Project3

November 15, 2018

1 Stat 305, Project 3

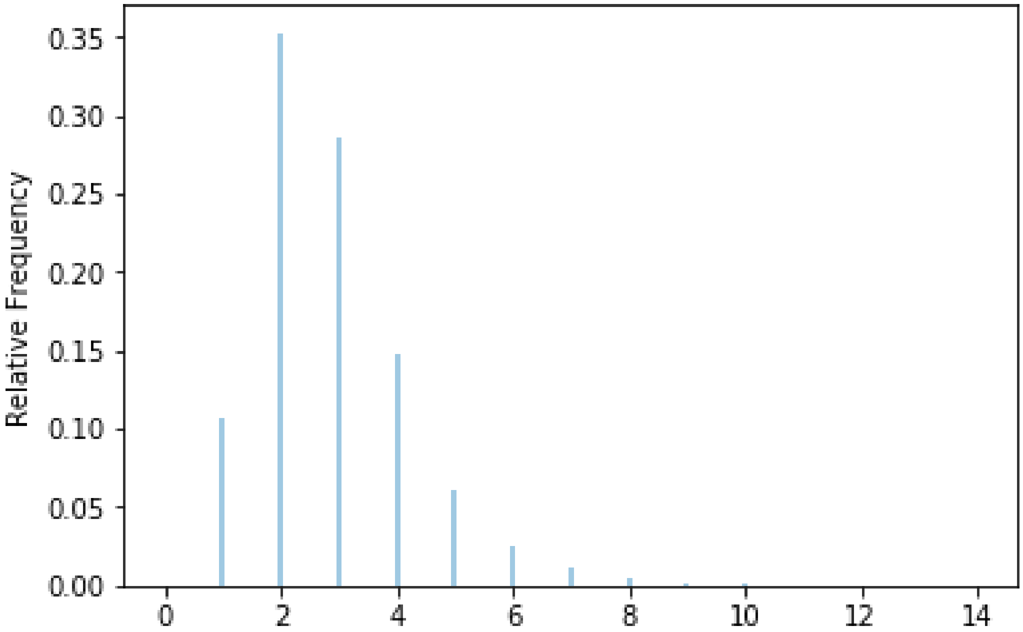
```
In [4]: #David Barnett and Ajay Patel
        #STAT 305-02

        #Our project is based on the football with a slightly below average team such as the D
        #so we will investigate the streaks of winning and losing in a season.
```

```
In [5]: from symbulate import *
        %matplotlib inline
```

```
In [6]: #returns the largerst success streak with counter variables
        def count_streak_of_success(omega):
            streak = 0
            curmax = 0
            for i, w in enumerate(omega):
                if w == 0:
                    if curmax < streak:
                        curmax = streak
                    streak = 0
                if w == 1:
                    streak += 1
            return curmax

        #returns the largest losing streak with counter variables
        def count_streak_of_failure(omega):
            streak = 0
            curmax = 0
            for i, w in enumerate(omega):
                if w == 1:
                    if curmax < streak:
                        curmax = streak
                    streak = 0
                if w == 0:
                    streak += 1
            return curmax
        #returns the larger of the 2 streaks with counter variables
        def max_streak(omega):
            streak_0 = 0
            streak_1 = 0
            curmax = 0
            for i, w in enumerate(omega):
                if w == 1:
                    if curmax < streak_1:
                        curmax = streak_1
                    streak_1 = 0
                    streak_0 += 1
                if w == 0:
                    if curmax < streak_0:
                        curmax = streak_0
                    streak_1 += 1
                    streak_0 = 0
            return curmax

        #returns amount of times success(win) streak > failure(losing) streak
        def prob_streak(omega):
            streak_0 = 0
            streak_1 = 0
            curmax = 0
            for i, w in enumerate(omega):
                if w == 1:
                    if curmax < streak_1:
                        curmax = streak_1
                    streak_1 = 0
                    streak_0 += 1
                if w == 0:
                    if curmax < streak_0:
                        curmax = streak_0
                    streak_1 += 1
                    streak_0 = 0
            if curmzx_0 > curmax_1:
                return True
            else:
                return False
```

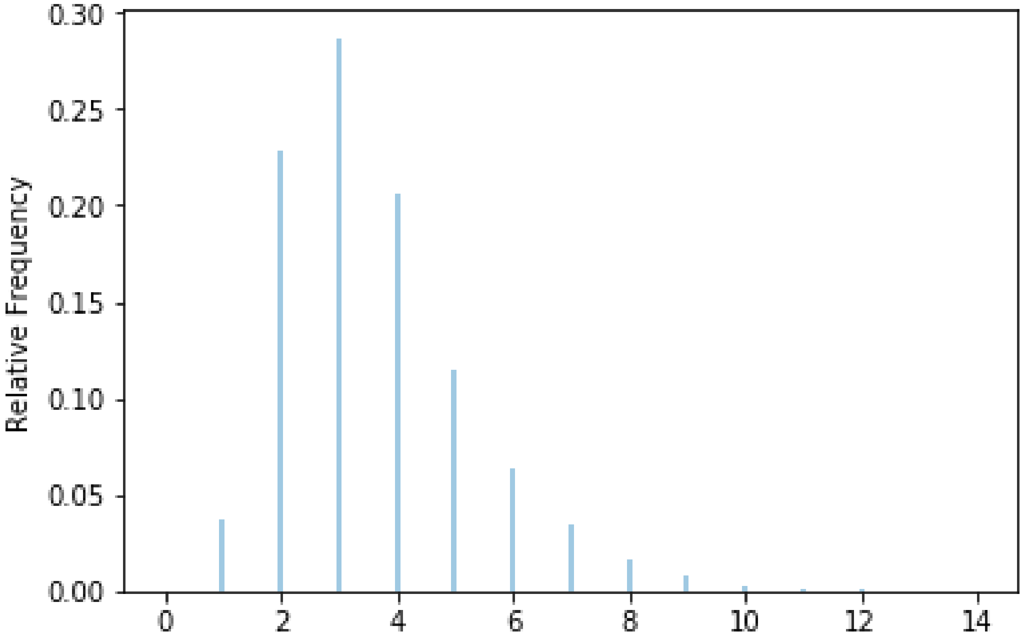
```
In [7]: #1a) Distribution of X with P(success)=.45

        P = BoxModel([1, 0], probs=[.45, .55], size=16)
        x = count_streak_of_success
        X = RV(P, x)
        X.sim(10000).plot()
        X.sim(10000).tabulate()
```

```
Out[7]: {2: 3501,
         4: 1423,
 3: 2902,
 1: 1077,
 9: 11,
 5: 653,
 6: 259,
 10: 7,
 7: 111,
 12: 4,
 8: 47,
 0: 3,
 11: 2}
```

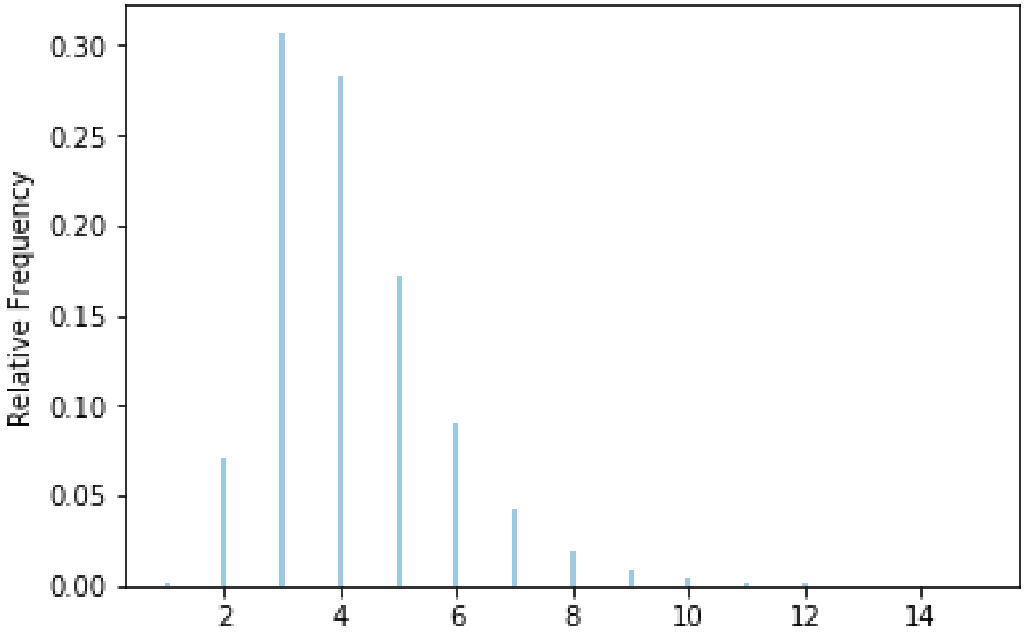


```
In [8]: #1a) Distribution of Y with P(Failure or Losing) = .55

        P = BoxModel([1, 0], probs=[.45, .55], size=16)
        y = count_streak_of_failure
        Y = RV(P, y)
        Y.sim(10000).plot()
        Y.sim(10000).tabulate()
```

```
Out[8]: {3: 2975,
         5: 1184,
         4: 2015,
         2: 2305,
         7: 280,
         6: 577,
         1: 378,
         8: 152,
         9: 68,
         10: 39,
         12: 9,
         13: 1,
         0: 1,
         11: 13,
         14: 3}
```

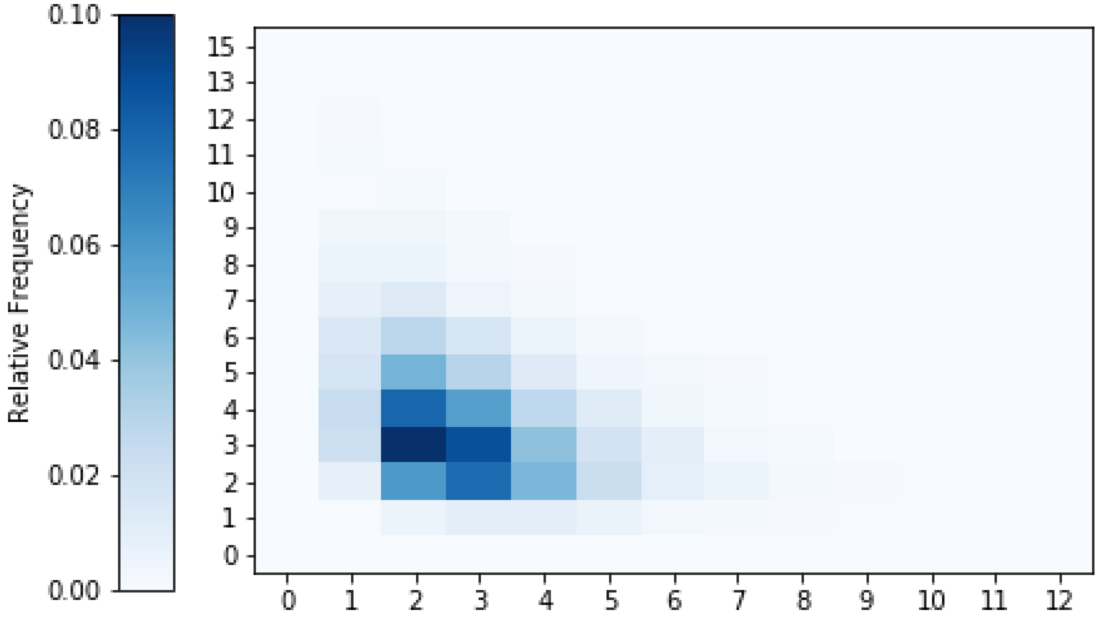


```
In [9]: #1a) Distribution of Z
        P = BoxModel([1, 0], probs=[.45, .55], size=16)
        z = max_streak
        Z = RV(P, z)
        Z.sim(10000).plot()
```



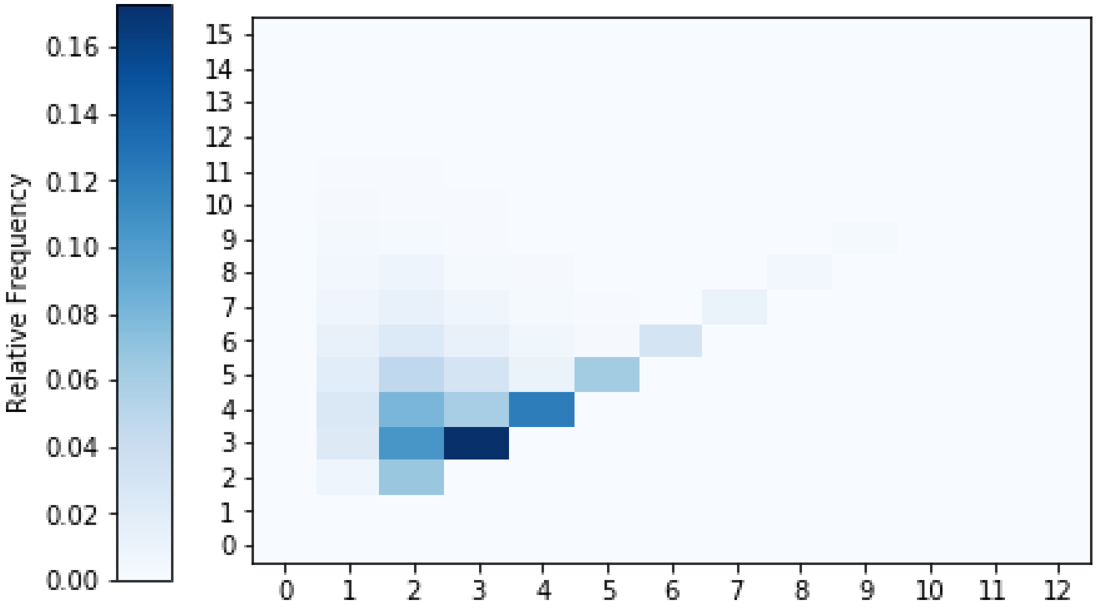
```
In [10]: #1b)
         P = BoxModel([1, 0], probs=[.45, .55], size=16)
         x = count_streak_of_success
         X = RV(P, x)
         y = count_streak_of_failure
         Y = RV(P, y)
         xy = (X & Y).sim(10000)
         xy.plot(['tile','joint'])
         xy.cov()
         count = 0
         for i in xy:
             if i[0] < i[1]:
                 count += 1

         1 - count/10000        #Equal to the probability that X > Y
```

```
Out[10]: 0.4766
```

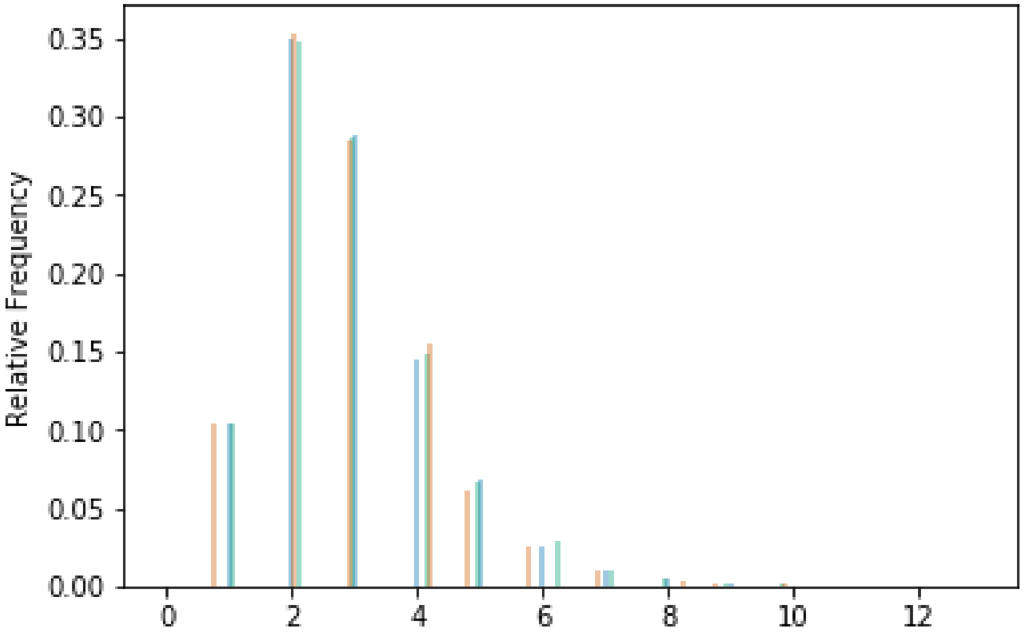


```
In [11]: #1c)
         P = BoxModel([1, 0], probs=[.45, .55], size=16)
         x = count_streak_of_success
         X = RV(P, x)
         z = max_streak
         Z = RV(P, z)
         xz = (X & Z).sim(10000)
         xz.plot(['tile','joint'])
         xz.cov()

         #The probability that X > Z is 0 because X can only equal Z at most in cases where X
```

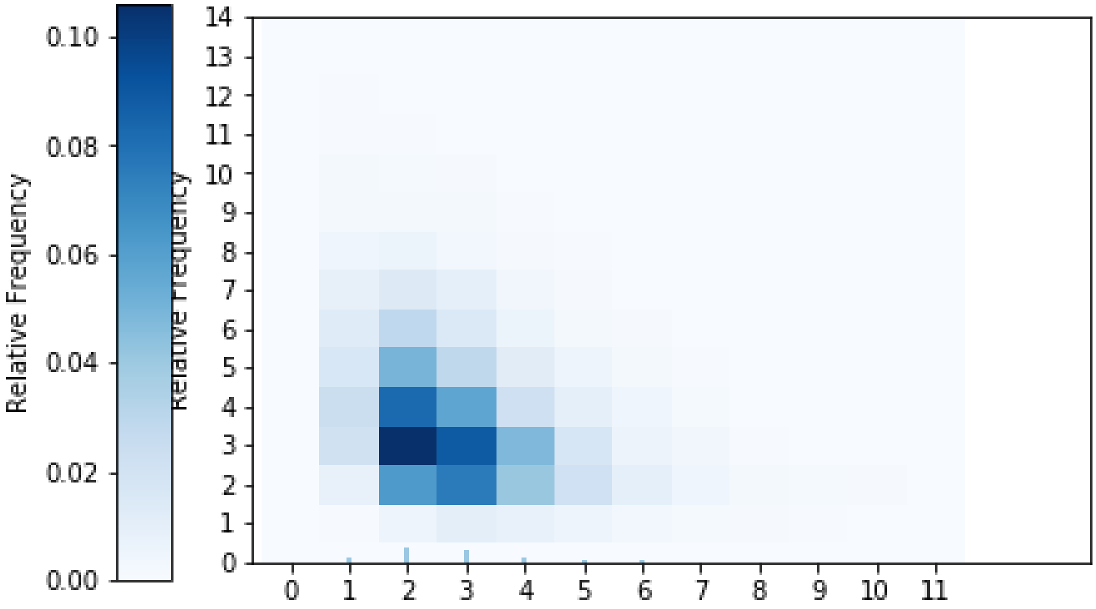
```
Out[11]: 0.607002650265026598
```



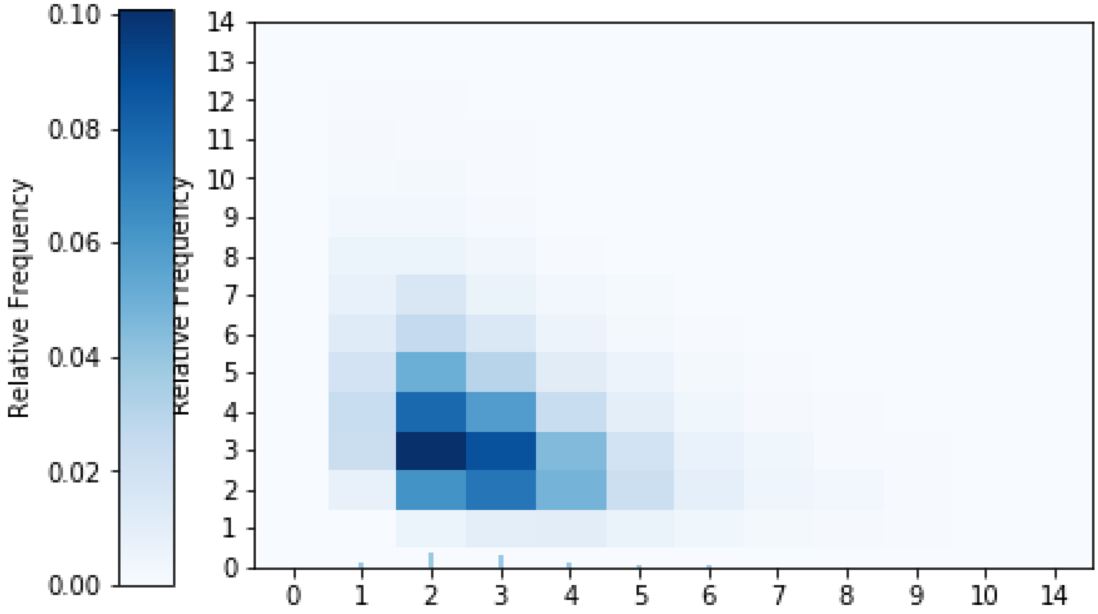
```
In [12]: #Number 2
         P1 = BoxModel([1, 0], probs=[.30, .70], size=16)
         x1 = count_streak_of_success
         X1 = RV(P, x1)
         X1.sim(10000).plot()


         P2 = BoxModel([1, 0], probs=[.50, .50], size=16)
         x2 = count_streak_of_success
         X2 = RV(P, x2)
         X2.sim(10000).plot(jitter = True)


         P3 = BoxModel([1, 0], probs=[.70, .30], size=16)
         x3 = count_streak_of_success
         X3 = RV(P, x3)
         X3.sim(10000).plot(jitter = True)

         #There are multiple graphs overlayed but it is hard to see unless you zoom in.
```



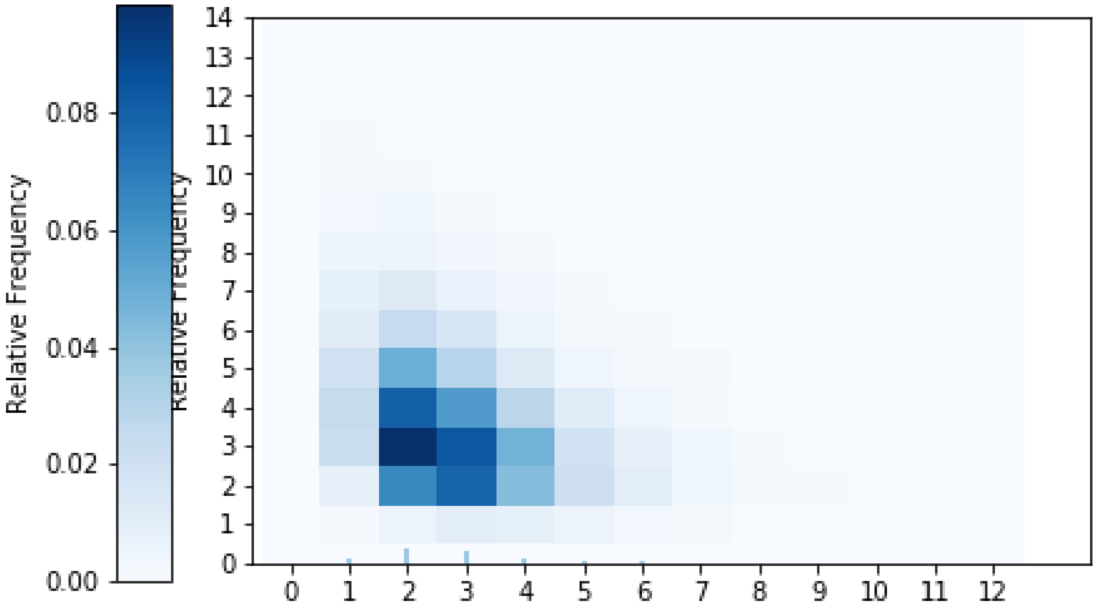
```
In [13]: #Number 2 - Covariance with P(Success)  = .30

         P1 = BoxModel([1, 0], probs=[.30, .70], size=16)
         x1 = count_streak_of_success
         X1 = RV(P, x1)
         X1.sim(10000).plot()
         y = count_streak_of_failure
         Y = RV(P, y)
         x1y = (X1 & Y).sim(10000)
         x1y.plot(['tile','joint'])

         for i in x1y:
             if i[0] < i[1]:
                 count += 1

         X1.sim(10000).mean(), X1.sim(10000).var(), 1 - count/10000, x1y.cov()

Out[13]: (2.895, 1.8394519900000004, -0.05739999999999896, -0.63711507150715063)
```



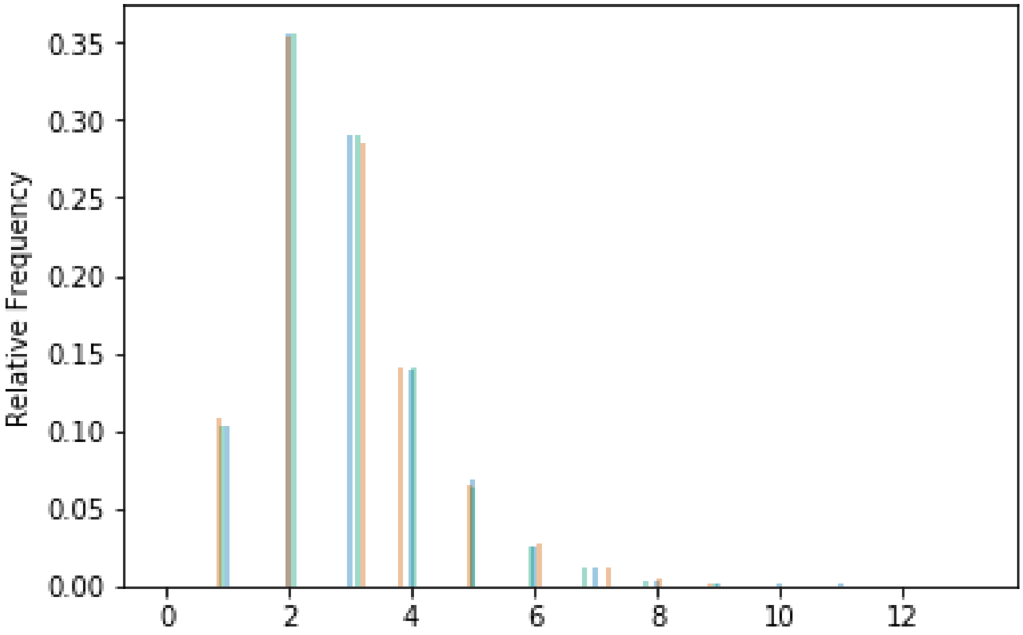
```
In [14]: #Number 2 - Covariance with P(Success)  = .50

         P1 = BoxModel([1, 0], probs=[.50, .50], size=16)
         x1 = count_streak_of_success
         X1 = RV(P, x1)
         X1.sim(10000).plot()
         y = count_streak_of_failure
         Y = RV(P, y)
         x1y = (X1 & Y).sim(10000)
         x1y.plot(['tile','joint'])

         for i in x1y:
             if i[0] < i[1]:
                 count += 1

         X1.sim(10000).mean(), X1.sim(10000).var(), 1 - count/10000, x1y.cov()

Out[14]: (2.897, 1.8779267899999998, -0.5828, -0.68454853485348477)
```



```
In [15]: #Number 2 - Covariance with P(Success)  = .70

         P1 = BoxModel([1, 0], probs=[.70, .30], size=16)
         x1 = count_streak_of_success
         X1 = RV(P, x1)
         X1.sim(10000).plot()
         y = count_streak_of_failure
         Y = RV(P, y)
         x1y = (X1 & Y).sim(10000)
         x1y.plot(['tile','joint'])

         for i in x1y:
             if i[0] < i[1]:
                 count += 1

         X1.sim(10000).mean(), X1.sim(10000).var(), 1 - count/10000, x1y.cov()

Out[15]: (2.8859, 1.8253958399999999, -1.1027, -0.6260349734973496)
```

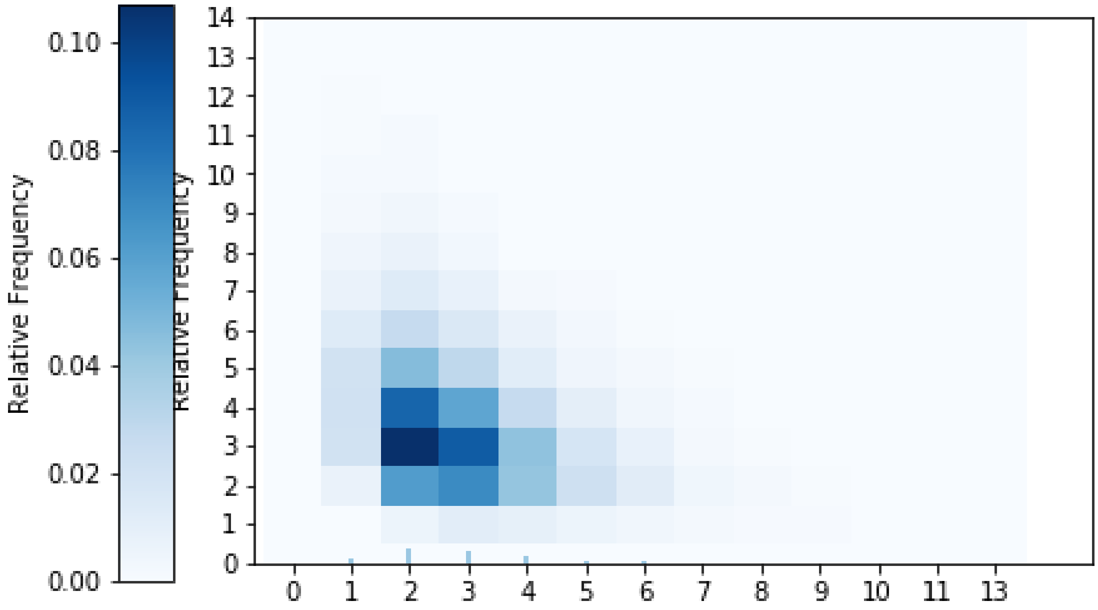


```
In [16]: #Number 3
         P = BoxModel([1, 0], probs=[.45, .55], size=16)
         x = count_streak_of_success
         X = RV(P, x)
         X.sim(10000).plot()


         Q = BoxModel([1, 0], probs=[.45, .55], size=160)
         y = count_streak_of_success
         Y = RV(P, y)
         Y.sim(10000).plot(jitter = True)


         R = BoxModel([1, 0], probs=[.45, .55], size=1600)
         z = count_streak_of_success
         Z = RV(P, z)
         Z.sim(10000).plot(jitter = True)
```

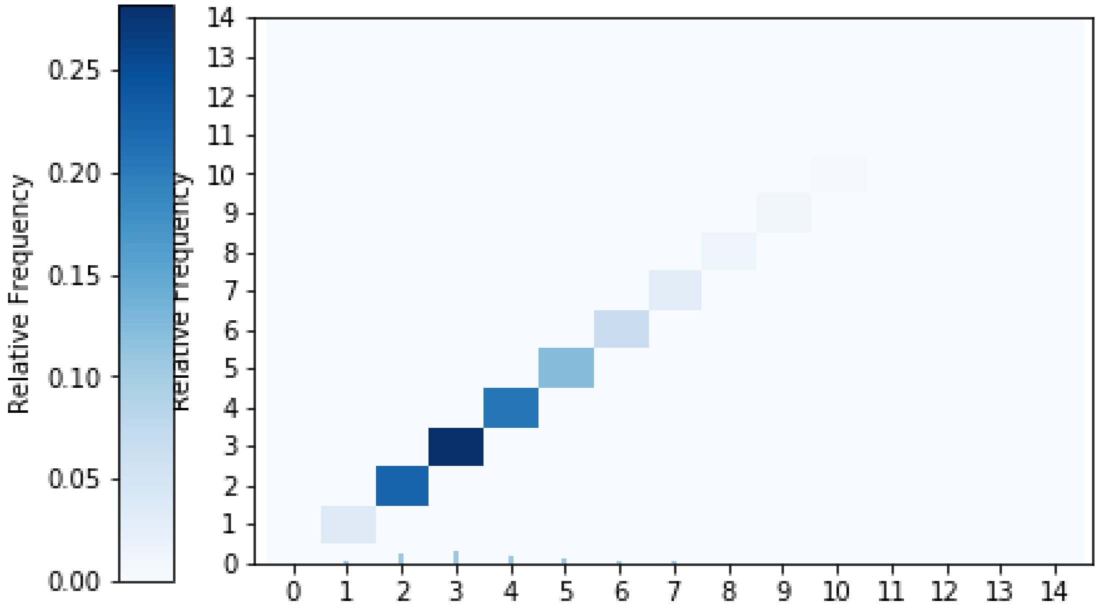


```
In [17]: #Number 3 - Covariance with size = 16

         P = BoxModel([1, 0], probs=[.45, .55], size=16)
         x = count_streak_of_success
         X = RV(P, x)
         X.sim(10000).plot()
         y = count_streak_of_failure
         Y = RV(P, y)
         x1y = (X & Y).sim(10000)
         x1y.plot(['tile','joint'])

         for i in x1y:
             if i[0] < i[1]:
                 count += 1

         X.sim(10000).mean(), X.sim(10000).var(), 1 - count/10000, x1y.cov()
```

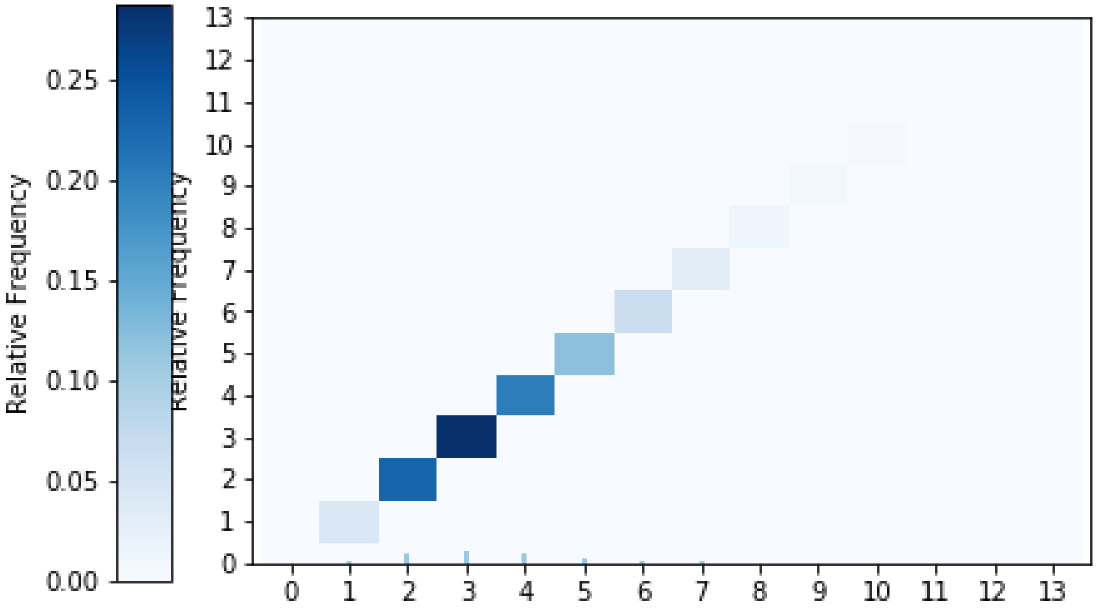
```
Out[17]: (2.8866, 1.81457024, -1.6353, -0.68238275827582717)
```



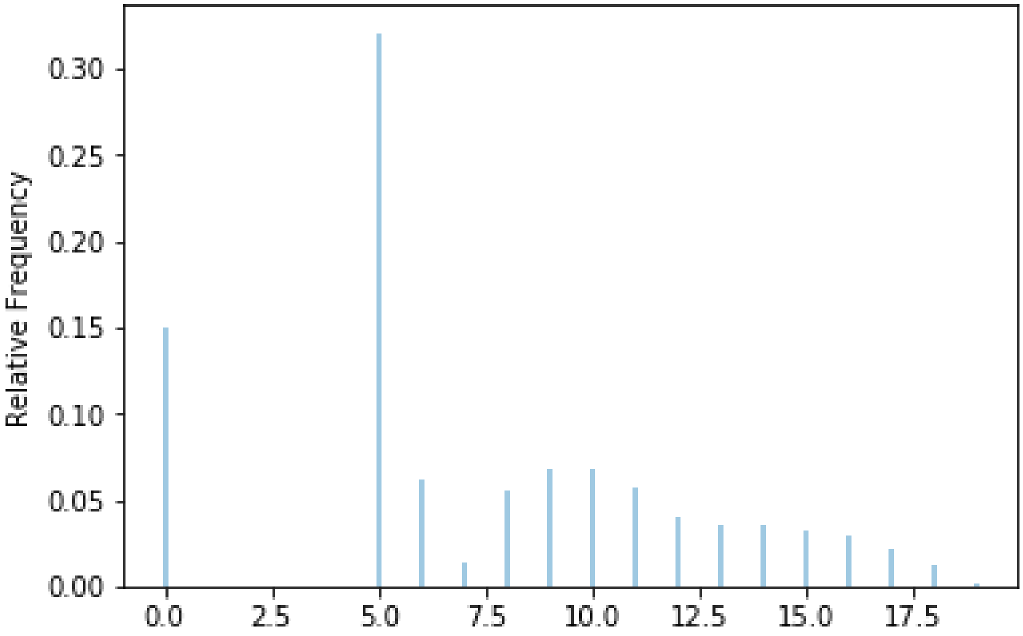
```
In [18]: #Number 3 - Covariance with size = 160

         Q = BoxModel([1, 0], probs=[.45, .55], size=160)
         x = count_streak_of_success
         X = RV(P, y)
         X.sim(10000).plot()
         y = count_streak_of_failure
         Y = RV(P, y)
         x1y = (X & Y).sim(10000)
         x1y.plot(['tile','joint'])

         for i in x1y:
             if i[0] < i[1]:
                 count += 1

         X.sim(10000).mean(), X.sim(10000).var(), 1 - count/10000, x1y.cov()

Out[18]: (3.6061, 2.87357751, -1.6353, 2.7956305630563061)
```



```
In [19]: #Number 3 - Covariance with size = 1600

         R = BoxModel([1, 0], probs=[.45, .55], size=1600)
         x = count_streak_of_success
         X = RV(P, y)
         X.sim(10000).plot()
         y = count_streak_of_failure
         Y = RV(P, y)
         x1y = (X & Y).sim(10000)
         x1y.plot(['tile','joint'])

         for i in x1y:
             if i[0] < i[1]:
                 count += 1

         X.sim(10000).mean(), X.sim(10000).var(), 1 - count/10000, x1y.cov()

Out[19]: (3.6266, 2.76439136, -1.6353, 2.7551282228222798)
```

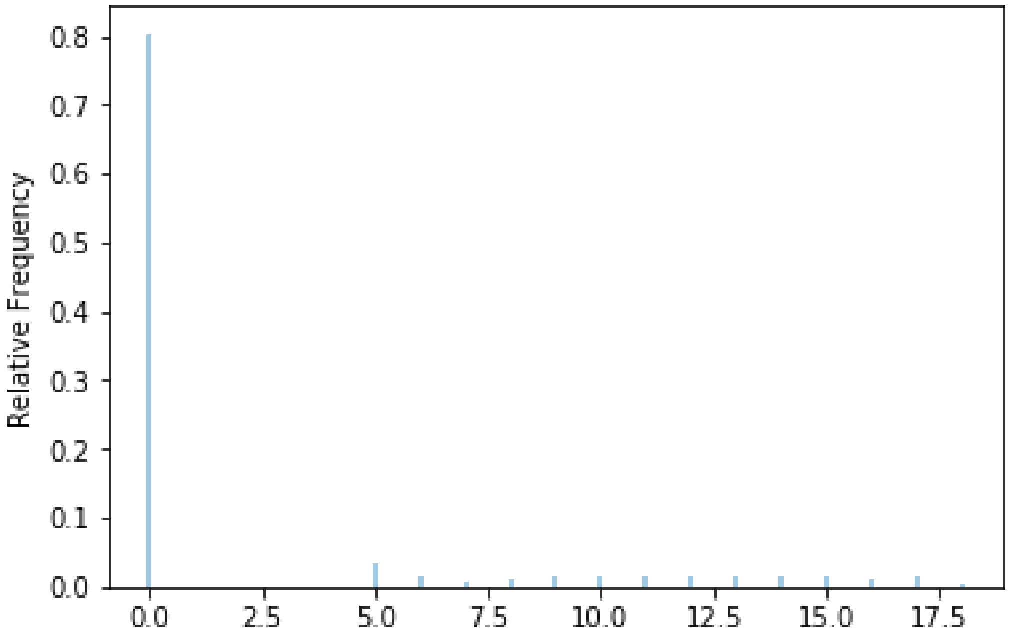


```
In [20]: #4
         #We will now investigate how many games it takes a team to win 5 games in a row with
         #of 80%. If the team never wins 5 games in a row, it gets coded as 0

         r = 5
         def count_until_rth_streak(omega):
             count = 0
             streak = 0
             curmax = 0
             for i, w in enumerate(omega):
                 if w == 0:
                     if curmax < streak:
                         curmax = streak
                         count += 1
                     streak = 0
                     count += 1
                 if w == 1:
                     streak += 1
                     count += 1
                 if streak == 5:
                      return count
             return 0



         P = BoxModel([1, 0], probs=[0.80, 0.20], size=16)
         X = RV(P, count_until_rth_streak)
         x = X.sim(10000)
         x.plot()

         #With a probability of .80 of winning, it takes on average 6-7 games to win 5 in a ro
```



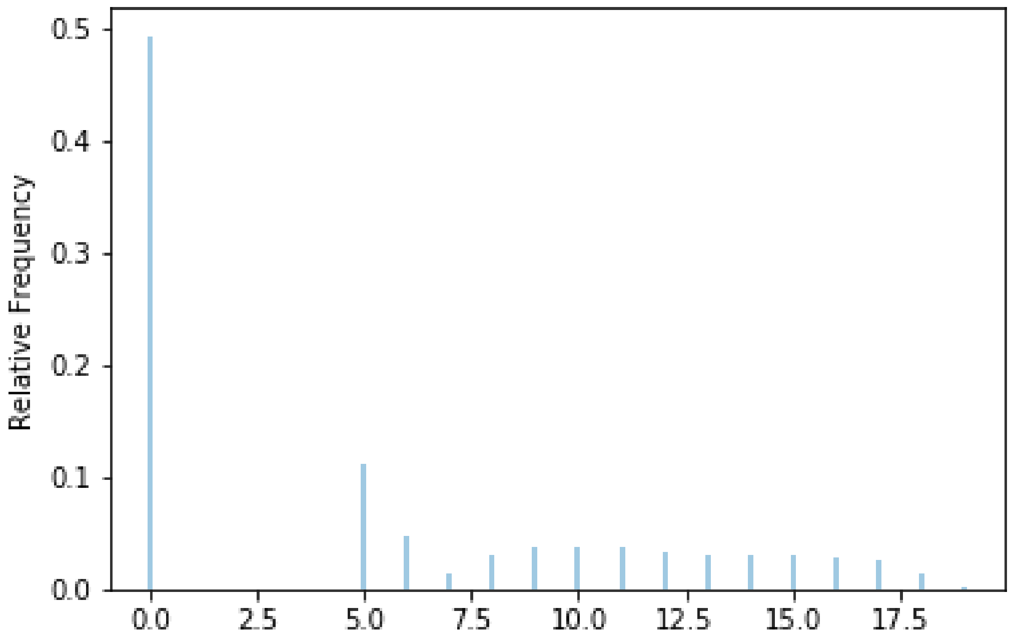
```
In [21]: #4
         #We will now investigate how many games it takes a team to win 5 games in a row with 
         #of 50%. If the team never wins 5 games in a row, it gets coded as 0

         r = 5
         def count_until_rth_streak(omega):
             count = 0
             streak = 0
             curmax = 0
             for i, w in enumerate(omega):
                 if w == 0:
                     if curmax < streak:
                         curmax = streak
                         count += 1
                     streak = 0
                     count += 1
                 if w == 1:
                     streak += 1
                     count += 1
                 if streak == 5:
```

```
            return count
    return 0


P = BoxModel([1, 0], probs=[0.5, 0.5], size=16)
X = RV(P, count_until_rth_streak)
x = X.sim(10000)
x.plot()


#With a probability of .5 of winning, it takes on average 10-12 games to win 5 in a r
#that more often than not, the team never reaches a win streak of 5 games
```



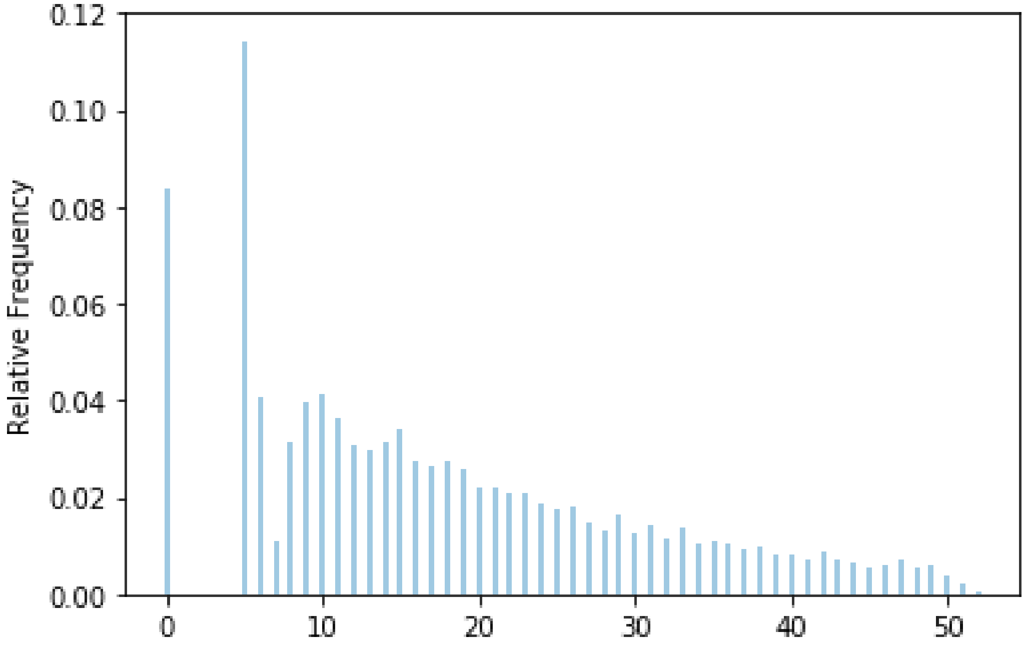
```
In [22]: #4
         #We will now investigate how many games it takes a team to win 5 games in a row with 
         #of 65%. If the team never wins 5 games in a row, it gets coded as 0

         r = 5
         def count_until_rth_streak(omega):
             count = 0
             streak = 0
             curmax = 0
             for i, w in enumerate(omega):
                 if w == 0:
                     if curmax < streak:
```

```
                curmax = streak
                count += 1
            streak = 0
            count += 1
        if w == 1:
            streak += 1
            count += 1
        if streak == 5:
             return count
    return 0


P = BoxModel([1, 0], probs=[0.65, 0.35], size=16)
X = RV(P, count_until_rth_streak)
x = X.sim(10000)
x.plot()


#With a probability of .65 of winning, it takes on average 8-9 games to win 5 in a ro
#that more often than not, the team never reaches a win streak of 5 games but a littl
#in the distribution of win streaks than in the previous graph
```



```
In [23]: #4
         #We will now investigate how many games it takes a team to win 5 games in a row out o
         #rather than 16 with a win probability of 65%. If the team never wins 5 games in a ro
```

```
r = 5
def count_until_rth_streak(omega):
    count = 0
    streak = 0
    curmax = 0
    for i, w in enumerate(omega):
        if w == 0:
            if curmax < streak:
                curmax = streak
                count += 1
            streak = 0
            count += 1
        if w == 1:
            streak += 1
            count += 1
        if streak == 5:
             return count
    return 0



P = BoxModel([1, 0], probs=[0.65, 0.35], size=48)
X = RV(P, count_until_rth_streak)
x = X.sim(10000)
x.plot()


#With a probability of .65 of winning and a total of 48 games, it takes on average 25
#We see that in the simulation a team will win 5 games in a row more times than not.
```


Relative Frequency
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0
10
20
30
40
50