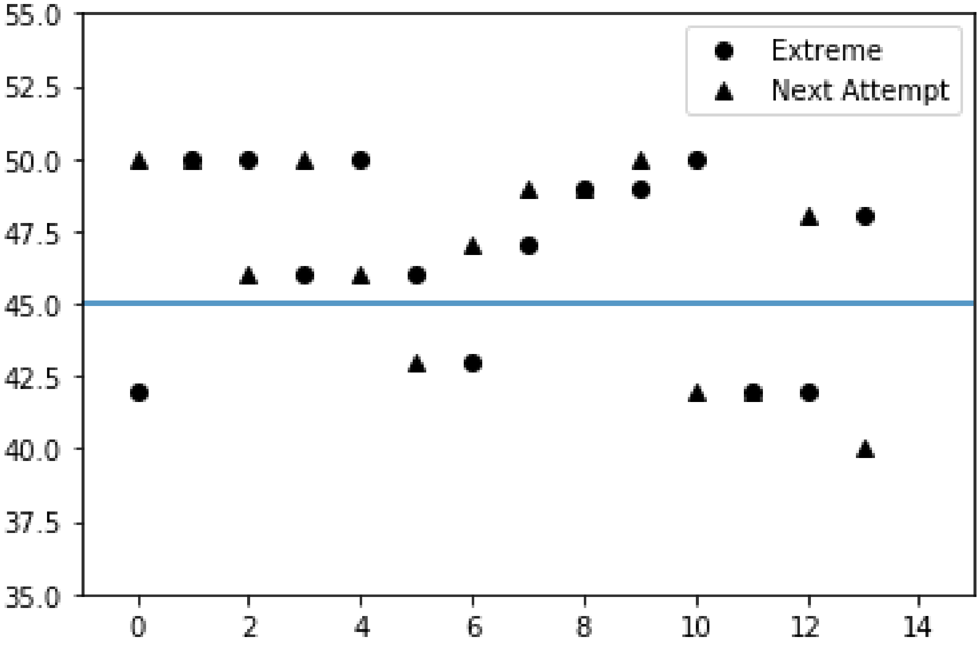
Week 5

August 5, 2018

```
In [80]: # Alla Topp
         # Finger Exercise p.
         # Sally`s golf problem

         import random
         import pylab

         def swing(numStrokes):
             return random.choice([40,41,42,43,45,46,47,48,49,50])

         def regressToMean(numStrokesPerTrial, numAttempts):
             #
             strokes = []
             for t in range(numAttempts):
                 strokes.append(swing(numStrokesPerTrial))
             # Find trials with extreme results and for each the next trial
             extremes, nextAttempt = [], []
             for i in range (len(strokes)-1):
                 if strokes[i] <= 45 or strokes[i] >= 45:
                     extremes.append(strokes[i])
                     nextAttempt.append(strokes[i+1])
             pylab.plot(range(len(extremes)), extremes, 'ko', label = 'Extreme')
             pylab.plot(range(len(nextAttempt)), nextAttempt, 'k^', label = 'Next Attempt')
             pylab.axhline(45)
             pylab.ylim(35,55)
             pylab.xlim(-1, len(extremes)+1)
             pylab.legend(loc = 'best')

         regressToMean(45, 15)
```



```
In [78]: # I disagree with the partner, that Sally will stoke 50 times to fill the next 9 hole
         # because he is assuming the gambler`s fallacy.
         # A better prediction would be assuming that she would be closed to 45, e.g 45
         # because the average of filling 1 hole is 5 strokes and with the first trial(first 9
         # Sally stroke with avarage of 4.444 , next time it will be less extream
         # based on the regression to the mean theory, so she will stroke with an average of 5
```

Week 5, LAB ex.2

August 5, 2018

```
In [1]: # Finger exercise, p. 2 of the textbook
        # Alla Topp
        # Calculate a probability of rolling two 3`s in k(2 to100)rolls of a fair die
```

```
In [16]: import pylab
         import math

         for k in range(2, 101): # count the number from 2 to 100 with k value
             print(k)

             p = 1/6 # probability of getting one

             # formula that calculate two 3`s in k rollings
             NK = math.factorial(k)/(math.factorial(2)*math.factorial(k-2))

             print(NK*(p**2)*((1-p)**(k-2)))
             print() # print an empty line
```

```
2
0.027777777777777776

3
0.06944444444444445

4
0.11574074074074076

5
0.1607510288065844

6
0.20093878600823048

7
0.2344285836762689

8
0.2604762040847432
```

```
9
0.2790816472336535

10
0.2907100492017224

11
0.2960935686313839

12
0.2960935686313839

13
0.2916073024399993

14
0.2835070995944438

15
0.27260298037927294

16
0.259621886075498

17
0.24519844796019263

18
0.2298735449626806

19
0.21409790952406527

20
0.19823880511487524

21
0.18258837313212192

22
0.16737267537111178

23
0.15276077514030045

24
0.13887343194572768
```

```
25
0.12579115212475334

26
0.1135614567792912

27
0.1022053111013621

28
0.091722715090966

29
0.08209749190240782

30
0.07330133205572129

31
0.06529716361285517

32
0.05804192321142682

33
0.051488802848846375

34
0.045589044189082724

35
0.040293347136815545

36
0.035552953356013724

37
0.03132045890886923

38
0.02755040366983867

39
0.024199678899182623

40
0.021227788508054932
```

```
41
0.018596994205774618

42
0.016272369930052794

43
0.014221786727485166

44
0.012415845555741017

45
0.010827772286983447

46
0.00943328646214467

47
0.008210453031866658

48
0.007139524375536225

49
0.006202778269526153

50
0.005384356136741453

51
0.004670104812479831

52
0.004047424170815854

53
0.003505122239432031

54
0.0030332788610469494

55
0.0026231185119116704

56
0.0022668925411582336
```

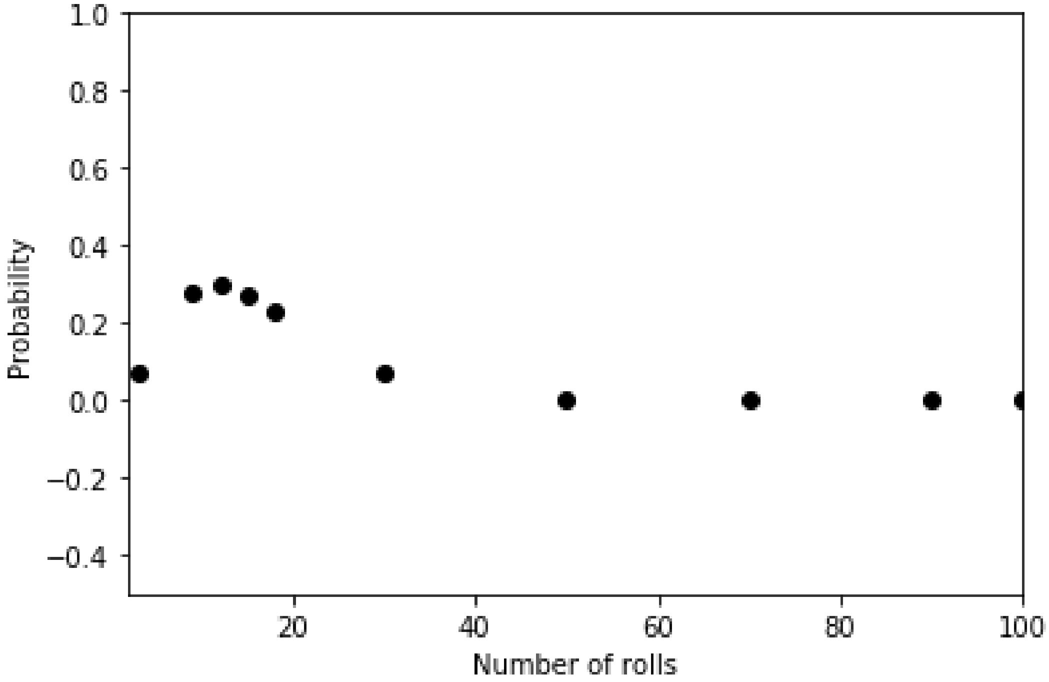
```
57
0.001957770831000293

58
0.00168974268151811

59
0.0014575265820112352

60
0.0012564884327683063

61
0.001082567717498117

62
0.000932211090067823

63
0.0008023128234190281

64
0.0006901615685324973

65
0.0005933928829975175

66
0.0005099470088259917

67
0.000438031405017198

68
0.0003760875699642609

69
0.00032276172049171644

70
0.00027687892689240385

71
0.00023742033586184388

72
0.00020350314502443764
```

```
73
0.00017436302331906044

74
0.00014933870052789895

75
0.00012785847647936557

76
0.00010942842581567323

77
9.362209764229821e-05

78
8.007153087828138e-05

79
6.845942575091157e-05

80
5.8512329701633824e-05

81
4.9994712086839035e-05

82
4.2703816574175005e-05

83
3.646519316519059e-05

84
3.112882343369929e-05

85
2.65657629705265e-05

86
2.2665234280409516e-05

87
1.9332111592114e-05

88
1.64847463188569e-05
```

```
89
1.4053088336956552e-05

90
1.1977063923542518e-05

91
1.0205176189535292e-05

92
8.693298235530064e-06

93
7.403633112676703e-06

94
6.303818048837048e-06

95
5.366153356984942e-06

96
4.566939027221228e-06

97
3.885904260004027e-06

98
3.3057171656284263e-06

99
2.8115635687045894e-06

100
2.3907853475379163e-06
```

```
In [24]: pylab.plot(k, NK, 'ko')
         pylab.plot(3,0.06944444444444445, 'ko')
         pylab.plot(9,0.2790816472336535, 'ko')
         pylab.plot(12, 0.2960935686313839, 'ko')
         pylab.plot(15, 0.27260298037927294, 'ko')
         pylab.plot(18, 0.2298735449626806, 'ko')
         pylab.plot(30, 0.07330133205572129, 'ko')
         pylab.plot(50, 0.005384356136741453, 'ko')
         pylab.plot(70, 0.00027687892689240385, 'ko')
         pylab.plot(90, 1.1977063923542518e-05, 'ko')
         pylab.plot(100, 2.3907853475379163e-06, 'ko')
         pylab.axis([2, 100, -.5, 1])
         pylab.xlabel('Number of rolls')
         pylab.ylabel('Probability')
```

```
Out[24]: Text(0,0.5,'Probability')
```



```
In [ ]: # We can see in the results when we roll the dice from 2 to 100 times,
        # that probability is growing until it hits 12th roll,
        # then probability start decresing until the 100th roll.
        # Unfortunately mu graph does not show the line, which would show the results
        # visually, but I entred number manually from the result to show
        # how probability is acting on the graph.
```