1-Code_for_Figures_2_to_4

November 2, 2018

1 Generating Figures 2,3,4 w/ Python & Stata

1.1 0 Preliminaries

1.1.1 0.1 Import third Party code

```
In [1]: import time
        import os        #for benchmarking
        import ipystata  #allows us to work with Stata
        import importlib #in case we need to update user-defined code e.g. importlib.reload(diff
        import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
```

Check the current directory

```
In [3]: current_dir=os.getcwd()
        print(current_dir)
```

```
C:\Users\Miller\Dropbox\josh\work\projects\HotHand-Surprised\Sandbox-Theory
```

1.1.2 0.2 Print out .py python files that exist in current directory

```
In [11]: for file in os.listdir(current_dir):
             if file[-3:]=='.py':
                 print(file)
```

```
DifferenceInProportions.py
diffP.py
grayscale.py
prop.py
prop_old.py
```

1.1.3 0.3 Import the necessary Python code

```
In [5]: import diffP      #The programs live here
        importlib.reload(diffP)
        #help(prop)
        print('Locked and loaded')
```

```
Locked and loaded
```

1.2 1. Generate the dictionary of sequence types and their frequency:

1.2.1 1.1 Here is an explanation of the objects in diffP.py (can be skipped)

B below is an $(n+1)^3$ matrix of dictionaries $B[m][h][s]$, where: + n is the number of shots, and k is the length of streaks we are interested in
+ $m = \min\{l_m, k\}$ where l_m is the length of the miss streak to the left + $h = \min\{l_h, k\}$ where l_h is the length of the hit streak to the left + s is the number of remaining shots to be take

The on element of the matrix $B[m][h][s]$ is the dictionary of elements

$$B[m][h][s] := \{ \quad (n_{M|kM}, n_{H|kM}, n_{M|kH}, n_{H|kH}) : \gamma \quad \}_{n_{M|kM}, n_{H|kM}, n_{M|kH}, n_{H|kH}, \gamma}$$

- The key $(n_{M|kM}, n_{H|kM}, n_{M|kH}, n_{H|kH})$ is a tuple consisting of the number of misses after k missess, the number of hits after k misses, etc.
- The value γ is the probability of a length n sequence with those key attributes, where at that trial in the sequence there are s shots remaining, the current streak of misses and hits (up to k) is (m, h), note if $m > 0$ then $h = 0$ and vice-versa.

If $k = 1$ and $n = 2$, there are 4 sequences: + $B[0][0][2] = \{(1,0,0,0) : .25, (0,1,0,0) : .25, (0,0,1,0) : .25, (0,0,0,1) : .25\}$

If $k = 1$ and $n = 3$, there are 8 sequences, but some sequences have identical keys, e.g. 010 or 101 have key $(0,1,1,0)$, so + $B[0][0][3] = \{(2,0,0,0) : .125, (0,1,1,0) : .25, \dots\}$

1.3 2 Create the data and graph for Figures 2, 3, 4

1.4 2.1 Create data

For each k, p, we (i) generate the dictionary that associates each count tuple with it's probability for all n, (ii) run the code to compute the expected difference as a function of n, (iii) write the csv.

Note: - for $k = 1$ the expected difference doesn't depend on rate of success (p or fgs_1 below), as was proven in Appendix A.4, theorem 4 - for $k = 2$ we consider 3 probabilites $p = .25, .5, .75$ to illustrate in Figure 4 - for $k = 3$ we consider the range of shooting percentages in the task for adjustments in Figure 2, and p=.25,.5,.75$ to illustrate in Figure 4

```
In [13]: #It's symmetric, but we'll do both
         #Note: did k=1, k=2 already,
         max_number_of_shots = 100
         max_streak_length = 3
         #It's symmetric, but we'll do both
         fgs_1 = [50]
```

```
        fgs_2 = [25, 50, 75]
        fgs_3 = [25, 32, 34, 35, 36, 39, 40, 41, 42, 44, 45, 46, 48, 50, 53, 54, 56, 57, 58, 59
        fgs = [None, fgs_1, fgs_2, fgs_3]
        #B=[None]

        for k in range(1,max_streak_length+1):
            for fg in fgs[k]:
                t0 = time.time()
                probability_of_hit = fg/100

                N = max_number_of_shots
                k = streak_length
                p = probability_of_hit
                B = diffP.outcome_and_frequency_dictionary(N,k,p)
                data= [(p,k,n,diffP.expected_difference(n,k,B)) for n in range(2*k+1,max_number
                labels = ['prob_hit', 'streak_length', 'nshots', 'expected_difference']
                expectdiff_df = pd.DataFrame.from_records(data, columns=labels)
                filename = 'Expected_diff' + '_' + str(k) + '_' + str(fg)' + '.csv'
                expectdiff_df.to_csv(filename, sep=',')
                del B
                del data
                t1 = time.time()
                total = t1-t0
                print('time in seconds=',total)

        print('Dictionary Ready')
```

```
time in seconds= 204.59098505973816
time in seconds= 205.78911900520325
time in seconds= 265.5576286315918
time in seconds= 251.3963007926941
time in seconds= 214.3576533794403
time in seconds= 206.26020002365112
time in seconds= 205.64297938346863
time in seconds= 208.6247878074646
time in seconds= 209.11938905715942
time in seconds= 206.78878211975098
time in seconds= 212.72659492492676
time in seconds= 207.26458477973938
time in seconds= 203.05377578735352
time in seconds= 209.5119891166687
time in seconds= 208.84478735923767
time in seconds= 204.3961799144745
time in seconds= 207.92779970169067
time in seconds= 211.08779001235962
time in seconds= 203.25337767601013
```

```
time in seconds= 204.31258058547974
time in seconds= 205.54258179664612
time in seconds= 206.9627833366394
Dictionary Ready
```

1.4.1 2.1.1 Make a bias adjustment file that is used in "Analysis_Surprised.do" to generate Figure 2

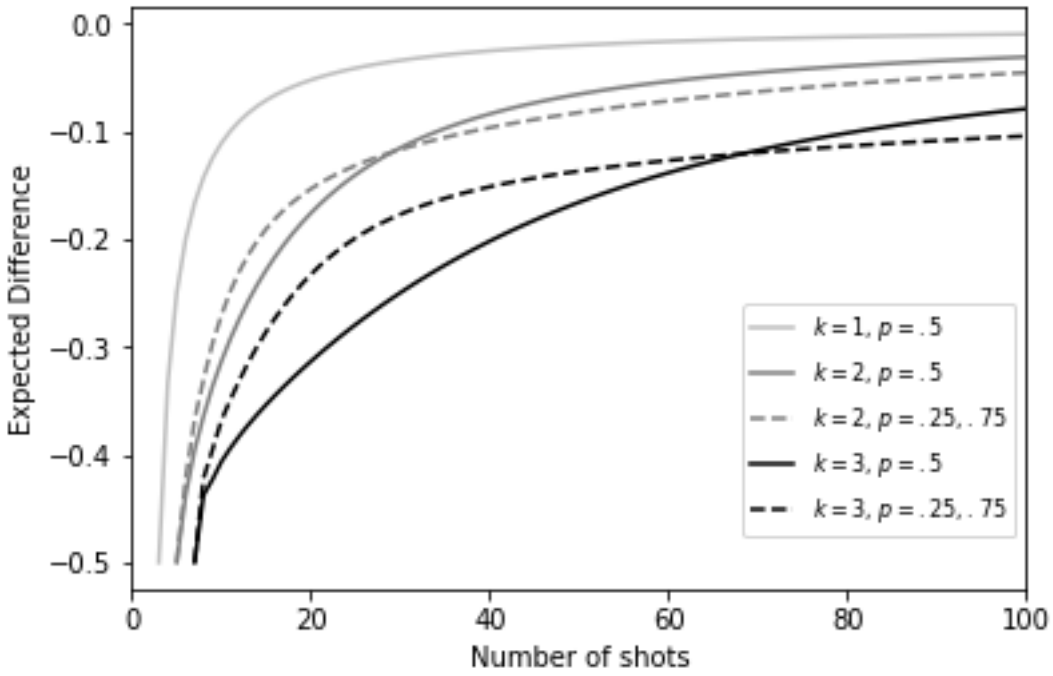
In [ ]:
```
%%stata

        clear
        use "..\0-RAWDATA\GilovichValloneTversky--CognitivePsychology--1985_CornellData.dta"

        collapse (count) nshots=make (mean) fgp = make (sum) nhits=make , by(sid)

        gen bias =.
        forvalues i = 1(1)26 {
                local nhits = nhits[`i']
                local nshots = nshots[`i']
                preserve

                        clear
                        import delimited Expected_diff_3_`nhits'.csv
                        mkmat nshots expected_difference, matrix(bias3)
                        matrix bias3=J(6,2,.)\bias3
                        local bias = bias3[`nshots',2]
                restore

                replace bias = `bias' in `i'
        }
        save ""..\3-Analysis_and_Fig2\biasGVT-0-0.dta ""
```

1.5 ---> To Generate Figure 2, go to directory "3-Analysis_and_Fig2"

1.6 2.2 Generate Figure 4

In [24]:
```
#plt.figure(figsize=(8, 6),facecolor="white")
import csv
#don't include gray borders
plt.figure(facecolor="white")
#plt.style.use('grayscale')

#don't incude negative x-axis
plt.xlim(0, 100)

#gray intensity for descrimination on my screen
g1 = .75
```

```
g2 = .5
g3 = 0
tableau3=[(g1,g1,g1),(g2,g2,g2),(g3,g3,g3)]
#tableau4=[(g-i/3*g,g-i/3*g,g-i/3*g) for i in range(4)]

max_number_of_shots = 100
max_streak_length = 3
fgs_1 = [50]
fgs_2 = [50, 25]  #note by symmetry this includes fg =75
fgs_3 = [50, 25]
fgs = [None, fgs_1, fgs_2, fgs_3]
#B=[None]

for k in range(1,max_streak_length+1):
    for fg in fgs[k]:
        x = []
        y = []
        filename = 'Expected_diff' + '_' + str(k) + '_' + str(fg) + '.csv'
        with open(filename,'r') as csvfile:
            plots = csv.reader(csvfile, delimiter=',')
            next(plots, None)
            # take the 3rd and 4th column from each row
            for row in plots:
                x.append(int(row[3]))
                y.append(float(row[4]))

            if fg == 50:
                plt.plot(x,y,color=tableau3[k-1] ,label='$k=$'+str(k)+', $p=.5$')
            else:
                plt.plot(x,y,'k--',color=tableau3[k-1] ,label='$k=$'+str(k)+', $p=.25,.


#plot text
#plt.text(45, .645, '$p=.75$')
#plt.text(45, .38, '$p=.5$')
#plt.text(45, .195, '$p=.25$')

#plot axis labels
plt.ylabel('Expected Difference')
plt.xlabel('Number of shots')

#plot legend
plt.legend(bbox_to_anchor=(1, .51), loc=1,prop={'size': 8})

#save figure
plt.savefig("ExpectedDiff.pdf", bbox_inches="tight");

#plt.axis([0, 6, 0, 20])
```



1.7 2.3 Generate data used for Figure 3, and create Figure 3

note: While the iterative procedure we used above is a lot more efficient to implement for calculated expected differences than formula we derived in the previous version of our paper (see Miller & Sanjurjo, 2015) that was based on the joint distributions of the number of runs of each length, it is not more efficient for the case of histograms conditional on the number of hits, e.g. as done in Figure 3 with 50 hits out of 100 shots. In that case the approach used in the formula we derived is more efficient. Nevertheless, to maintain some uniformity in approach, the recursive dictionary-based approach is extended to this case.

1.7.1 2.3.1 Create the dictionary of tuple-probability pairs for Figure 3

This section of code is not efficient and took 1.5 hours on my machine.

note: in this case the tuple has an additional entry for the total number of hits, and this extra entry demands a lot more memory. To adjust for this, the method "outcome_and_frequency_dictionary_totalhits" uses a dictionary of dictionaries, instead of a matrix of dictionaries, which is what was used before. In addition, memory that is no longer needed, is freed up along the way. See diffP for details.

```
In [6]: #This is not efficient with memory
        max_number_of_shots = 100
        streak_length = 3
        probability_of_hit = .5  # it doesn't matter which one we choose, since we will conditio

        N = max_number_of_shots
```

```
        k = streak_length
        p = probability_of_hit

        t0 = time.time()

        B = diffP.outcome_and_frequency_dictionary_totalhits(N,k,p)

        t1 = time.time()
        total = t1-t0
        print('time in seconds=',total)
```

```
Step  4
Step  5
Step  6
Step  7
Step  8
Step  9
Step  10
Step  11
Step  12
Step  13
Step  14
Step  15
Step  16
Step  17
Step  18
Step  19
Step  20
Step  21
Step  22
Step  23
Step  24
Step  25
Step  26
Step  27
Step  28
Step  29
Step  30
Step  31
Step  32
Step  33
Step  34
Step  35
Step  36
Step  37
Step  38
Step  39
Step  40
```

```
Step  41
Step  42
Step  43
Step  44
Step  45
Step  46
Step  47
Step  48
Step  49
Step  50
Step  51
Step  52
Step  53
Step  54
Step  55
Step  56
Step  57
Step  58
Step  59
Step  60
Step  61
Step  62
Step  63
Step  64
Step  65
Step  66
Step  67
Step  68
Step  69
Step  70
Step  71
Step  72
Step  73
Step  74
Step  75
Step  76
Step  77
Step  78
Step  79
Step  80
Step  81
Step  82
Step  83
Step  84
Step  85
Step  86
Step  87
Step  88
```

```
Step  89
Step  90
Step  91
Step  92
Step  93
Step  94
Step  95
Step  96
Step  97
Step  98
Step  99
Step  100
time in seconds= 5678.250269412994
```

1.7.2 2.3.2 Create a histogram for Figure 3

This takes 30 minutes as the number of items in memory is large.

note: - the NameError. There was an error in writing the csv. That code has been deleted. In the following cell we write the histogram datafile - There was an error in the method "histogram_counts_totalhits" which defined the difference incorrectly. That has been fixed in the .py file, but the code was not run again to generate a new csv, so this mistake is fixed later using Stata (see cell below)

```
In [11]: n = 100
         number_of_hits = 50

         t0 = time.time()
         histogram = diffP.histogram_counts_totalhits(n,number_of_hits,streak_length,B)
         t1 = time.time()
         total = t1-t0
         print('time=',total)
         print('Histogram Ready')
```

```
time= 1710.5111014842987
Histogram Ready
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-11-108eb2413c2c> in <module>()
         12 t0 = time.time()
         13 with open('histogram.csv','wb') as f:
    ---> 14     w = csv.writer(f)
         15     w.writerows(histogram.items())
         16
```

```
NameError: name 'csv' is not defined
```

1.7.3 2.3.3 Write csv of histogram data for Figure 3

Write the outcome of the histogram in memory.

```
In [19]: import csv
         t0 = time.time()


         with open('histogram.csv', 'w') as f:
             for key in histogram.keys():
                 f.write("%s,%s\n"%(key,histogram[key]))

         t1 = time.time()
         total = t1-t0
         print('time=',total)
         print('Histogram Written to CSV')
```

```
time= 0.33301568031311035
Histogram Written to CSV
```

1.7.4 2.3.4 Graph the histogram of Figure 3

```
In [3]: %%stata
        clear
        pwd
        cd "C:\Users\Miller\Dropbox\josh\work\projects\HotHand-Surprised\Sandbox-Theory"
        import delimited "histogram_modified.csv", asdouble
        rename v1 diff
        rename v3 count

        * fix output error in Python code (now fixed in Python code, but didn't run it again.)
        replace diff= -diff
        sort diff
        replace diff=diff*100
        histogram diff [fweight = count], discrete width(4) fraction kdensity kdenopts(width(4))
            xtitle("Difference (percentage points)") ytitle(Fraction of sequences) scheme(s1manu

        *graph save g3 "Diffbigbins-100-50-3-v2.gph", replace
        *graph export "Diffbigbins-100-50-3-v2.pdf", as(pdf) replace
```

```
C:\Users\Miller\Dropbox\josh\work\projects\HotHand-Surprised\Sandbox-Theory
```

```
C:\Users\Miller\Dropbox\josh\work\projects\HotHand-Surprised\Sandbox-Theory

(3 vars, 28854 obs)

(28,853 real changes made)

(28,853 real changes made)

(start=-100, width=4)
```