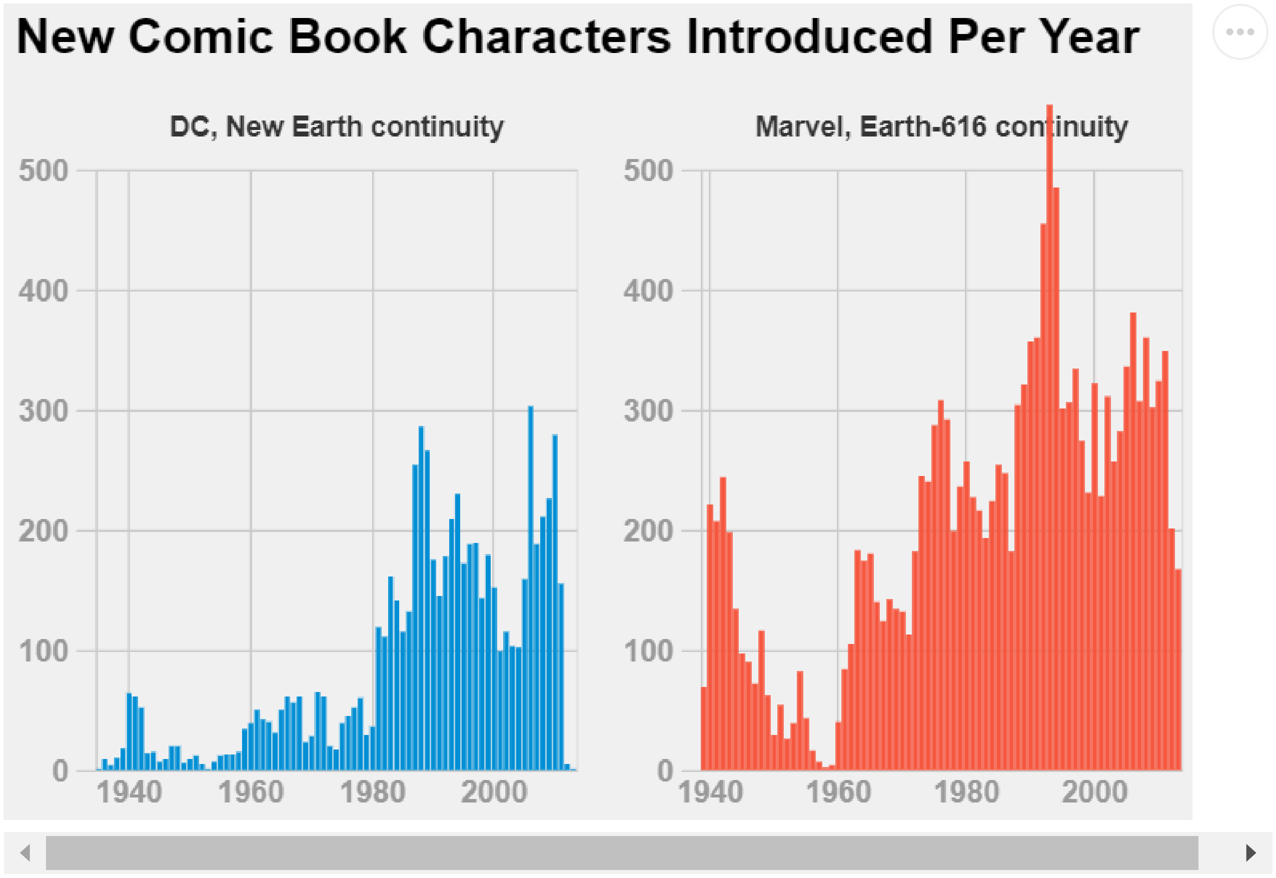
In [16]:

```
# modify the following two lines
colorConditionMarvel = alt.condition(brush,alt.value('#f6573f'),alt.value('gray'),legen
i_bar_marvel = bar_marvel.add_selection(
    #step 3
    brush
).encode(
    #step 4,
    color=colorConditionMarvel,
)
```

In [17]:

```
# 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 interactively sele
# (and the selections should be linked)
top_charts
```

Out[17]:



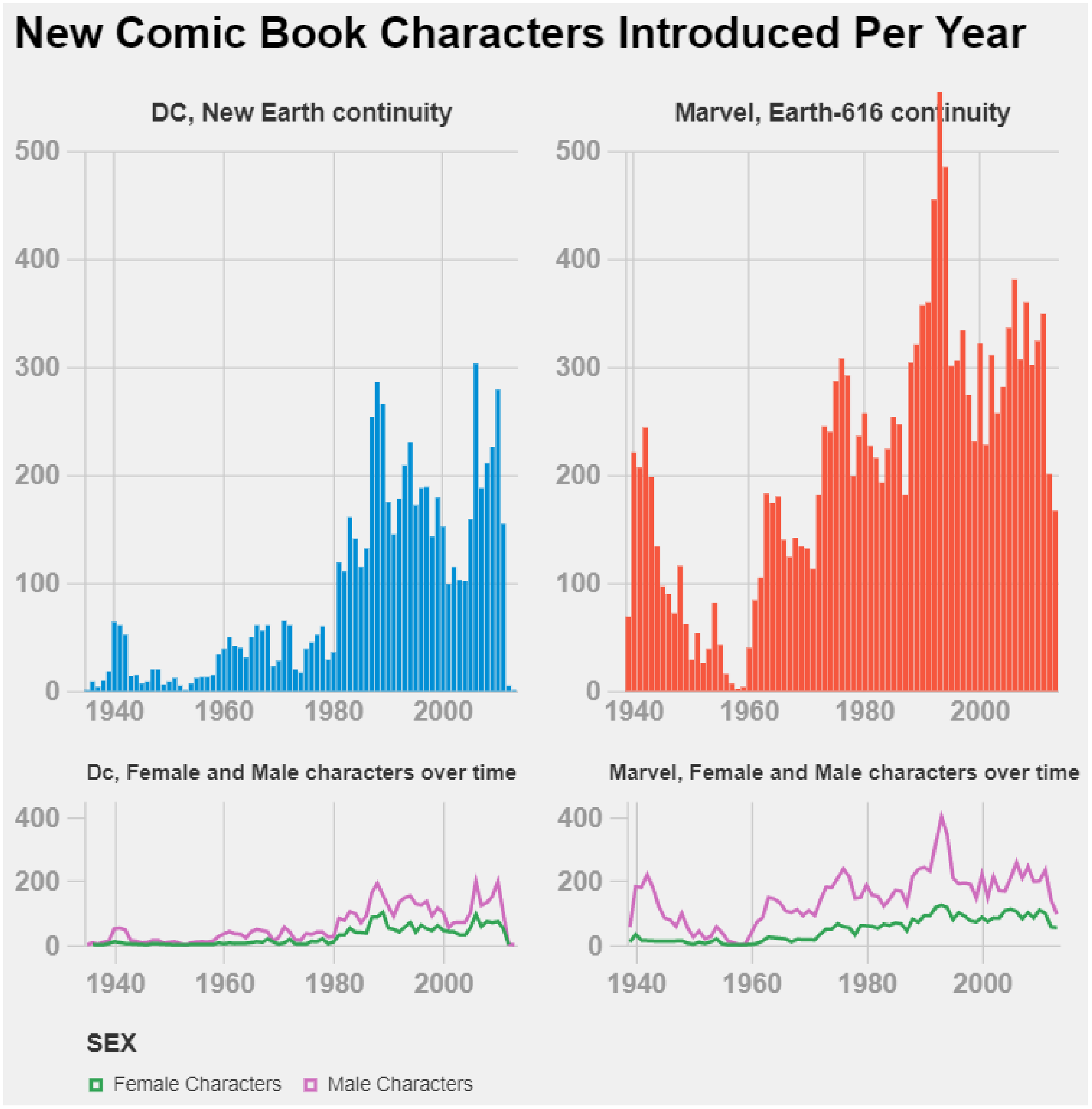
Problem 2.1.5

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 [18]:
```
# modify the code below
i_line_dc = line_dc.add_selection(
    #step 3
    brush
).transform_filter(brush)


i_line_marvel = line_marvel.add_selection(
    #step 3
    brush
).transform_filter(brush)
```

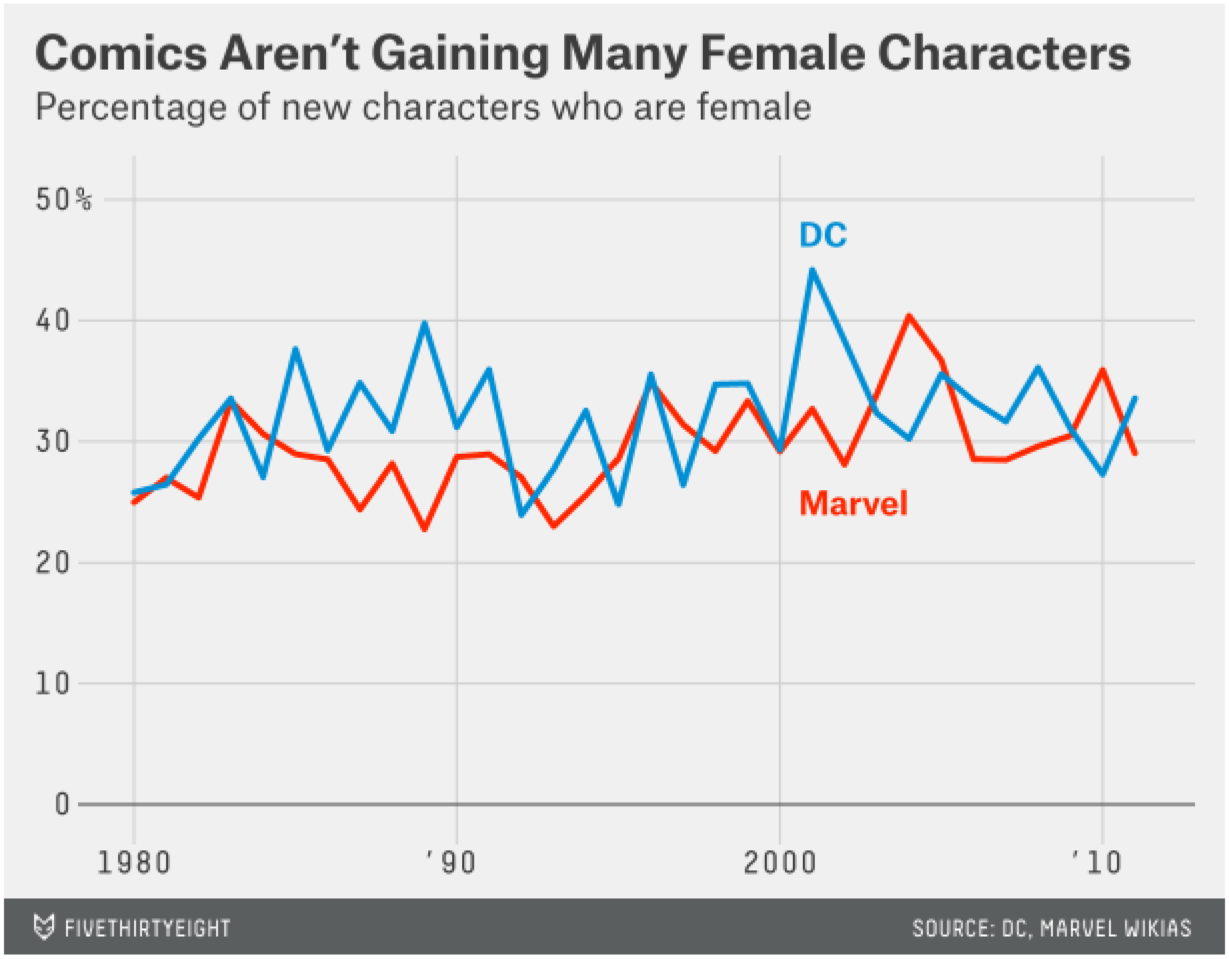
In [19]:
```
# let's put everything together with your new interactive charts. If you did everything
# should generate the visualizations we want

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

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

Out[19]:



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 [24]:
```
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].astype('int')

    _df['% Female'] = _df['SEX_Female Characters'] / _df.total
    _df = _df.reset_index()
    _df = _df[['YEAR','% Female','SEX_Female Characters','SEX_Male Characters','total']]
    _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'] / t2['total
    return(toret)

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

changedata = pd.melt(changedata,id_vars=['YEAR','publisher'],value_vars=['% Female',
                                                                         'Year-over-year change in
                                                                         '% Female characters to da
```

In [25]:

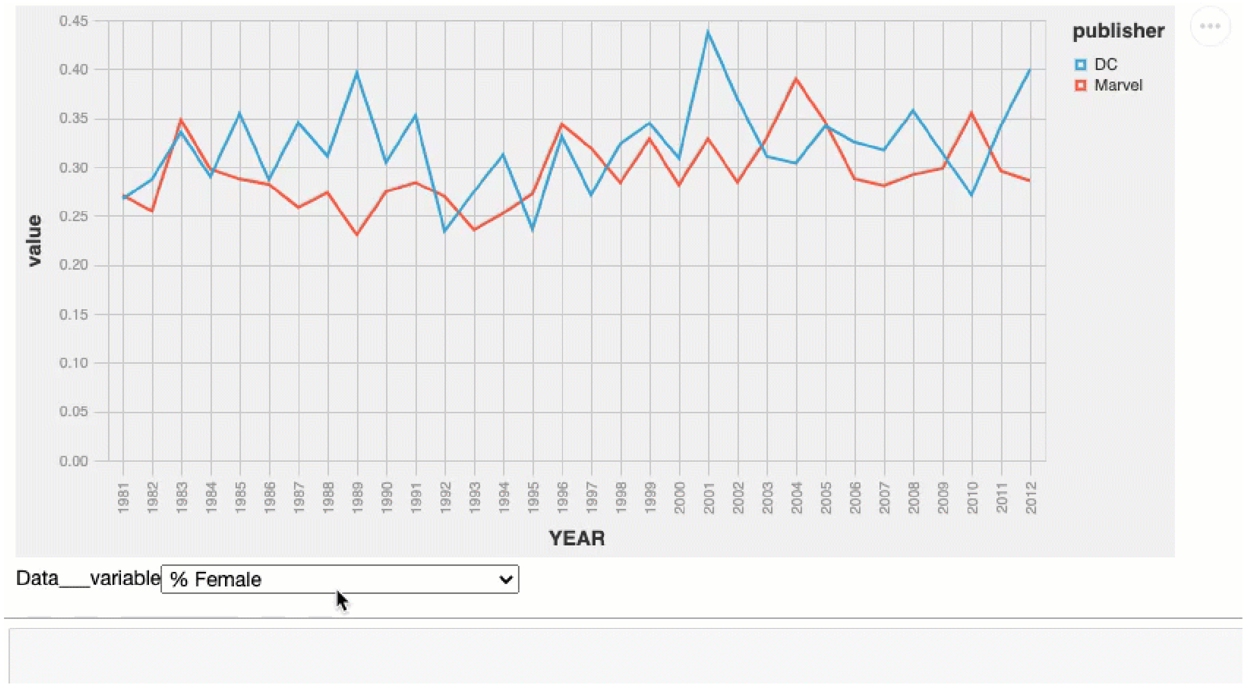
```
# let's see what's inside
changedata.sample(5)
```

Out[25]:

	YEAR	publisher	variable	value
190	2011	DC	% Female characters to date	0.317159
172	1993	DC	% Female characters to date	0.315227
95	2012	Marvel	Year-over-year change in % Female	-0.033189
146	1999	Marvel	% Female characters to date	0.278010
50	1999	DC	% Female	0.344633

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).

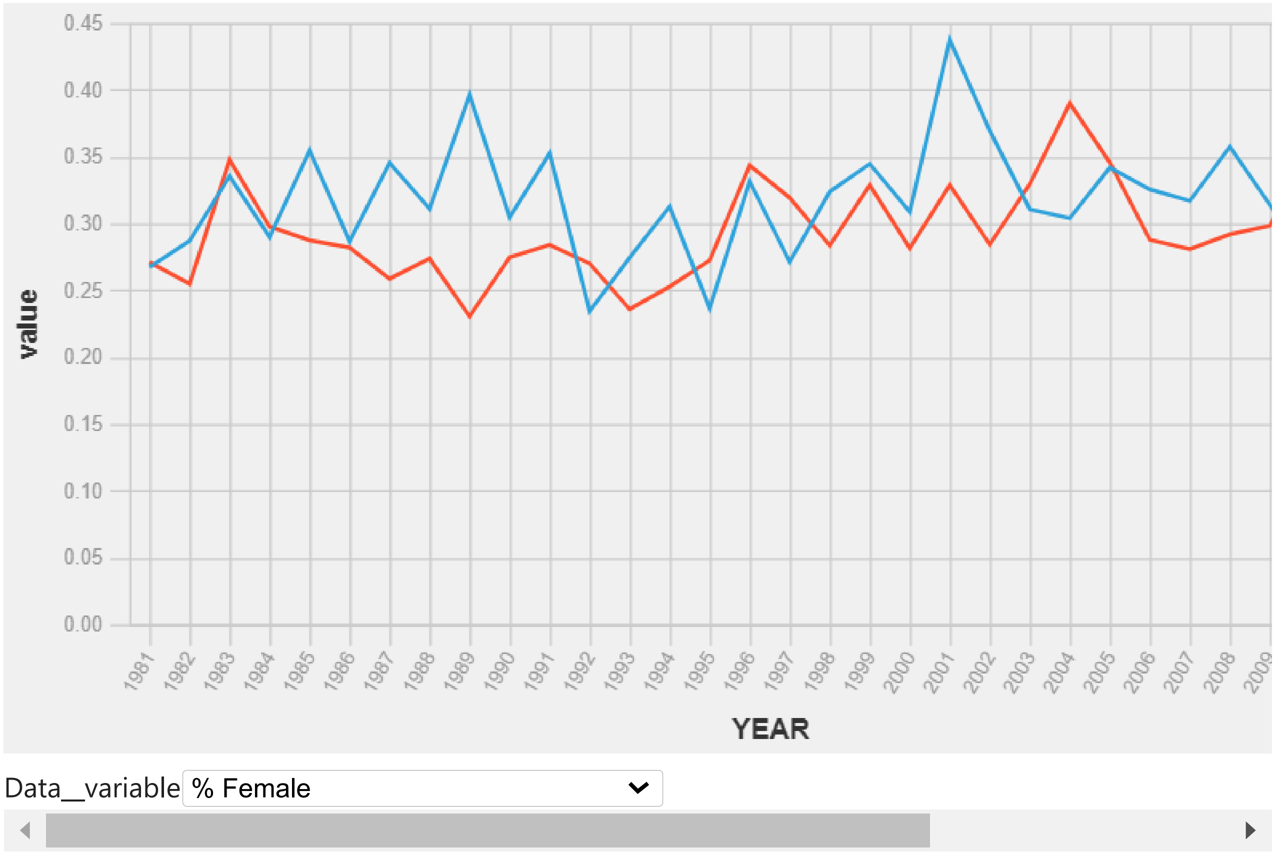
In [107…

```
def generateLineChartP21():

    metricOptions = ['% Female','Year-over-year change in % Female','% Female character
    input_dropdown = alt.binding_select(options=metricOptions)



    dropdown_selection = alt.selection_single(fields=['variable'], bind=input_dropdown,

    line = alt.Chart(changedata).mark_line().encode(
    x=alt.X('YEAR:N',axis=alt.Axis(values=np.arange(1981, 2013),tickCount=len(changedat
    y=alt.Y('value:Q'),
        color = 'publisher'
    ).add_selection(
    dropdown_selection
).transform_filter(
    dropdown_selection
)




    return(line)
```

In [108…

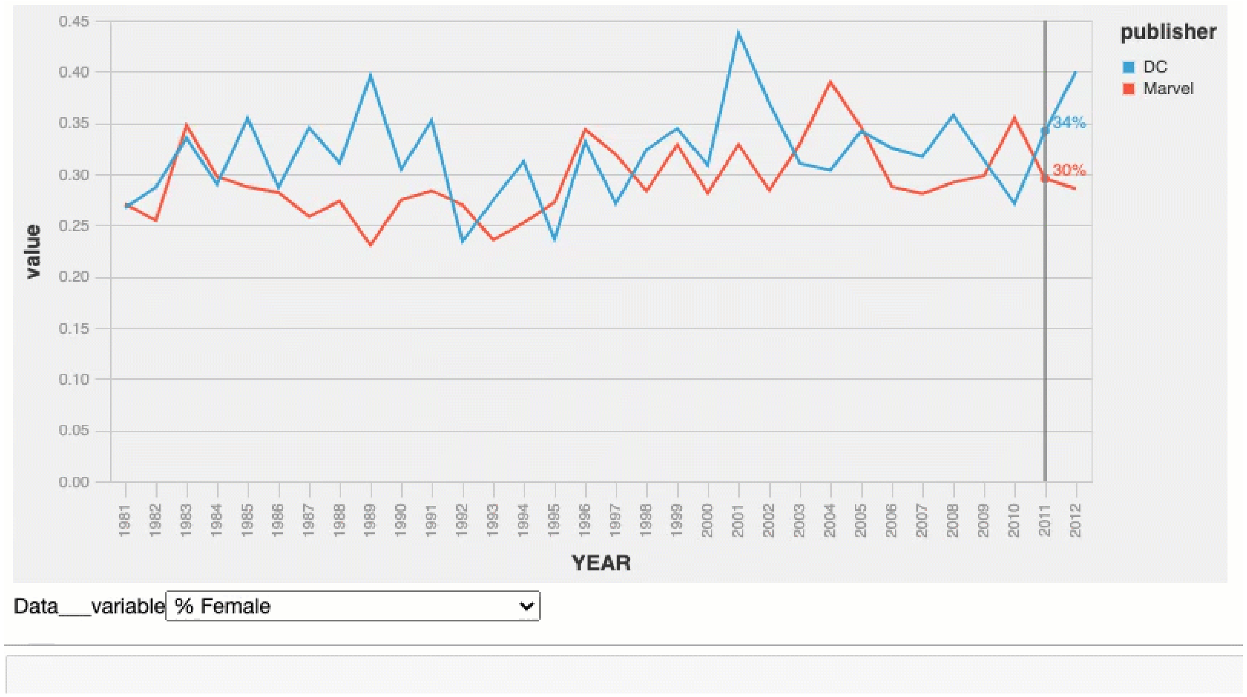
```
# if you did everything correctly, this should generate the visualization:
generateLineChartP21()
```

Out[108…



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. 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.

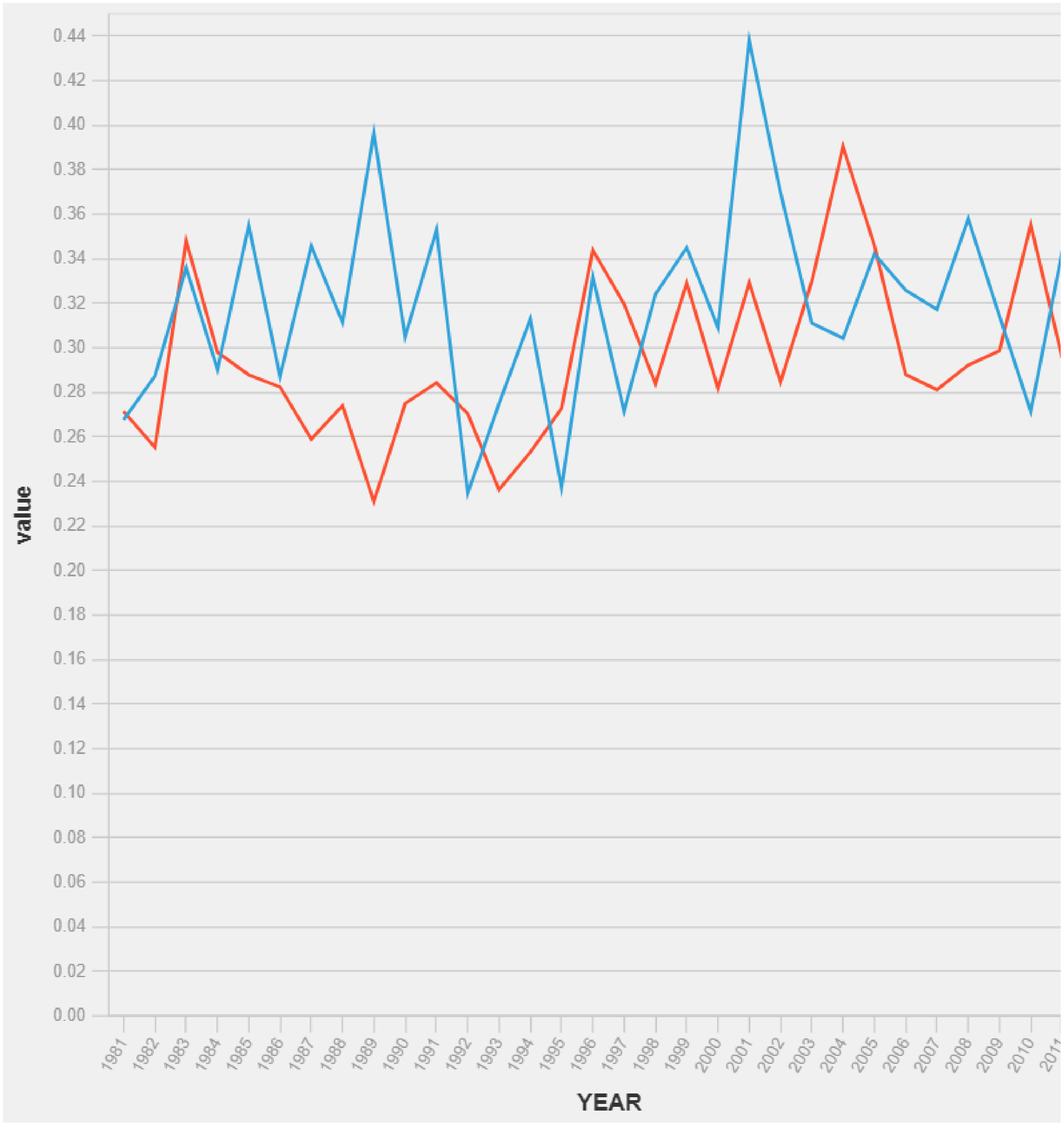
In [103…

```
def generateLineChartP22():

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

    input_dropdown = alt.binding_select(options=metricOptions)

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

    dropdown_selection = alt.selection_single(fields=['variable'], bind=input_dropdown,

    line = alt.Chart(changedata).mark_line().encode(
    x=alt.X('YEAR:N',axis=alt.Axis(values=np.arange(1981, 2013),tickCount=len(changedat
    y=alt.Y('value:Q'),
        color = 'publisher'
    )



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

    points = line.mark_point().encode(
    opacity=alt.condition(nearest, alt.value(1), alt.value(0))
)

    text = line.mark_text(align='left', dx=5, dy=-5).encode(
    text=alt.condition(nearest, 'value:Q', alt.value(' '))
)


    rules = alt.Chart(changedata).mark_rule(color='gray').encode(
    x='YEAR:N',
).transform_filter(
    nearest
)



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



    return final_chart
```

In [104…

```
generateLineChartP22()
```

Out[104…



Data__variable % Female

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 [ ]:
```
# YOUR ANSWER HERE
```