# 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,explore,...} is implemented by ... to support ...
  - {select,explore,...} is implemented by ... to support ...
- 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

### 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.

### Answer

- What's Remembered from the 90s
    - {select} is implemented by **clicking** on singer's profile image to support listening his/her representative music.
    - {select} is implemented by **mouse over** to support reading singer's profile, such as the full name, the year of start the career, and the quantity of music playing times.
    - {explore} is implemented by a kind of **line chart** to support reading the rank of number of plays on Spotify.
    - {filter} is implemented by **typing** the specific track name to check the status of number of plays on Spotify.
- Biggie or Tupac
    - {select} is implemented by **mouse over** to support reading singer's profile, such as the full name, the year of start the career, and the quantity of music playing times.
    - {explore} is implemented by a **plot chart** to support reading the distribution of number of Spotify playcounts by singers.
    - {filter} is implemented by **typing** the specific track name to check the status of Spotify playcounts.
    - {filter} is implemented by **clicking buttons** to support check the status of number of Spotify playcounts by either "Just biggie and tupac" or "Just Jay-Z". You can also check the number of all rappers.
- Present-day Popularity of Five Decades of Music
    - {explore} is implemented by a **bar chart** to support reading the rank of Spotify playcounts by track name.
    - {filter} is implemented by **typing** the specific artist name to check the status of Spotify playcounts.
    - {filter} is implemented by **clicking buttons** to support check the status of number of Spotify playcounts by the era of birth of artists.
- XXXX Tracks: Historic Billboard Performance vs. 2014 Spotify Plays
    - {select} is implemented by **mouse over** to support reading the track's profile, including the name of track, the artist, the name and the year of rank and the number of rank.
    - {explore} is implemented by a **plot chart** to support reading the status of Billboard ranking performance versus 2014 Spotify plays by years.
    - {filter} is implemented by **typing** the specific track name to check the status of comparison of that track.
    - {filter} is implemented by **clicking buttons** to support check the status of comparison by either the era of tracks or the status of awards.
    - {filter} is implemented by **typiing** the specific year to check the status of comparison on that year.
- The Long-term Future of Hits from 2013
    - {explore} is implemented by a **time-series line chart** to support reading the status of the long-term comparison by tracks.
    - {filter} is implemented by **typing** the specific track name to add it to the comparison.
    - {filter} is implemented by **selecting** tracks form the list to add them to the comparison.

---

## 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]:
```
# 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['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 | HAIR | SEX | GSM | ALIVE | APPEARANCES | APPE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **14601** | 122158 | Matthew Murdock (Skrull) (Earth-616) | \/Matthew_Murdock_(Skrull)_(Earth-616) | Secret Identity | Bad Characters | Green Eyes | No Hair | Male Characters | NaN | Deceased Characters | 1.0 | |
| **12099** | 723736 | Dane Gavin (Earth-616) | \/Dane_Gavin_(Earth-616) | Public Identity | Neutral Characters | Blue Eyes | Black Hair | Male Characters | NaN | Living Characters | 1.0 | |
| **8406** | 253096 | Doctor Manyac (Earth-616) | \/Doctor_Manyac_(Earth-616) | NaN | Bad Characters | NaN | Bald | Male Characters | NaN | Deceased Characters | 2.0 | |
| **983** | 68548 | Ravan (New Earth) | \/wiki\/Ravan_(New_Earth) | Secret Identity | Bad Characters | Brown Eyes | Black Hair | Male Characters | NaN | Deceased Characters | 28.0 | 1 |
| **2775** | 120115 | Bolshoi (New Earth) | \/wiki\/Bolshoi_(New_Earth) | NaN | Bad Characters | NaN | NaN | Male Characters | NaN | Living Characters | 8.0 | 198 |

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

*Original article available at FiveThirtyEight*

By Walt Hickey

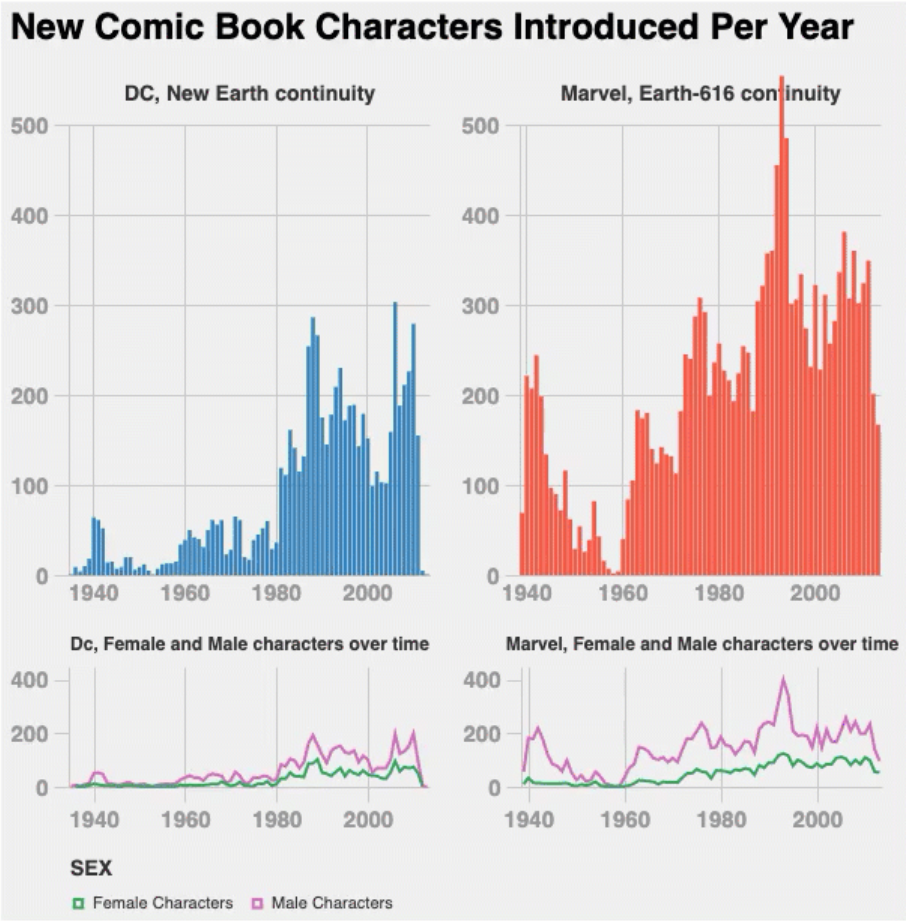
Get the data on GitHub

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

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

Out[10]:

| | page_id | name | urlslug | ID | ALIGN | EYE | HAIR | SEX | GSM | ALIVE | APPEARANCES | APPE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **10785** | 481346 | Bull Murdock (Earth-616) | \/Bull_Murdock_(Earth-616) | Public Identity | Bad Characters | NaN | Bald | Male Characters | NaN | Living Characters | 1.0 | |
| **4978** | 43057 | Zaneth (New Earth) | \/wiki\/Zaneth_(New_Earth) | NaN | Good Characters | NaN | NaN | Male Characters | NaN | Deceased Characters | 2.0 | S |
| **13** | 2614 | Ororo Munroe (Earth-616) | \/Ororo_Munroe_(Earth-616) | Public Identity | Good Characters | Blue Eyes | White Hair | Female Characters | NaN | Living Characters | 1512.0 | |

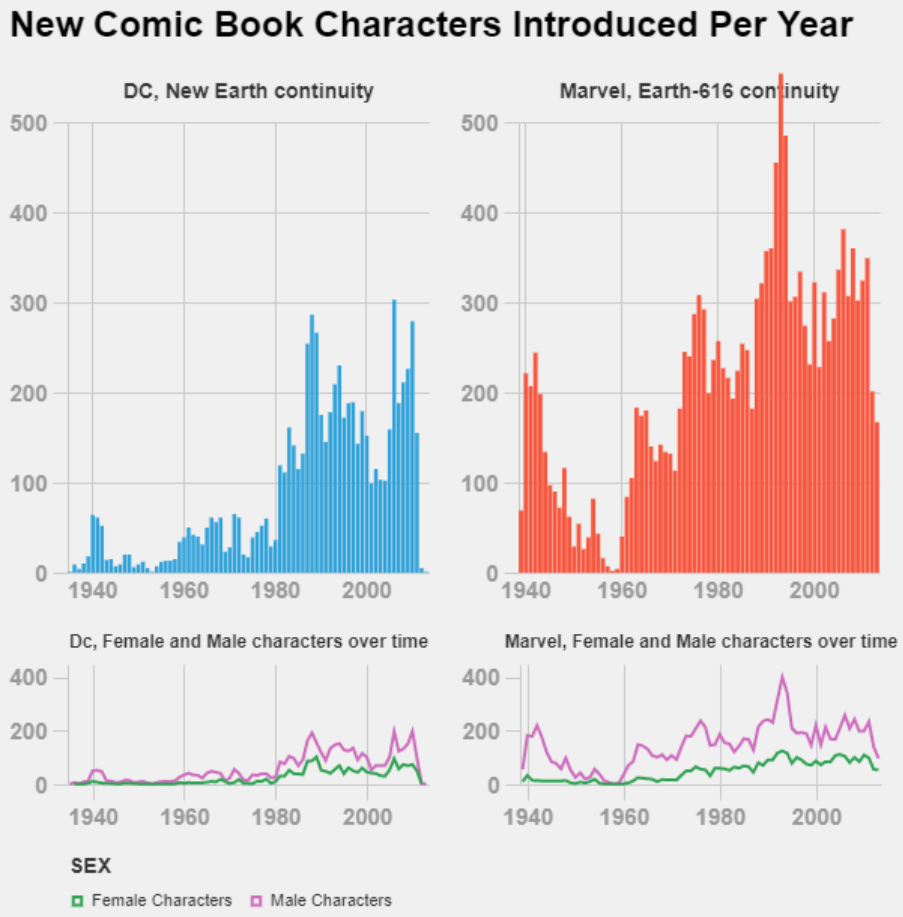
| | page_id | name | urlslug | ID | ALIGN | EYE | HAIR | SEX | GSM | ALIVE | APPEARANCES | APPE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **4** | 1576 | Richard Grayson (New Earth) | \/wiki\/Richard_Grayson_(New_Earth) | Secret Identity | Good Characters | Blue Eyes | Black Hair | Male Characters | NaN | Living Characters | 1237.0 | 1 |
| **4585** | 62366 | Wilde Norton (New Earth) | \/wiki\/Wilde_Norton_(New_Earth) | Secret Identity | Bad Characters | Blue Eyes | Brown Hair | Male Characters | NaN | Living Characters | 3.0 | 1 |

In [11]:

```
# we're largely going to use the same "base" visualization here for the bar. We can start with a 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 the axes
                                  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 bar chart base above
# 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[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(encodings=['x'])
# YOUR CODE HERE
# raise NotImplementedError()
```

## 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 = (brush, alt.value('#2182bd'), alt.value('gray'))


# YOUR CODE HERE
# raise NotImplementedError()
```

## 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:

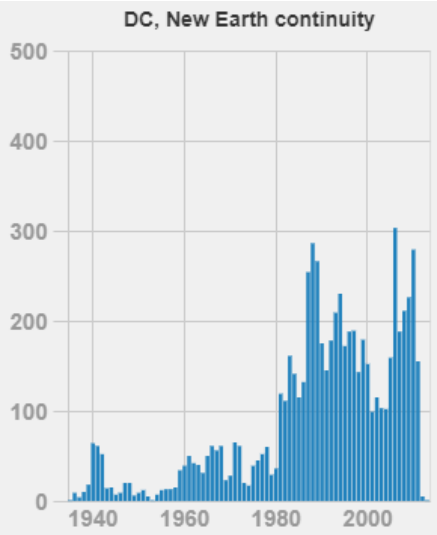
```
i_bar_dc = bar_dc.encode(color = 'TEST')
```

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.encode(color=alt.condition(brush, alt.value('#2182bd'), alt.value('gray'))).add_selection(brush)

# YOUR CODE HERE
# raise NotImplementedError()
```

In [15]:
```
# if you did the last step correctly, you should be able to see the selection work for the DC bar chart
i_bar_dc
```

Out[15]:



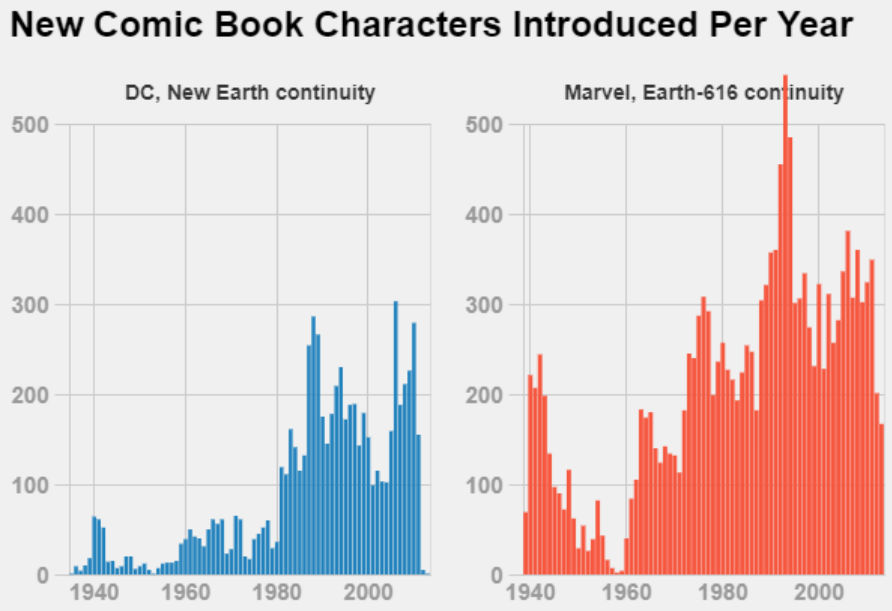

## Problem 2.1.4

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 = (brush, alt.value('#2182bd'), alt.value('gray'))
i_bar_marvel = bar_marvel.encode(color=alt.condition(brush, alt.value('#f6573f'), alt.value('gray'))).add_selection(brush)

# YOUR CODE HERE
# raise NotImplementedError()
```

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 select
# (and the selections should be linked)
top_charts
```

Out[17]:



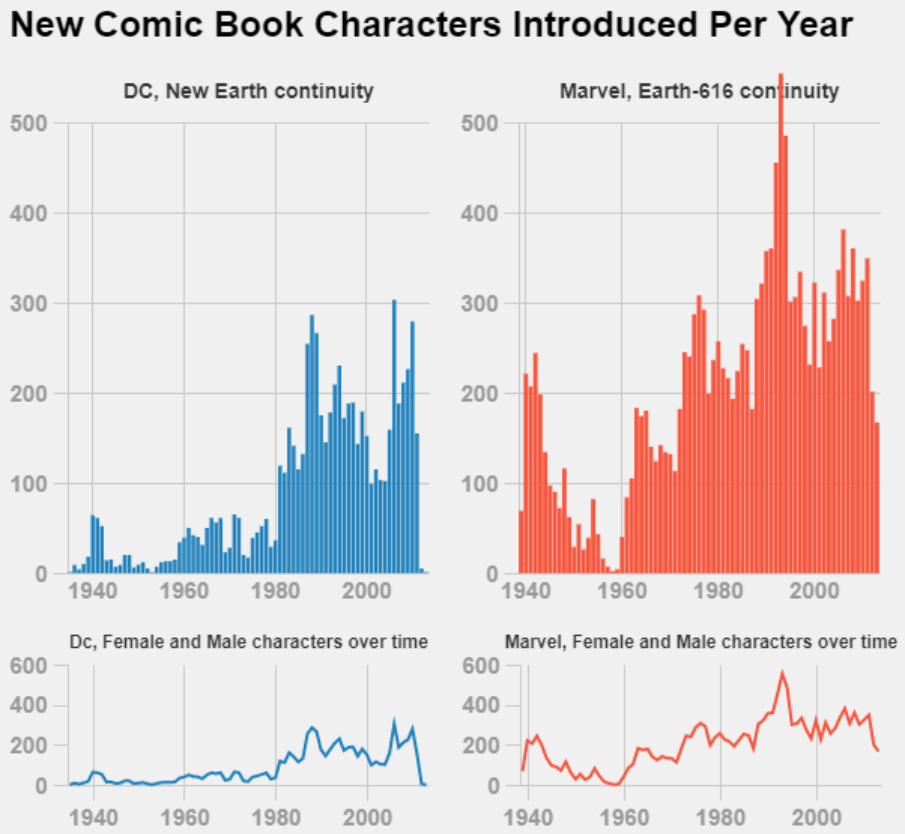

## Problem 2.1.5

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.encode(color=alt.condition(brush, alt.value('#2182bd'), alt.value('gray'))).transform_filter(brush).ad
i_line_marvel = line_marvel.encode(color=alt.condition(brush, alt.value('#f6573f'), alt.value('gray'))).transform_filter(b

# YOUR CODE HERE
# raise NotImplementedError()
```

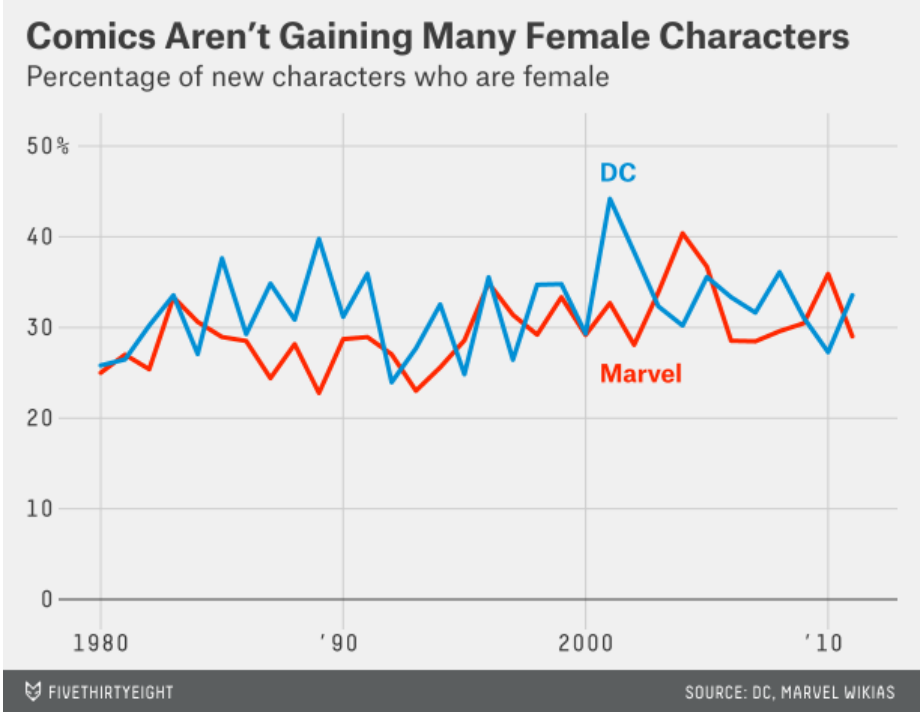
In [19]:
```
# let's put everything together with your new interactive charts. If you did everything correctly this part
# should generate the visualizations we want

# 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 [20]:

```
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',
                                                                         '% Female characters to date'])
```

In [21]:

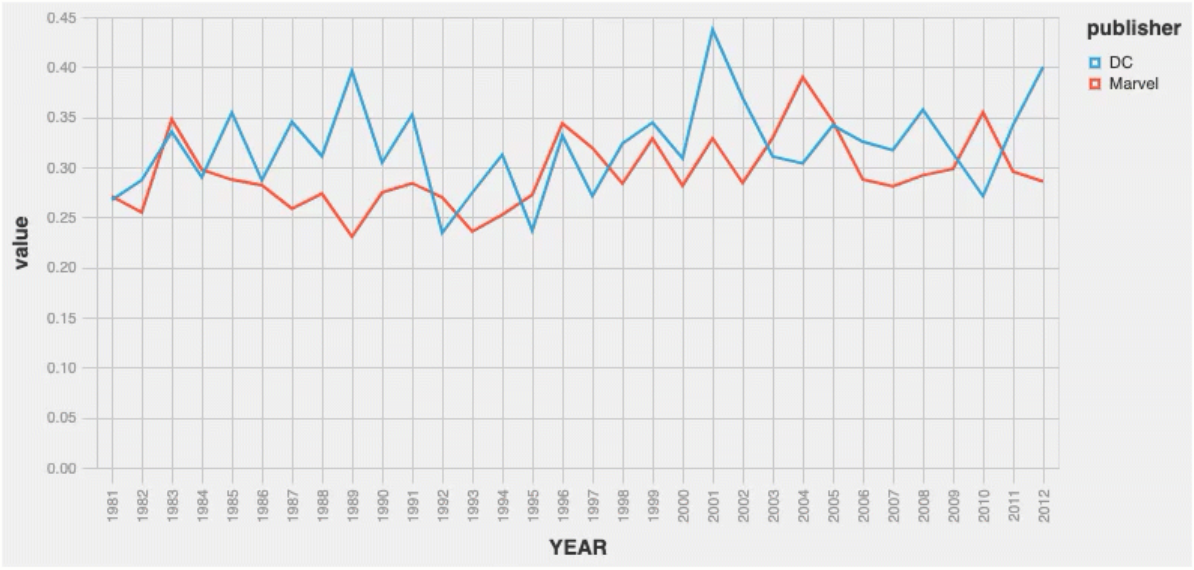
```
# let's see what's inside
changedata.sample(5)
```

Out[21]:

| | YEAR | publisher | variable | value |
|---|---|---|---|---|
| **177** | 1998 | DC | % Female characters to date | 0.309546 |
| **30** | 2011 | Marvel | % Female | 0.295522 |
| **70** | 1987 | Marvel | Year-over-year change in % Female | -0.083072 |
| **14** | 1995 | Marvel | % Female | 0.272414 |
| **107** | 1992 | DC | Year-over-year change in % Female | -0.335394 |

## 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:

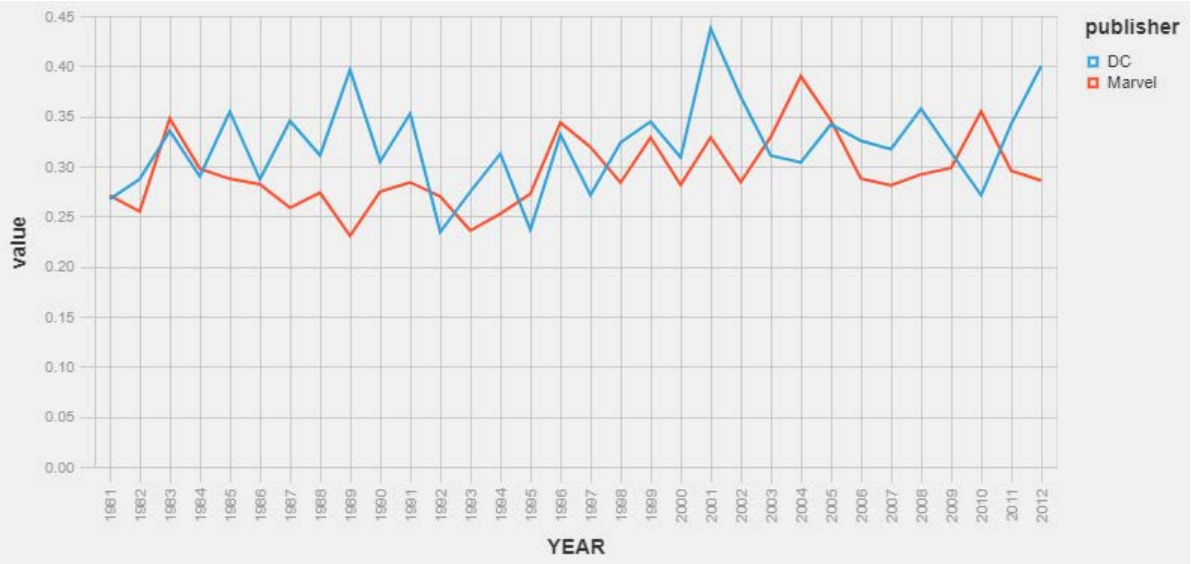



Modify `generateLineChartP21` below to generate this chart. If you haven't already, you'll want to take a look at the binding_select examples. Make sure you can get the chart working without interactivity first (hint: see if you can figure out how to filter to specific variables of interest).

In [22]:
```
def generateLineChartP21():

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

    dropdown_selection = alt.selection_single(fields=['variable'], bind=input_dropdown, name="Data_")

    line = alt.Chart(changedata).mark_line().encode(
                                        x=alt.X('YEAR:N'),
                                        y=alt.Y('value:Q'),
                                        color='publisher').add_selection(dropdown_selection).transform_filter(dropdown_sel

    # YOUR CODE HERE
    # raise NotImplementedError()
    return(line)
```

In [23]:
```
# if you did everything correctly, this should generate the visualization:
generateLineChartP21()
```

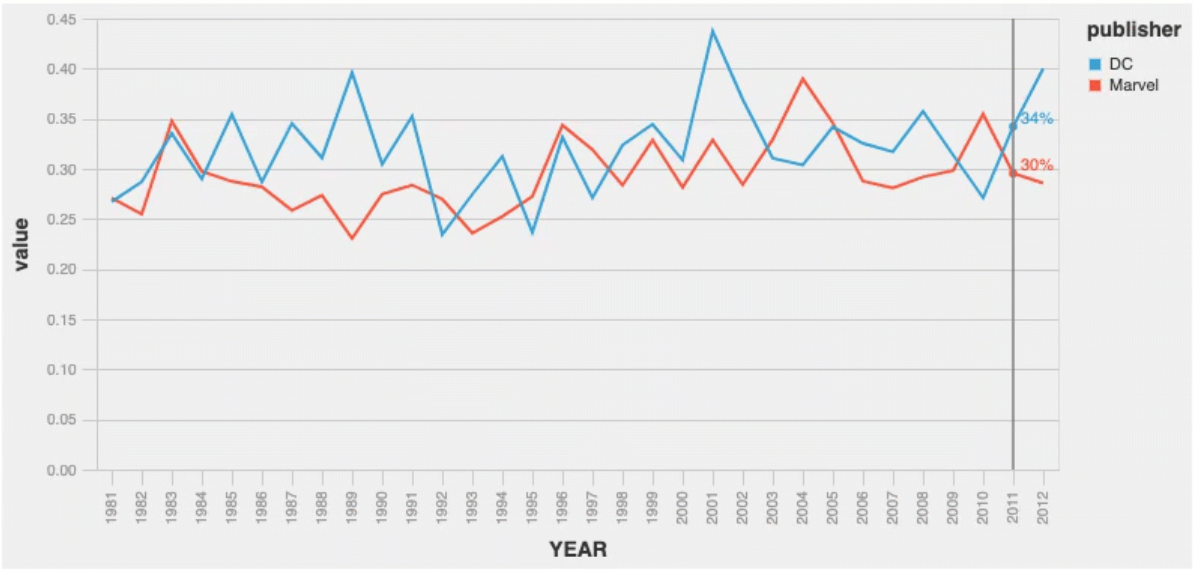
Out[23]:




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:




Data___variable % Female

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 [24]:

```
def createTool(frame):
    metricOptions = ['% Female','Year-over-year change in % Female','% Female characters to date']
    input_dropdown = alt.binding_select(options=metricOptions)
```

```
    dropdown_selection = alt.selection_single(fields=['variable'], bind=input_dropdown, name="Data_")

    # Create a selection that chooses the nearest point & selects based on x-value
    nearest = alt.selection(type='single', nearest=True, on='mouseover', fields=['YEAR'], empty='none')
    # Transparent selectors across the chart. This is what tells us the x-value of the cursor
    selectors = alt.Chart(frame).mark_point().encode(x='YEAR:N', y='value:Q', opacity=alt.value(0)).add_selection(nearest)
    # Draw points on the line, and highlight based on selection
    points = alt.Chart(frame).mark_point().encode(x='YEAR:N', y='value:Q', opacity=alt.condition(nearest, alt.value(1), al
    # Draw text labels near the points, and highlight based on selection
    text = alt.Chart(frame).mark_text(align='left', dx=5).encode(x='YEAR:N', y='value:Q', text=alt.condition(nearest, alt.
    # Draw a rule at the location of the selection
    rules = alt.Chart(frame).mark_rule(color='gray').encode(x='YEAR:N').transform_filter(nearest)

    return selectors, points, text, rules
```

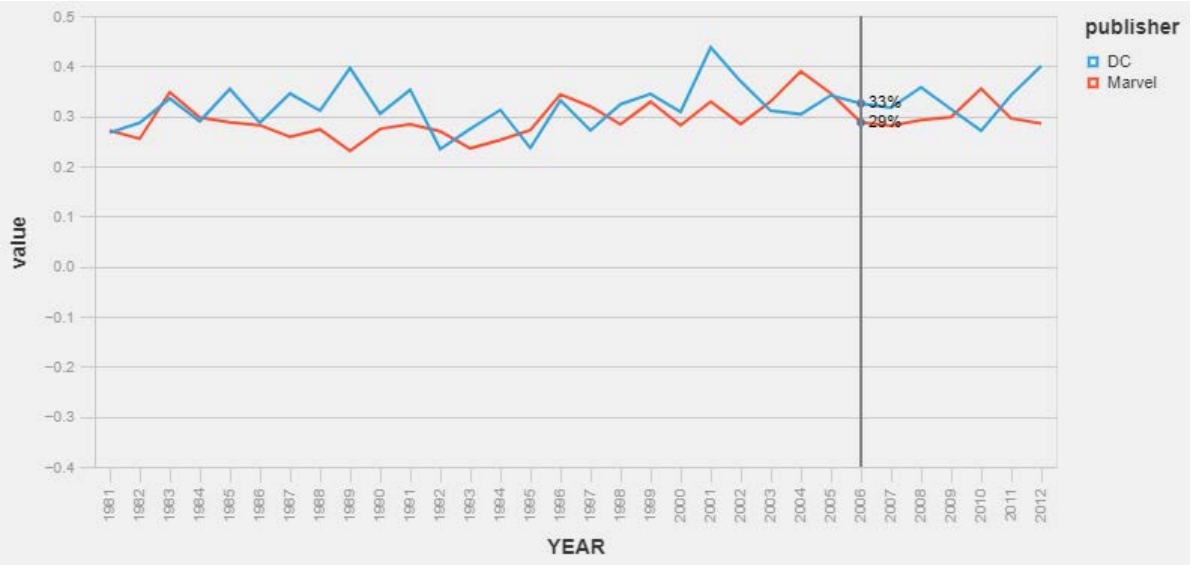
In [25]:

```
def generateLineChartP22():
    line_base = generateLineChartP21()

    selectors, points, text, rules = createTool(changedata)
    return alt.layer(line_base, selectors, points, text, rules)
    # YOUR CODE HERE
    # raise NotImplementedError()
```

In [26]:

```
generateLineChartP22()
```

Out[26]:




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 [27]:

```
# YOUR ANSWER HERE
```