B551 Assignment 3: Probability

Spring 2015
Due: Sunday October 18, 11:59PM
(You may submit up to 48 hours late for a 10% penalty.)

This assignment will give you a chance to practice probability problems, both on pen-and-paper and in practice.

For this assignment, unlike in past assignments, **you are to work individually, not in a team**. As in past assignments, we've assigned you to a team based on your teamwork preferences.

Academic integrity. You may discuss the assignment with other people at a high level, e.g. discussing general strategies to solve the problem, talking about Python syntax and features, etc. You may also consult printed and/or online references, including books, tutorials, etc., but you must cite these materials (e.g. in source code comments). However, the work and code that you submit must be your own work, which you personally designed and wrote. You may not share written answers or code with any other students, nor may you possess code or solutions written by another student, either in whole or in part, regardless of format.

Part 0: Getting started

We've created a github repo for you, as usual. To get started, clone the github repository:

`git clone https://github.iu.edu/cs-b551/`*your-user-name*`-a3`

Part 1: Written problems

1. Eight parakeets, four green and four blue, land on a telephone wire in random order. What is the probability that no two adjacent parakeets are the same color?

2. A certain CPU is designed with 8 computing cores. There's a 30% probability that any given computing core will have a manufacturing defect that prevents it from functioning correctly. The defects occur randomly and independent of each other.

   (a) What's the probability that a given CPU will have 8 functioning compute cores?

   (b) To prevent losing money from so many defective CPUs, the manufacturer comes up with an ingenious plan. They introduce a family of 3 CPU models: the Great model, the Advanced model, and the Extreme model. In reality, the Great and Advanced models are simply defective versions of the Extreme model. That is, the only difference between these models is the number of functioning compute cores; Great is guaranteed to have at least 1 functioning core, Advanced has at least 4 functioning cores, and the Extreme has 8 functioning cores. If the company tries to make one thousand 8-core CPUs, how many of each model (Great, Advanced, and Extreme) can it expect to make?

   (c) If the Great model costs $50, the Advanced model costs $100, and the Extreme model costs $1000, when the company sells all of the thousand CPUs, what's their expected revenue?

3. Suppose that in some judicial system, an accused person is tried by a 3-judge panel. Suppose that when the accused person is, in fact, guilty, each judge will independently vote guilty with probability 0.7, whereas when the defendant is, in fact, innocent, this probability drops to 0.2. Suppose that 70% of accused people are actually guilty.

   (a) Suppose Judge 1 has voted guilty. What's the probability that the accused person is in fact guilty?

   (b) Suppose all three judges vote guilty. Now what's the probability that the accused person is in fact guilty?

(c) Suppose Judge 1 and Judge 2 have voted innocent. What's the probability that Judge 3 votes guilty?

4. A meteorological observing station in remote Indiana uses a wireless network link to transmit data from a moisture sensor to a central computer. Each day the sensor sends a `yes` message to the computer if it has rained that day, or a `no` message if it has not rained. Unfortunately, the wireless link is noisy, so that 40% of the `yes` messages sent by the sensor are incorrectly received as `no` messages by the computer, and 20% of the `no` messages transmitted by the sensor are incorrectly received as `yes` messages by the computer. Suppose that if it rains one day in this location, the probability of it raining the next day is 65%; if it does not rain one day, the probability of rain the next day is 25%.

   During the four days of operation, the computer receives the following sequence of messages from the sensor:

   ```
   yes yes no yes
   ```

   (a) Draw a Bayes network to model this system in terms of 4 unobserved variables and 4 observed variables. State the conditional independence assumptions that it makes.

   (b) Suppose you know that it did not rain the first day. Use the variable elimination algorithm to estimate the probability that it rained on the fourth day.

Part 2: Programming problem

The following problem requires you to write a program in Python. As in the past, you may import standard Python modules for routines not related to AI, such as basic sorting algorithms and basic data structures like queues. You must write all of the rest of the code yourself. If you have any questions about this policy, please ask us. We recommend using the CS Linux machines (e.g. `burrow.soic.indiana.edu`). You may use another development platform (e.g. Windows), but make sure that your code works on the CS Linux machines before submission. Remember to include a detailed comments section at the top of your code that describes: (1) a description of how you formulated the problem and how your solution works, (2) any problems you faced, assumptions you made, etc., and (3) a brief analysis of how well your program works and how it could be improved in the future.

Zacate is a traditional dice game played in some parts of Latin America. It can be played alone or against two or more players. Here are the rules. At the beginning of each turn, the player rolls a set of 5 dice. He or she inspects the dice, and chooses any subset (including none or all) to roll again. He or she again inspects the dice, and again rerolls any subset. The turn is then over, and the player must assign the outcome to exactly one of the following categories, depending on which 5 dice are showing after the third roll:

- Unos: The player can add the number of dice that show 1 to his or her score.
- Doses: The player can count the number of dice that show 2, multiply by 2, and add to the score.
- Treses: The player can count the number of dice that show 3, multiply by 3, and add to the score.
- Cuatros: The player can count the number of dice that show 4, multiply by 4, and add to the score.
- Cincos: The player can count the number of dice that show 5, multiply by 5, and add to the score.
- Seises: The player can count the number of dice that show 6, multiply by 6, and add to the score.
- Pupusa de queso: If the five dice are either 1, 2, 3, 4, 5 or 2, 3, 4, 5, 6, the player can add 40 points to their score. (Note that for this and all other categories, the order of the dice is not important.)

- Pupusa de frijol: If the four of the five dice are either 1, 2, 3, 4, or 2, 3, 4, 5, or 3, 4, 5, 6, the player can add 30 points to their score.
- Elote: If three of the dice show the same number, and the other two dice are also the same, the player can add 25 points to their score.
- Triple: If three of the dice are the same, the player can add up the values of all five dice and add this to their score.
- Cuádruple: If four of the dice are the same, the player can add up the values of all five dice and add the sum to their score.
- Quíntupulo: If all five dice are the same, the player can add 50 points to their score.
- Tamal: The sum of all five dice, no matter what they are.

Players often have a choice of which category to fill with any particular roll, but **each category may be filled only once per game.** A player can also choose to assign a roll to a category that does not match the requirements, but then nothing is added to their score. After 13 turns, all categories are full, and the game ends. If the player managed to get a score of at least 63 totaled across the first six categories (Unos through Seises), they get a bonus of 35 points added to their score.

Implement a program that plays one-player Zacate as well as possible, i.e. getting the highest possible score. Run your program many times so that it plays many games. What is the average score your program is able to obtain? Your goal is to achieve as high an average score as possible. As with Assignment 2, a small portion of your grade will be based on how well your program works with respect to the rest of the class.

To get you started, we've implemented some skeleton code that should be in your cloned repository. The skeleton code implements a very naive automatic player that simply rolls and rerolls dice blindly and randomly assigns rolls to categories. The main program is called zacate.py, and it runs the game 100 times and computes the mean score. You can run it like this:

```
python zacate.py
```

The automatic player is implemented in ZacateAutoPlayer.py, and this is the file you should modify. It uses a helper module called ZacateState.py to keep track of the scoreboard and to roll the dice. While you should look at ZacateState.py and zacate.py to understand how these classes work, and you may want to modify ZacateState.py and zacate.py for debugging purposes, your final submission should run **using the ZacateState.py and zacate.py files that we supplied without any modifications**, or else we won't be able to grade your submission correctly.

What to turn in

Turn in the files required above by simply putting the finished version (of the code with comments and PDF file for the first question) on GitHub (remember to `add`, `commit`, `push`) — we'll grade whatever version you've put there as of 11:59PM on the due date. To make sure that the latest version of your work has been accepted by GitHub, you can log into the github.iu.edu website and browse the code online.