CS 3430: SciComp with Py
Assignment 10
Conditional Probabilities and Independent Events

Vladimir Kulyukin
Department of Computer Science
Utah State University

April 1, 2017

1 Learning Objectives

1. Conditional Probabilities
2. Independent Variables
3. Simulation of Random Sampling
4. Bar Graphs

2 Introduction

In this assignment, we will investigate a fundamental relationship between conditional probabilities and dependent and independent events. We will use this relationship to simulate several random sampling experiments on the purchasing habits of various age groups discussed in Lecture 21.

3 Conditional Probability and Independent Variables

Let `A` and `B` be two events. For example, `A` is the event of a fair coin flipping heads and `B` is the event of the same coin flipping tails. Recall that the conditional probability of `A` given that `B` has occurred is denoted as $P(A|B)$ (this reads "the probability of A given B") and is defined as

$$P(A|B) = \frac{P(A,B)}{P(B)} \tag{1}$$

In equation (1), $P(A,B)$ is the probability of both events `A` and `B` occuring and $P(B)$ is the probability of the event `B`.

So, what makes two variables independent? In terms of conditional probability, it simply means that the occurrence of one does not depend on or cannot be conditioned by the occurrence of the other. In other words, the probability of `A` occurring given that `B` has occurred is the same as the probability of `A` occurring by itself:

$$P(A|B) = P(A) \tag{2}$$

Or, equivalently, the probability of `B` occurring given that `A` has occurred is the same as the probability of `B` occurring by itself:

$$P(B|A) = P(B) \tag{3}$$

Suppose `E` is the probability of a student `X` keeping his eyes on his smartphone, `A` is the probability of the student `X` attending some class `C`. Let $P(E) = 0.75$, i.e., `X` keeps his eyes fixed on his smartphone's screen 75% of the time when he is awake. It is quite probable that he is looking at his smartphone when he is asleep, too, but estimating that probability is rather difficult. If we know that `X` attends approximately 60% of the class `C` lectures, then $P(A) = 0.60$. Finally, we contact professor `Y`, who teaches `C`, to learn from her that the probability of `X` attending her class and keeping his eyes fixed on his smartphone is 45%, i.e., $P(E,A) = 0.45$. Are `E` and `A` independent? Computing the conditional probabilities answers this question:

$$P(E|A) = \frac{P(E,A)}{P(A)} = \frac{0.45}{0.60} = 0.75 = P(E) \tag{4}$$

Since $P(E|A) = P(E) = 0.75$, `E` and `A` are independent. If our probabilities are reliable, then we may be right to conclude, all things being equal, that `X` keeps his eyes on his smartphone regardless of what class he sits in. If we set $P(E) = 0.80$, then, since $P(E|A) \neq P(E)$, we conclude that `E` and `A` are dependent.

4 Simulation of Random Sampling

Let us continue working with the example from Lecture 21 of simulating spending habits of different agent groups. We seed the random number generator with the current time converted to an integer. Then we define the 6-tuple of our age groups, two dictionaries, `peopleInAgeGroup` and `purchasesInAgeGroup`, and two variables - `numOfPurchases` and `numOfPeople`.

```
import time
random.seed(int(time.time()))
## age groups
ageGroups = (20, 30, 40, 50, 60, 70)
## number of people in each age group
peopleInAgeGroup = {20:0, 30:0, 40:0, 50:0, 60:0, 70:0}
## number of purchases in each age group
purchasesInAgeGroup = {20:0, 30:0, 40:0, 50:0, 60:0, 70:0}
## total number of purchases
numOfPurchases = 0
## total number of people
numOfPeople = 1000000
```

We can then define the function `generateAgeDependentSpendingData(nofp)` that takes the parameter `nofp` that specifies the number of people in our simulated random sample:

```
def generateAgeDependentSpendingData(nofp):
    global numOfPurchases, ageGroups, numOfPeople
    numOfPeople = nofp
    numOfPurchases = 0
    resetTables()
    for _ in xrange(numOfPeople):
        ## randomly choose an age group
        ageGroup = random.choice(ageGroups)
        ## the younger you are the less likely you are to buy stuff
        purchaseProbability = float(ageGroup) / 100.0
        ## modify the number of people in ageGroup
        peopleInAgeGroup[ageGroup] += 1
        ## if the purchase probability > random
        if (random.random() < purchaseProbability):
            numOfPurchases += 1
            purchasesInAgeGroup[ageGroup] += 1
```

The function `generateAgeDependentSpendingData(nofp)` generates random samples with age dependent purchasing probabilities: the older a person is, the more that person is likely to spend.

Define the function `generateAgeIndependentSpendingData(nofp, probOfPurchase=0.4)` that generates random samples with age independent purchasing probabilities. The first parameter, `nofp`, specifies the number of people in the random sample and the second parameter, `probOfPurchase`, specifies the probability of a person in any age group making a purchase, which defaults to 40%.

```
def generateAgeIndependentSpendingData(nofp, probOfPurchase=0.4):
   ## your code
   pass
```

Define the function `arePurchaseAndAgeGroupIndependent(ageGroups, probDiff=0.01)`, where `ageGroups` is the age group tuple, i.e., `(20, 30, 40, 50, 60, 70)`, and the second argument is the probability threshold for considering two probabilities equal when their absolute difference is less than or equal to `probDiff`.

```
def arePurchaseAndAgeGroupIndependent(ageGroups, probDiff = 0.01):
   ## your code
   pass
```

Now we define the function `runExperiment` and several command line arguments that control our random sampling experiments. I deliberately commented out the function `saveDataPlot` which we will define in the next section. Uncomment it after you implement it.

```
def runExperiment(ageGroups, nofp=1000, dependent=True,
                  probOfPurchase=0.4, probDiff=0.1):
    if dependent is True:
        generateAgeDependentSpendingData(nofp)
        arePurchaseAndAgeGroupIndependent(ageGroups, probDiff=probDiff)
        #saveDataPlot(nofp, probOfPurchase, dependent)
    else:
        generateAgeIndependentSpendingData(nofp, probOfPurchase=probOfPurchase)
        arePurchaseAndAgeGroupIndependent(ageGroups, probDiff=probDiff)
        #saveDataPlot(nofp, probOfPurchase, dependent)

import argparse
if __name__ == '__main__':
    ap = argparse.ArgumentParser()
    ap.add_argument('-nofp', '--nofp', required = True, help = 'number of people', type=int)
    ap.add_argument('-pp', '--pp', required = True, help = 'probability of purchase', type=float)
    ap.add_argument('-pd', '--pd', required=True, help='probability difference', type=float)
    ap.add_argument('-dp', '--dp', required=True, help='dependency flag 0/1', type=int)
    args = vars(ap.parse_args())
    dependency = args['dp']
    if dependency == 0:
        print('Running independent experiment')
        runExperiment(ageGroups, nofp=args['nofp'], dependent=False, probDiff=args['pd'],
                      probOfPurchase=args['pp'])
    elif dependency == 1:
        print('Running dependent experiment')
        runExperiment(ageGroups, nofp=args['nofp'], dependent=True, probDiff=args['pd'])
    else:
        print('Incorrect value for dependency flag %d' % args['dp'])
```

Below is a trial run of what your implementation saved in `random_sampling.py` should output. Of course, the actual probabilities will be different due to random number generation.

```
$ python random_sampling.py -nofp 3000 -pp 0.4 -pd 0.1 -dp 0
Running independent experiment
Purchase and AgeGroup=20 are independent
P(Purchase)=0.388333; P(Purchase|AgeGroup=20)=0.382845
Purchase and AgeGroup=30 are independent
P(Purchase)=0.388333; P(Purchase|AgeGroup=30)=0.414683
Purchase and AgeGroup=40 are independent
P(Purchase)=0.388333; P(Purchase|AgeGroup=40)=0.350538
Purchase and AgeGroup=50 are independent
P(Purchase)=0.388333; P(Purchase|AgeGroup=50)=0.396078
Purchase and AgeGroup=60 are independent
P(Purchase)=0.388333; P(Purchase|AgeGroup=60)=0.368317
Purchase and AgeGroup=70 are independent
P(Purchase)=0.388333; P(Purchase|AgeGroup=70)=0.412639
```

In the above trial run, since the last command line argument `-dp` is set to 0, the random sampling experiment is run with a generated random sample of 3000 persons with age independent purchasing probabilities.

```
$ python random_sampling.py -nofp 3000 -pp 0.4 -pd 0.1 -dp 0
Running independent experiment
Purchase and AgeGroup=20 are independent
P(Purchase)=0.388333; P(Purchase|AgeGroup=20)=0.382845
Purchase and AgeGroup=30 are independent
P(Purchase)=0.388333; P(Purchase|AgeGroup=30)=0.414683
Purchase and AgeGroup=40 are independent
P(Purchase)=0.388333; P(Purchase|AgeGroup=40)=0.350538
Purchase and AgeGroup=50 are independent
P(Purchase)=0.388333; P(Purchase|AgeGroup=50)=0.396078
Purchase and AgeGroup=60 are independent
P(Purchase)=0.388333; P(Purchase|AgeGroup=60)=0.368317
Purchase and AgeGroup=70 are independent
P(Purchase)=0.388333; P(Purchase|AgeGroup=70)=0.412639
```

Now we run `random_sampling.py` with the command line `-dp` set to 1, which causes a generated random sample of 10,000 persons to have age dependent purchasing probabilities:

```
$ python random_sampling.py -nofp 10000 -pp 0.4 -pd 0.01 -dp 1
Running dependent experiment
Purchase and AgeGroup=20 are dependent
P(Purchase)=0.442900; P(Purchase|AgeGroup=20)=0.204819
Purchase and AgeGroup=30 are dependent
P(Purchase)=0.442900; P(Purchase|AgeGroup=30)=0.279449
Purchase and AgeGroup=40 are dependent
P(Purchase)=0.442900; P(Purchase|AgeGroup=40)=0.395307
Purchase and AgeGroup=50 are dependent
P(Purchase)=0.442900; P(Purchase|AgeGroup=50)=0.496770
Purchase and AgeGroup=60 are dependent
P(Purchase)=0.442900; P(Purchase|AgeGroup=60)=0.592457
Purchase and AgeGroup=70 are dependent
P(Purchase)=0.442900; P(Purchase|AgeGroup=70)=0.695521
```

Note that the value of `-pd` is 0.01 and, at this threshold, `Purchase` and `AgeGroup` are dependent. This may not be true for some age groups when we set the value of `-pd` equal to 0.1, as shown below.

```
$ python random_sampling.py -nofp 10000 -pp 0.4 -pd 0.1 -dp 1
Running dependent experiment
Purchase and AgeGroup=20 are dependent
P(Purchase)=0.443800; P(Purchase|AgeGroup=20)=0.187957
Purchase and AgeGroup=30 are dependent
P(Purchase)=0.443800; P(Purchase|AgeGroup=30)=0.306395
Purchase and AgeGroup=40 are independent
P(Purchase)=0.443800; P(Purchase|AgeGroup=40)=0.399272
Purchase and AgeGroup=50 are independent
P(Purchase)=0.443800; P(Purchase|AgeGroup=50)=0.522424
Purchase and AgeGroup=60 are dependent
P(Purchase)=0.443800; P(Purchase|AgeGroup=60)=0.584010
Purchase and AgeGroup=70 are dependent
P(Purchase)=0.443800; P(Purchase|AgeGroup=70)=0.699501
```

5 Plotting Purchase Probabilities in Bar Graph

In addition to histograms and scatter plots, which you have become closely familiar with through class examples and assignments, bar graphs are another really useful plot type you should be familiar with. The function that plots a bar graph in `matplotlib` is `plt.bar(x, y, barwidth)`.

Write the function `saveDataPlot(nofp, probOfPurchase, dpflag)` that generates a bar graph that plots the age groups against their purchase probabilities. The first argument `nofp` specifies the number of people in the sample, the second argument `probOfPurchase` specifies the probability of purchase for independent generation, e.g., 0.4, and the second argument `dpflag` specifies whether a generated sample has dependent or independent probabilities.

The generated plot is saved in a PNG file. If `dpflag` is `True`, then the name of the generated file should consist of the value of `nofp` and the suffix `_dep`, e.g., `2000_dep.png`. The file should be saved in the current directory. The `matplotlib` function to save plots as images is `plt.savefig(file_path)`, e.g., `plt.savefig('3000_dep.png')`. The generated plot should also have a horizontal red line drawn at the level of `probOfPurchase`.

```
def saveDataPlot(nofp, probOfPurchase, dpflag):
    ## your code
    pass
```

Figures 1 and 2 show two plots that you should use as models. The plot in 1 shows dependent generation while the plot in 2 show independent generation. The zip archive for this assignment contains several other sample plots.

6 What To Submit

The zip for this assignment contains `random_sample.py` with the starter code and several sample plots I generated with my code for various sample sizes with dependent and independent probabilities. You can use those as additional examples. Write the function definitions in `random_sample.py` and submit it via Canvas.

Happy Hacking!

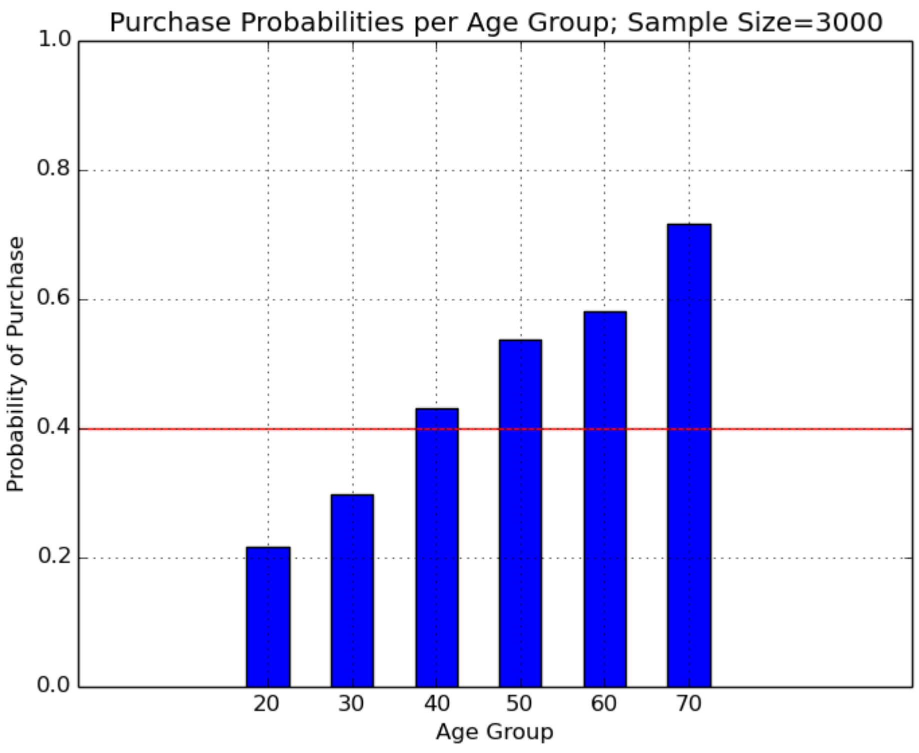

Figure 1: Bar graph of a sample of 3000 persons with dependent purchase probabilities and purchase probability equal to 0.4

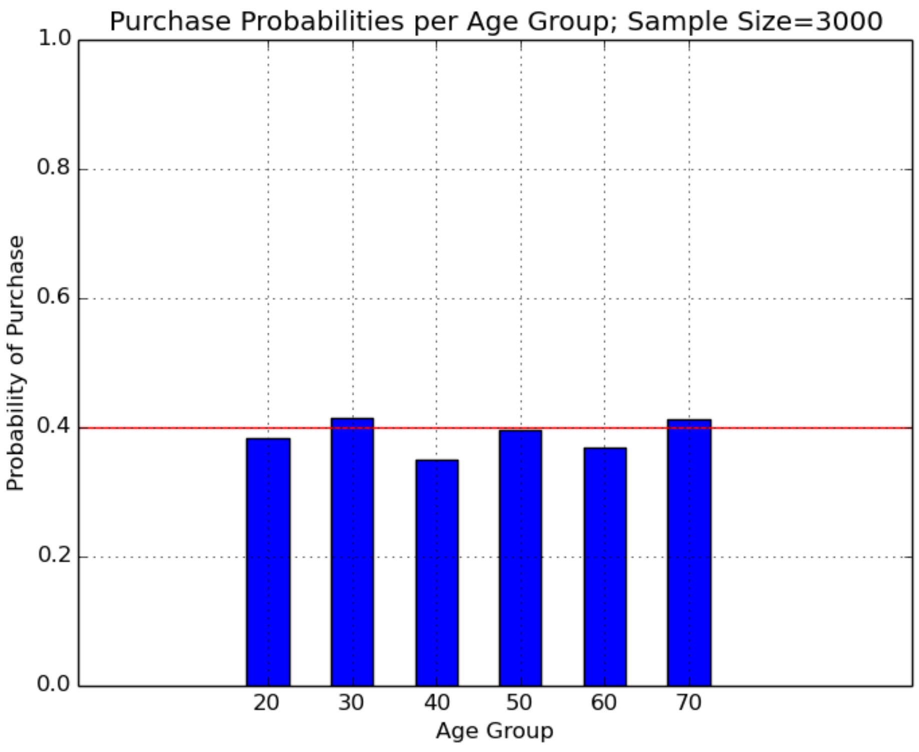

Figure 2: Bar graph of a sample of 3000 persons with independent purchase probabilities and purchase probability equal to 0.4