Lab 3 Bechdel Test (Lily Orth-Smith)

March 6, 2016

1 Analysis of the Roles of Women in Movies in Connection to Profitability and Change in the Movie Industry over Time

1.1 Introduction

In this lab, I analyze a data set of around 1800 movies, released between 1970 and 2013. I examine whether the role women play in these movies is connected to their profitability. I use domestic gross, budget and return on investment as indicators of a movie's profitability, and the Bechdel test as a binary indicator of whether women play a significant role in a movie. A movie passes the Bechdel Test if it has two named women who talk to each other more than once about something other than men.

I also examined how the profitability of movies has changed over time, and how the difference in profitability betweeen movies that pass and movies that fail has changed over time.

In the cell below, I load in my data and functions I will use to analyze that data.

```
In [1]: %matplotlib inline
        import numpy as np
        import statistics as stat
        import matplotlib.pyplot as plt
        import scipy.stats
        import csv
        import math

        with open('movies.csv', 'r') as f:
            reader = csv.reader(f)
            data_list = list(reader)

        del(data_list[0])
        tuple_list = []
        removed = 0

        for kk in range(len(data_list)):
                #exclude datasets with N/A for domestic gross 2013 and international gross 2013
            if (data_list[kk][11] == "#N/A") or (data_list[kk][12] == "#N/A"):
                removed+=1
            else:
                        #0 title
                Tuple = (data_list[kk][2],
                        #1 year
                         int(data_list[kk][0]),
                        #2 binary
                        data_list[kk][5],
                        #3 clean test (how did movie fail)
                        data_list[kk][4],
                        #4 test (includes whether there was disagreement)
                        data_list[kk][3],
                        #5 budget (2013 dollars)
                        int(data_list[kk][10]),
                        #6 domestic gross (2013 dollars)
                        int(data_list[kk][11]),
                        #7 international gross (2013 dollars)
                        int(data_list[kk][12]),
                        #8 domestic gross/budget ratio
                        (int(data_list[kk][11])/int(data_list[kk][10])),
                        #9 international gross/budget ratio (2013 dollars)
                        (int(data_list[kk][12])/int(data_list[kk][10])),
                        #10 return on investment
                        ((int(data_list[kk][11])-int(data_list[kk][10]))/int(data_list[kk][10])))

            tuple_list.append(Tuple)

        #function creates a new tuple list conaining tuples from the original
        #operator can be ==, < or >
        def includeIfValue(some_tuple_list, element, operator, value):
            new_list = []
            if (operator == "=="):
                for ii in range(len(some_tuple_list)):
                    if some_tuple_list[ii][element] == value:
                        new_list.append(some_tuple_list[ii])
            elif  (operator == "<"):
                for ii in range(len(some_tuple_list)):
                    if some_tuple_list[ii][element] < value:
                        new_list.append(some_tuple_list[ii])
            elif (operator == ">"):
                for ii in range(len(some_tuple_list)):
                    if some_tuple_list[ii][element] > value:
                        new_list.append(some_tuple_list[ii])
            return new_list

        #reads in a tuple list, a function that returns a boolean (input is a tuple), and a value
        #returns a list of tuples
        def newTupleList(some_tuple_list, f, Value):
            new_list = []
            for ii in range(len(some_tuple_list)):
                if (f(some_tuple_list[ii], Value)):
                    new_list.append(some_tuple_list[ii])
            return new_list

        #prints an element of a tuple list
        def printElementOf(some_tuple_list, element):
            for ii in range(len(some_tuple_list)):
                print(some_tuple_list[ii][element])

        #turns an element of a tuple list into an array
        def tupleList(some_tuple_list, element):
            new_list = []
            for ii in range(len(some_tuple_list)):
                new_list.append(some_tuple_list[ii][element])
            return new_list

        #finds the percent of an element in a list that is a value
        def percent(some_tuple_list, element, value):
            elementPresent = 0
            for ii in range(len(some_tuple_list)):
                if some_tuple_list[ii][element] == value:
                    elementPresent+=1
            return (elementPresent/len(some_tuple_list))*100

        #averages a column in a tuple list (averageType = mean, median or mode)
        def averageTuple(some_tuple_list, element, averageType):
            new_list = []
            for ii in range(len(some_tuple_list)):
                new_list.append(some_tuple_list[ii][element])
            try:
                if averageType == "mean":
                    return stat.mean(new_list)
                if averageType == "median":
                    return stat.median(new_list)
                if averageType == "mode":
                    return stat.mode(new_list)
            except:
                print("No data points found")
                return 0

        #finds the correlation coefficient bewteen two elements in a tuple array
        def pearsonRTuple(some_tuple_list, element1, element2):
            new_list1 = []
            new_list2 = []
            for ii in range(len(some_tuple_list)):
                new_list1.append(some_tuple_list[ii][element1])
                new_list2.append(some_tuple_list[ii][element2])
            return scipy.stats.pearsonr(new_list1, new_list2)[0]

        #creates a histogram of an element in a tuple array, with a specified bin size
        #if bin_size = 0, auto sets binsize
        def createHist(some_tuple_list, element, bin_size):
            new_list = []
            for ii in range(len(some_tuple_list)):
                new_list.append(some_tuple_list[ii][element])
            maximum = int(math.ceil(max(new_list)))
            if bin_size != 0:
                bins_list = []
                for hh in range(int(math.ceil(maximum/bin_size))):
                    bins_list.append(bin_size*hh)
                plt.hist(new_list, bins_list)
            else:
                plt.hist(new_list)

        def createScatter(some_tuple_list, elementx, elementy):
            new_listx = []
            new_listy = []
            for ii in range(len(some_tuple_list)):
                new_listx.append(some_tuple_list[ii][elementx])
                new_listy.append(some_tuple_list[ii][elementy])
            plt.scatter(new_listx, new_listy)

        def createScatterColor(some_tuple_list, elementx, elementy, color):
            new_listx = []
            new_listy = []
            for ii in range(len(some_tuple_list)):
                new_listx.append(some_tuple_list[ii][elementx])
                new_listy.append(some_tuple_list[ii][elementy])
            plt.scatter(new_listx, new_listy, c=color)

        def newPlot(title, xlabel, ylabel):
            secondPlot = plt.figure()
            ax2 = secondPlot.add_subplot(111)
            plt.title(title)
            plt.xlabel(xlabel)
            plt.ylabel(ylabel)

        def newLogPlot(title, xlabel, ylabel):
            secondPlot = plt.figure()
            ax2 = secondPlot.add_subplot(111)
            ax2.set_yscale("log")
            plt.title(title)
            plt.xlabel(xlabel)
            plt.ylabel(ylabel)
```

1.2 1: Mean of Domestic Gross and Budget

Initially, I looked at the difference bewteen the mean of passing and failing movies' budgets and domestic grosses. I found that the average budget and domestic gross of passing movies was lower than the average of all movies, and lower than the average of failing movies.

```
In [2]: passing_movies = includeIfValue(tuple_list, 2, "==", "PASS")
        failing_movies = includeIfValue(tuple_list, 2, "==", "FAIL")

        #mean of budget
        print("All values in 2013 USD \n")
        print("Mean budget (all):", averageTuple(tuple_list, 5, "mean"))
        print("Mean budget (pass):", averageTuple(passing_movies, 5, "mean"))
        print("Mean budget (fail):", averageTuple(failing_movies, 5, "mean"), "\n")

        #mean of domestic gross
        print("Mean domestic gross (all):", averageTuple(tuple_list, 6, "mean"))
        print("Mean domestic gross (pass):", averageTuple(passing_movies, 6, "mean"))
        print("Mean domestic gross (fail):", averageTuple(failing_movies, 6, "mean"), "\n")

        #difference between average budget and domestic gross of passing and failing movies
        print("On average, movies that pass the Bechdel test have a budget $", (averageTuple(failing_mo
        print("On average, movies that pass the Bechdel test have a domestic gross $",(averageTuple(fai
```

```
All values in 2013 USD

Mean budget (all): 55806054.96376812
Mean budget (pass): 46602678.97375
Mean budget (fail): 63213198.61770624

Mean domestic gross (all): 95258685.79375696
Mean domestic gross (pass): 79412348.83125
Mean domestic gross (fail): 108012276.91046278

On average, movies that pass the Bechdel test have a budget $ 16610519.643956237 less than
movies that fail the test
On average, movies that pass the Bechdel test have a domestic gross $ 28599928.079212785 less than movi
```

1.3 2: Mean Return on Investment (Domestic Gross divided by Budget)

The official definition of return on investment is the profits divided by the cost, or total sales minus budget divided by budget. Since domestic gross of a movie is the total sales of tickets, domestic gross divided by budget does not fit this this definition. However, domestic gross is not the total profits of the movie–it only includes box office sales. In addition, subtracting the budget from the domestic gross before dividing brings the ratio down by exactly 1.0 in every case, so it does not effect any correlation between data points. Because of this, although the ratio of domestic gross to budget does not fit the conventional definition of return on investment, I will refer to the ratio of domestic gross to budget as return on investment.

Below I find the mean return on investment for all movies, movies that pass, and movies that fail.

```
In [3]: #mean domestic gross/budget
        print("Mean return on investment (all):", averageTuple(tuple_list, 8, "mean"))
        print("Mean return on investment (pass):", averageTuple(passing_movies, 8, "mean"))
        print("Mean return on investment (fail):", averageTuple(failing_movies, 8, "mean"), "\n")
```

```
Mean return on investment (all): 3.819204502965776
Mean return on investment (pass): 2.9825565024981877
Mean return on investment (fail): 4.492563054649953
```

On average, failing movies make around 1.5 times their budget more than passing movies. This makes it appear that, on average, failing movies are more profitable than passing movies.

1.4 3: Histograms of Return on Investment

When we look at histograms of the return on investment this becomes visually clear. Figures one through three, displayed below, show the return on investment on the x scale, and the number of data points in that range on the y scale.

1.4.1 Figure 1: Return on Investment of All Movies

```
In [4]: newLogPlot("Figure 1:\n Return on Investment of All Movies", "Return on Investment", "Occurence
        createHist(tuple_list, 8, 15)
        plt.axis([0, 300, 10**0, 10**4])
```

```
Out[4]: [0, 300, 1, 10000]
```

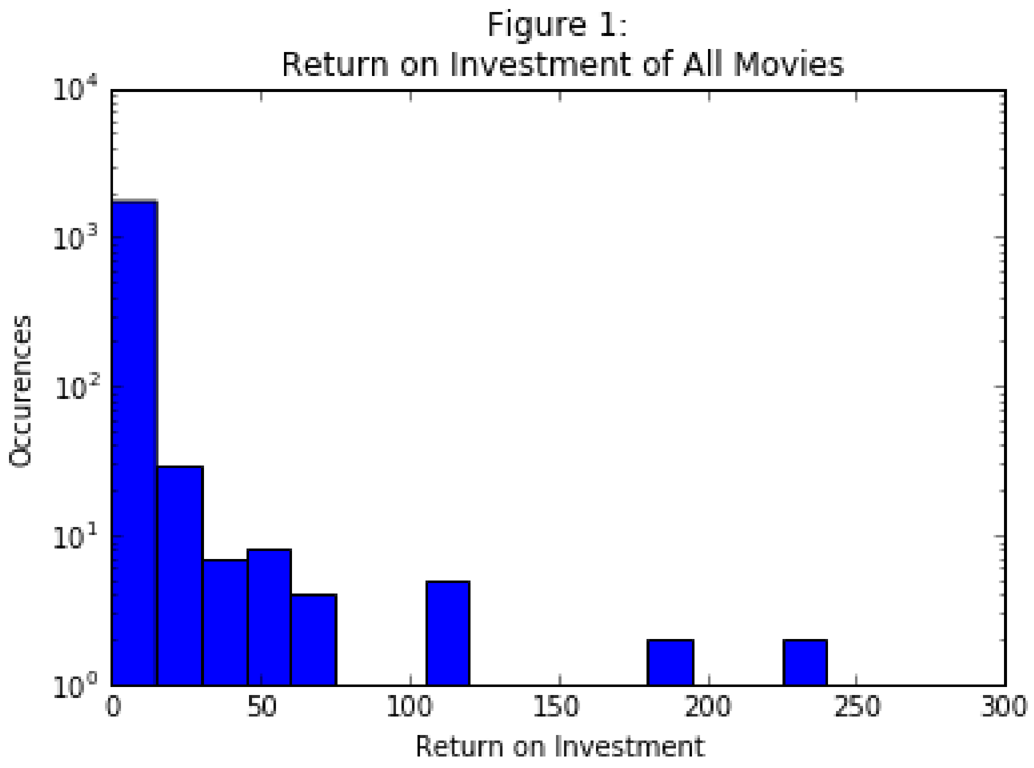

Figure 1 (above) is a histogram of the return on investment for all movies on a log scale. It shows that most movies gross between 0 and 15 times their budget, with a few outliers, which gross 100 or 200 times their budget.

1.4.2 Figure 2: Return on Investment of Passing Movies

```
In [5]: newLogPlot("Figure 2:\n Return on Investment of Passing Movies", "Return", "Occurences")
        createHist(passing_movies, 8, 15)
        plt.axis([0, 300, 10**0, 10**4])

Out[5]: [0, 300, 1, 10000]
```

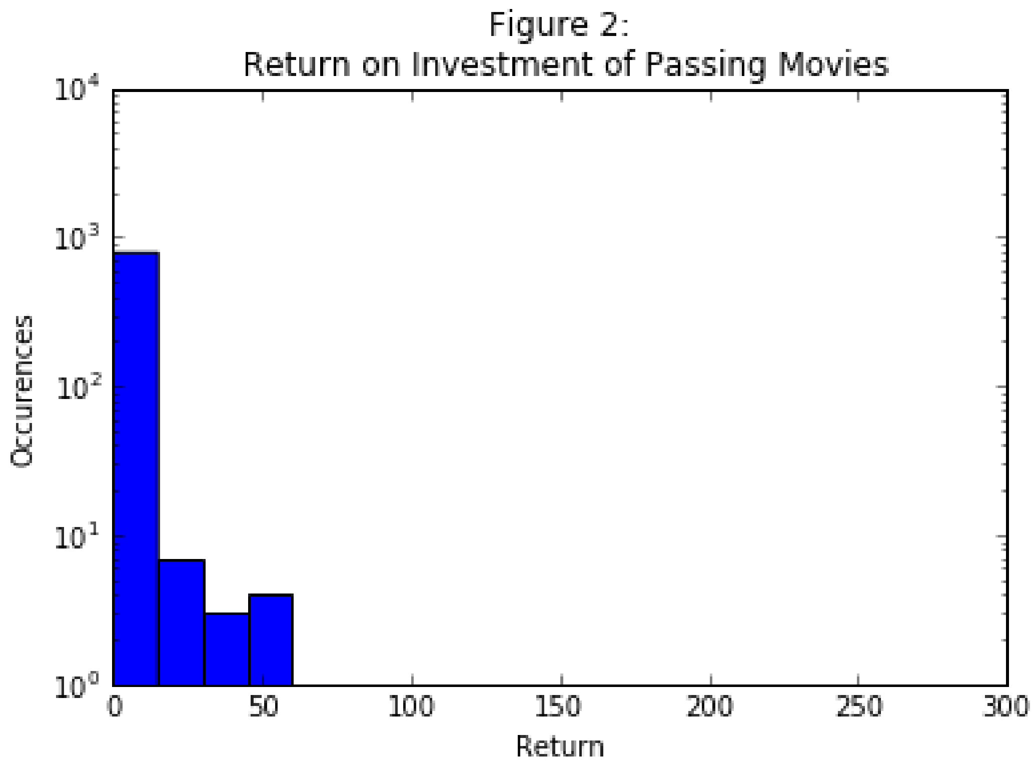

Figure Two (above) is a histogram of the return on investment for movies that pass the Bechdel test on a log scale. Of the movies that pass, most movies gross between 0 and 15 times their budget. And, there are no movies that gross more than around 50 times their budget.

1.4.3 Figure 3: Return on Investment of Failing Movies

```
In [6]: newLogPlot("Figure 3:\n Return on Investment of Failing Movies", "Return on Investment", "Occur
        createHist(failing_movies, 8, 15)
        plt.axis([0, 300, 10**0, 10**4])

Out[6]: [0, 300, 1, 10000]
```

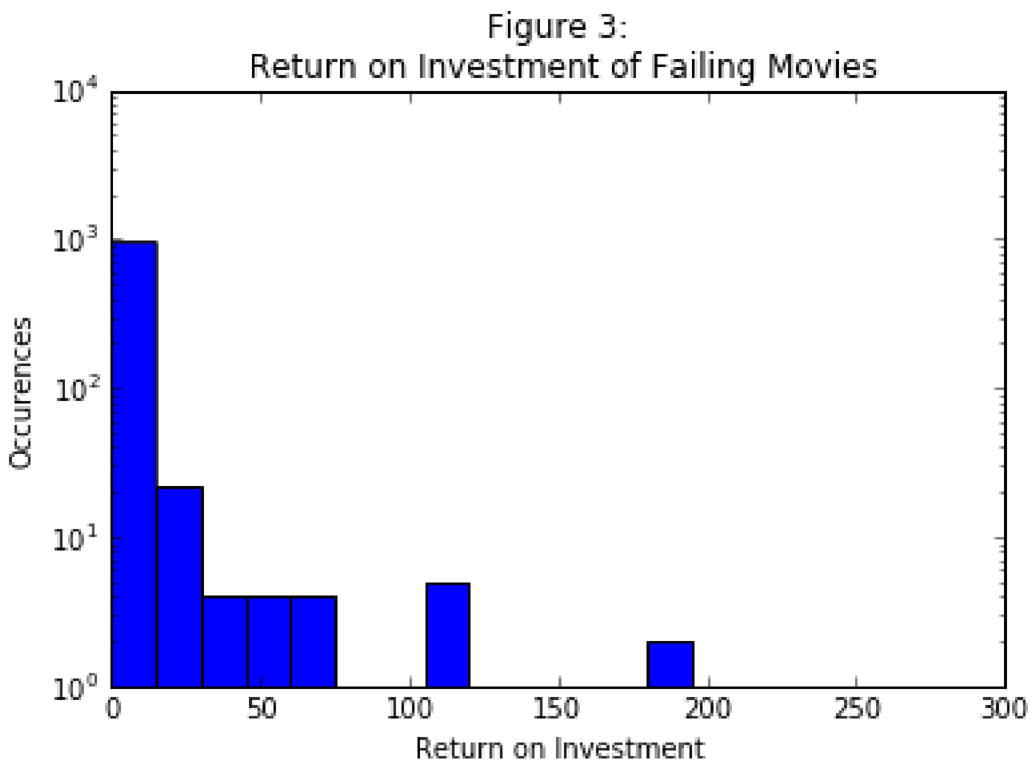

Figure 3 (above) is a histogram of the return on investment for movies that fail the Bechdel test on a log scale. Like in Figure 1 and Figure 2, most movies gross between 0 and 15 times their budget, with a few outliers grossing 100 to 200 times their budget.

*Note: I am not sure where the movies in the bin just below 250 went in Figure 3. I believe they should be there. I checked that no data lost when I split tuple_list into passing_movies and failing_movies, so it is probably an issue with the histogram. I am not sure how to fix it.

Figures 1, 2 and 3 show that the vast majority of movies (around 1000 passing and 1000 failing movies) gross between 0 and 15 times their budget. There are far fewer movies that have a return on investment of around 50, and even fewer that have one above 100– only two or three in a data set of around 1800. However, all of these outliers fail the Bechdel test: they are only present in Figure 3, but not in Figure 2.

1.5 4: Correlation Between Budget and Domestic Gross

I quantified the difference in the distribution of passing and faling movies using pearson r correlation coefficiencts. The closer the correlation coefficient is to one, the more correlated the two variables are.

```
In [7]: #correlation coefficient of budget and domestic gross
        print("Correlation Coefficienct of Budget and Domestic Gross (all):", pearsonRTuple(tuple_list,
        print("Correlation Coefficienct of Budget and Domestic Gross (pass):", pearsonRTuple(passing_mo
        print("Correlation Coefficienct of Budget and Domestic Gross (fail):", pearsonRTuple(failing_mo
```

```
Correlation Coefficienct of Budget and Domestic Gross (all): 0.455300863937
Correlation Coefficienct of Budget and Domestic Gross (pass): 0.581798092307
Correlation Coefficienct of Budget and Domestic Gross (fail): 0.382366618619
```

I found that budget and domestic gross are more correlated for passing movies than failing movies. The fact that passing movies have a higher correlation coefficient than failing movies (by 0.2) indicates that the list of failing movies has more outliers than the passing list. And, these outliers can be seen in Figures 1, 2 and 3 (above).

Because the mean of failing movies' return on investment is higher than the mean of passing movies, we can conclude that the outliers in the list of failing movies pulled the mean return of investment up for those movies.

However, we can't exclude these outliers from the data set simply because they have a return on investment, because these movies are the movies Hollywood aims to produce. And, because these high-grossing movies are very popular, they reflect the public's tastes. If failing movies were consistently more popular and higher grossing than passing movies, this would reveal a bias in the American public towards movies that do not represent women.

1.6 5: Restricting the Data Set to Years 2000-2013

However, there is a connection between these outliers. Women's role in society has changed a lot since the 1970s, and so has the performance of movies that pass the Bechdel test. When the data set is restricted to movies made between 2000 and 2013, the gap between passing and failing movies shrinks dramatically.

1.6.1 5.1: Mean Return on Investment (2000-2013)

When the data set was restricted to movies released between 2000 and 2013, the difference between mean return on investment for passing movies and failing movies was much smaller than the difference in mean return on investment over the whole data set.

```
In [8]: post_1999 = includeIfValue(tuple_list, 1, ">" , 1999)
        print("Mean return on investment post 1999 (all):",averageTuple(post_1999, 8, "mean"))
        post_1999p = includeIfValue(post_1999, 2, "==", "PASS")
        print("Mean return on investment post 1999 (pass):", averageTuple(post_1999p, 8, "mean"))
        post_1999f = includeIfValue(post_1999, 2, "==", "FAIL")
        print("Mean return on investment post 1999 (fail)", averageTuple(post_1999f, 8, "mean"))
```

```
Mean return on investment post 1999 (all): 2.4144101913430167
Mean return on investment post 1999 (pass): 2.33636151146807
Mean return on investment post 1999 (fail) 2.4841336786979693
```

The difference between mean return on investment of failing and passing movies (released between 2000 and 2013) is around 0.148. In other words, if a passing movie and a failing movie were made with a budget of $1, the failing movie would gross around 0.15 cents more than the passing movie.

However, the difference in mean return on investment of failing and passing movies made between 1970 and 2013 is 1.51. So, if a passing movie and a failing movie were made with a budget of $1, the failing movie would gross around $1.51 more than the passing movie (see section 2 for this data).

1.6.2 5.2: Correlation Coeffient between Budget and Domestic Gross (2000-2013)

In addition, in the restricted data set, the correlation coefficients of budget and domestic gross for passing movies and failing movies is very close, indicating that the distributions of passing movies and failing movies released between 2000 and 2013 are very similar.

```
In [9]: print("Correlation Coefficienct of Budget and Domestic Gross (all)", pearsonRTuple(post_1999, 5
        print("Correlation Coefficienct of Budget and Domestic Gross (pass)", pearsonRTuple(post_1999p,
        print("Correlation Coefficienct of Budget and Domestic Gross (fail)", pearsonRTuple(post_1999f,
```

```
Correlation Coefficienct of Budget and Domestic Gross (all) 0.69893290357
Correlation Coefficienct of Budget and Domestic Gross (pass) 0.690944103243
Correlation Coefficienct of Budget and Domestic Gross (fail) 0.701268070219
```

The difference between the correlation coefficients of budget and domestic gross for passing movies and failing movies in the restricted data set is around 0.01, however the difference is 0.2 for the unrestricted data set (section 4).

The results of 5.1 and 5.2 suggest that the relationship between budget and domestic gross changes over time, and that difference between the performance of passing and failing movies has decreased over time.

1.7 6: Plotting Mean Return on Investment over Time

When the mean return of investment is plotted over time, it becomes clear that the profitability of movies has changed a lot over time–and the gap in profitability between passing and failing movies has shrunk.

In the graphs below, I found the mean return on investment of all movies released over the course of one year, and plotted those means against time.

```
In [10]: mean_decade = []

         mean_year = []
         mean_yearp = []
         mean_yearf = []

         mean_gross_year = []
         mean_gross_yearp = []
         mean_gross_yearf = []

         mean_budget_year = []
         mean_budget_yearp = []
         mean_budget_yearf = []

         year = []
         #checks what decade the movie was made in
         def decade(some_tuple, aValue):
             if (str(some_tuple[1])[2]==aValue):
                 return True
             else:
                 return False

         #creates a list of the means of each decade's domestic gross/budget ratio
         for kk in [1, 0, 9, 8, 7]:
             mean_decade.append(averageTuple(newTupleList(tuple_list, decade, str(kk)), 8, "mean"))

         #creates a list of means per year of gross, budget, return for all movies passing movies, and
         for hh in range(1973, 2014):
             #mean return per year
             mean = (averageTuple(includeIfValue(tuple_list, 1, "==", hh), 8, "mean"), hh)
             meanp = (averageTuple(includeIfValue(passing_movies, 1, "==", hh), 8, "mean"), hh)
             meanf = (averageTuple(includeIfValue(failing_movies, 1, "==", hh), 8, "mean"), hh)
             #mean gross per year
             mean_gross = (averageTuple(includeIfValue(tuple_list, 1, "==", hh), 6, "mean"), hh)
             mean_grossp = (averageTuple(includeIfValue(passing_movies, 1, "==", hh), 6, "mean"), hh)
             mean_grossf = (averageTuple(includeIfValue(failing_movies, 1, "==", hh), 6, "mean"), hh)
             #mean budget per year
             mean_budget = (averageTuple(includeIfValue(tuple_list, 1, "==", hh), 5, "mean"), hh)
             mean_budgetp = (averageTuple(includeIfValue(passing_movies, 1, "==", hh), 5, "mean"), hh)
             mean_budgetf = (averageTuple(includeIfValue(failing_movies, 1, "==", hh), 5, "mean"), hh)
```

```
            #checks if the mean is 0 (there were no data points in that year), then appends tuple to m
            #mean return
            if (mean[0] != 0):
                mean_year.append(mean)
            if (meanp[0] != 0):
                mean_yearp.append(meanp)
            if (meanf[0] != 0):
                mean_yearf.append(meanf)
            #mean gross
            if (mean_gross[0] != 0):
                mean_gross_year.append(mean_gross)
            if (mean_grossp[0] != 0):
                mean_gross_yearp.append(mean_grossp)
            if (mean_grossf[0] != 0):
                mean_gross_yearf.append(mean_grossf)
            #mean budget
            if (mean_budget[0] != 0):
                mean_budget_year.append(mean_budget)
            if (mean_budgetp[0] != 0):
                mean_budget_yearp.append(mean_budgetp)
            if (mean_budgetf[0] != 0):
                mean_budget_yearf.append(mean_budgetf)
```

```
No data points found
No data points found
No data points found
```

*In years where there were no data points (passing, failing or otherwise), no mean was plotted.

1.7.1 Figure 4: Mean Return on Investment Over Time

```
In [11]: newPlot("Figure 4: \n Mean Return on Investment Per Year (all)", "Year", "Return on Investment
         createScatterColor(mean_year, 1, 0, "blue")
```

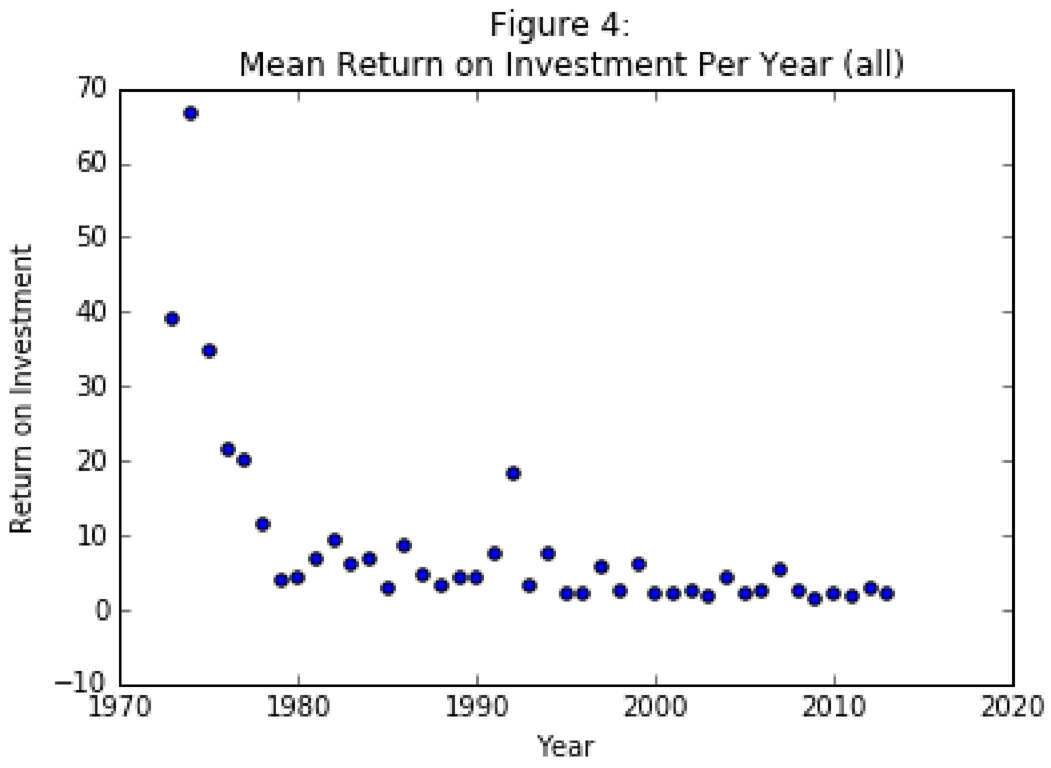

Figure 4 (above) is a scatter plot of the mean return on investment for all movies released in a given year, plotted over time.

1.7.2 Figure 5: Mean Return on Investment Over Time

```
In [12]: newPlot("Figure 5: \n Mean Return on Investment Per Year (pass=green, fail=orange)", "Year", "
         createScatterColor(mean_yearp, 1, 0, "green")
         createScatterColor(mean_yearf, 1, 0, "orange")
```

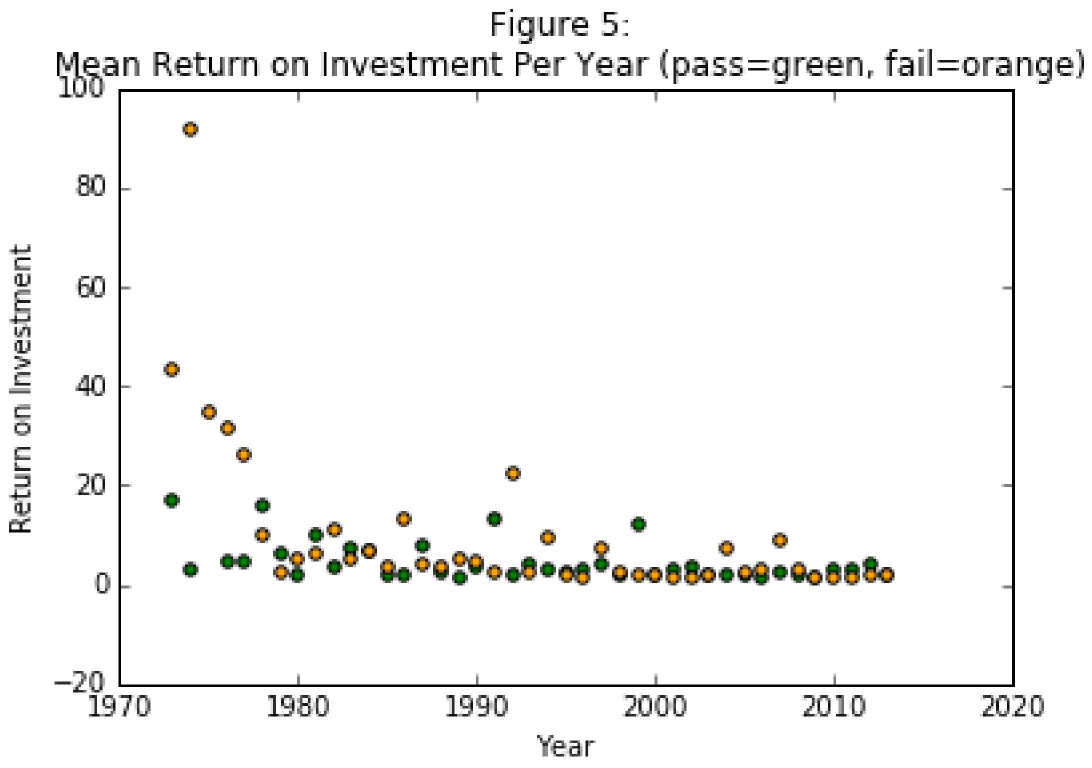

Figure 5 (above) is a scatter plot of the mean return on investment of movies that passed and movies that failed in a given year, plotted over time. The means of movies that pass are rendered in green, and the means of failing movies are rendered in orange. Means that had no data were excluded from the plot.

Figure 4 illustrates that the return on investment of movies has dropped over time. And, Figure 5 shows that the mean return on investment has been converging for passing and failing movies. In fact, most years when failing movies had very high return on investments were in the 1970s. After that, the mean return on investment falls, and the mean return on investment for passing and failing movies converge.

This shows that over time, the profitability of movies has decreased, and the difference in profitability between passing and failing movies has also decreased. In fact, in between 2010 and 2013, passing movies had a slightly higher return on investment that failing movies had.

1.8 7: Lower Budget, Same Return on Investment

Although average return on investment and the average domestic gross for passing and failing movies have been converging, the average budget for passing and failing movies have not.

1.8.1 Figure 6: Mean Domestic Gross and Budget over Time

```
In [13]: newPlot("Figure 6: \n Mean Domestic Gross and Budget over Time (all)", "Year", "Dollars (2013 
         plt.axis([1970, 2020, 0, 0.7*10**9])
         createScatterColor(mean_gross_year, 1, 0, "purple")
         createScatterColor(mean_budget_year, 1, 0, "pink")
```

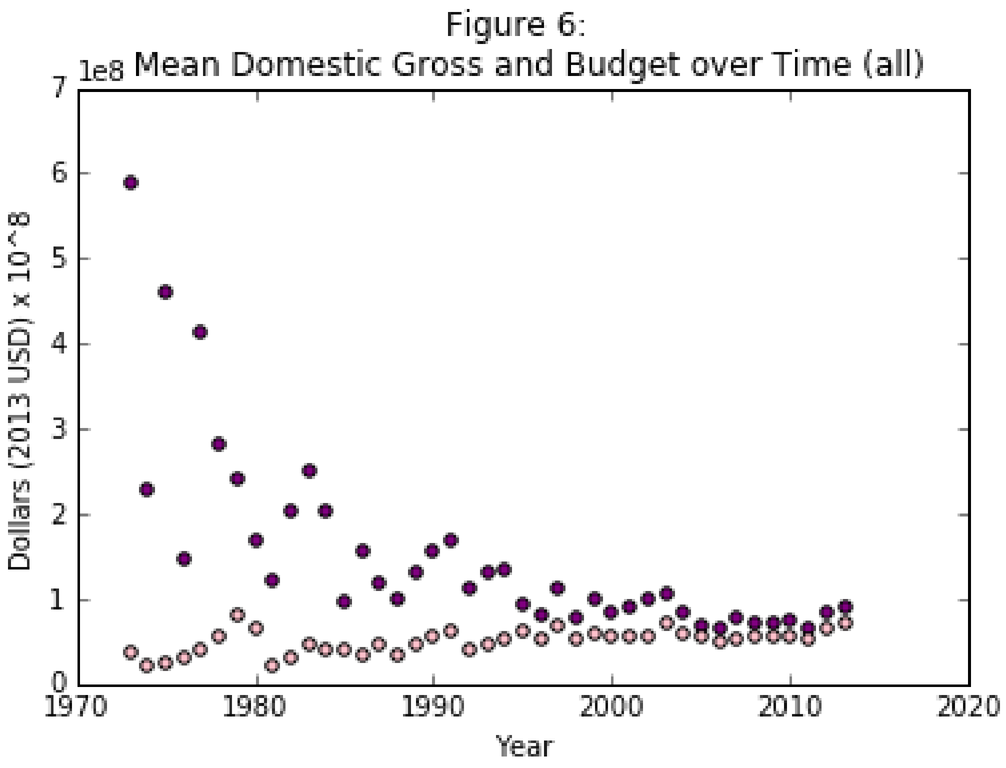

Figure 6 shows mean domestic gross, in purple, and mean budget, in pink, plotted against time. Over time, the mean gross has gone down, while the mean budget has gone up.

1.8.2 Figure 7: Mean Domestic Gross over Time (passing and failing)

```
In [14]: newPlot("Figure 7: \n Mean Domestic Gross over Time (passing, green; failing, red) \n", "Year"
         plt.axis([1970, 2020, 0, 0.7*10**9])
         createScatterColor(mean_gross_yearp, 1, 0, "green")
         createScatterColor(mean_gross_yearf, 1, 0, "red")
```

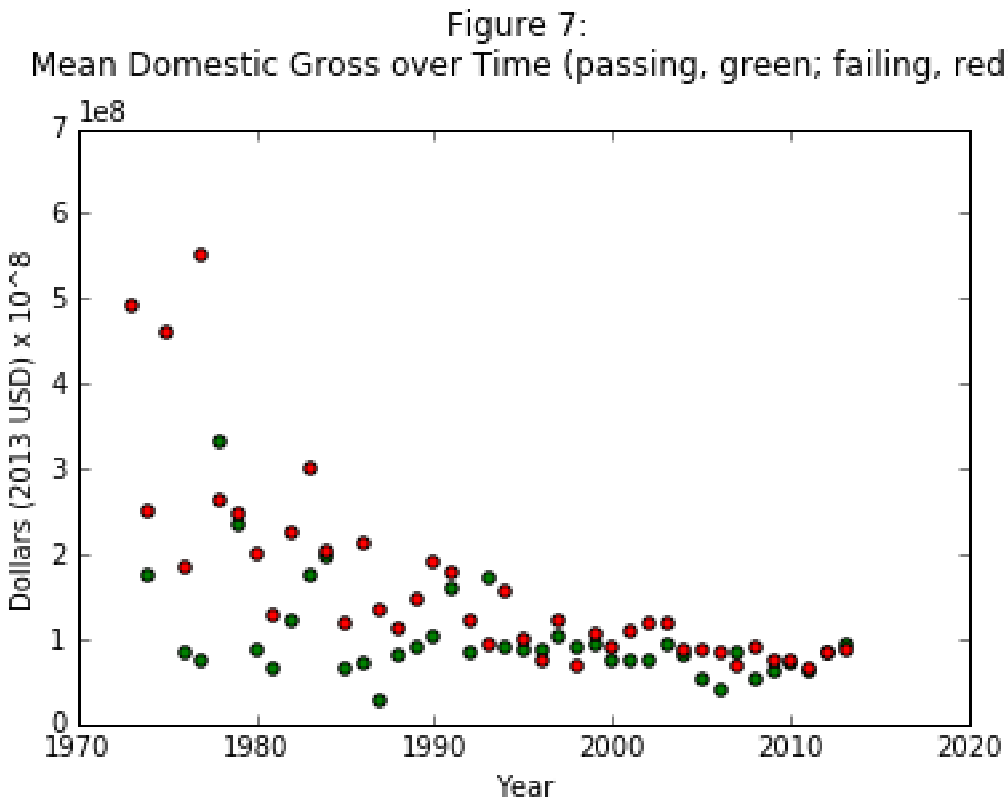

Figure 7 shows mean domestic gross for passing movies (green), and failing movies (red), over time. As time went on, the mean domestic gross for passing and failing movies has grown closer. Between 2010 and 2013, it appears that the domestic gross
*The mean for passing movies in 1973 was excluded from this plot, as it was an outlier. The mean of that year contained one data point, the Excorcist, which grossed around $1.1 billion in USD.

1.8.3 Figure 8: Mean Budget over Time (passing and failing)

```
In [15]: newPlot("Figure 8: \n Mpassing (green) and failing (orange) budget", "Dollars (2013 USD) x 10^8
         plt.axis([1970, 2020, 10**0, 0.4*10**9])
         createScatterColor(mean_budget_yearp, 1, 0, "green")
         createScatterColor(mean_budget_yearf, 1, 0, "red")
```

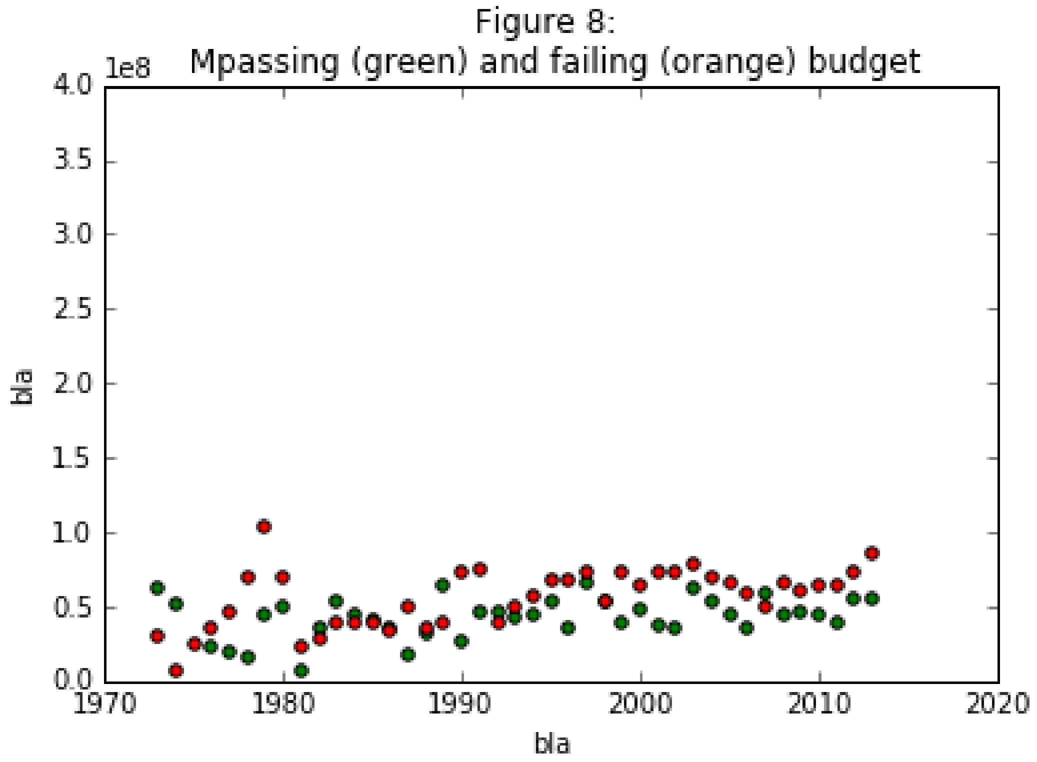

Figure 9 shows mean budget over time for passing movies (green) and failing movies (red). While the mean domestic gross for passing and failing movies (Figure 8) have converged over time, the mean budget of passing movies remains below the mean budget of failing movies. In fact, between 1995 and 2013, it appears that the mean budget for passing and failing movies have been diverging, not converging.

This shows that although the profitability of passing movies has drawn closer to the profitability of failing movies, the movie industry does not invest as much money in passing movies as failing movies. In recent years, however, passing movies have still been high-grossing. Between 2010 and 2013, passing movies and failing movies had very similar domestic grosses (Figure 7), even though the mean budget for passing movies was much lower (Figure 8).

1.9 Conclusion

At first, it appears that movies that pass the Bechdel test are not as profitable as movies that fail the Bechdel test, because the mean gross and mean return on investment is higher for movies that fail. However, when the passage of time is taken into account, it becomes clear that the difference in profitability for passing and failing movies has decreased: passing and failing movies released between 2010 and 2013 perform similarly, but passing and failing movies released in the 70s performed very differently.

However, the dramatic difference in return on investment and domestic gross could be due to selection bias. The data set I analyzed is crowd-sourced, so in early years, there are less movies in the data base, because people don't add them. And, many movies that were added in the early years are very famous, high-grossing movies such as The Excorcist, The Godfather, Charlie and the Chocolate Factory, The Texas Chain Saw Massacre, and Escape from the Planet of the Apes. As time went, however, more and more movies were added, so the sample gives a better approximation of the population at that time.

Very recently (between 2010 and 2013), it appears that movies that pass the Bechdel test had a slightly higher return on investment than failing movies, because the gross of passing movies was similar to that of failing movies, while passing movie's budgets remained much lower.

Based on this data, and the fact that high-budget passing movies, such as the Excorcist, have had very high grosses as early as 1973, I believe that today, passing movies' profitability is the same, if not greater, than failing movies' profitability.

Which raises the question: If passing movies are about as profitable as failing movies, why arent' all movies And why do they have consistently lower budgets? Since there is no good economic reason, the answer must be social. Perhaps unconciously, male producers do not write movies that have many women in them. Or perhaps people believe that movies with women in them will be less successful, and underfund them. Either way, there is no reason that less than half of movies in this data set should pass the Bechdel test. It's 2016. All of them should pass.