Inference about inference - exercises

Exercise 1: Tricky Agents

What would happen if Sally knew you were watching her and wanted to deceive you?

a) Complete the code below so that `chooseAction` chooses a misdirection if Sally is deceptive. Then describe and show what happens if you knew Sally was deceptive and chose action “b”.

```
x
```

```
var actionPrior = Categorical({vs: ['a', 'b', 'c'], ps: [1/3, 1/3, 1/3]});
```

```
var foodPrior = Categorical({vs: ['bagel', 'cookie', 'doughnut'], ps: [1/3, 1/3, 1/3]});
```

```

```

```
var vendingMachine = function(state, action) {
```

```
  return (action == 'a' ? categorical({vs: ['bagel', 'cookie', 'doughnut'], ps: [.8, .1, .1]}
```

```
          action == 'b' ? categorical({vs: ['bagel', 'cookie', 'doughnut'], ps: [.1, .8, .1]}
```

```
      action == 'c' ? categorical({vs: ['bagel', 'cookie', 'doughnut'], ps: [.1, .1, .8]}) :
```

```
      'nothing');
```

```

```

```
var chooseAction = function(goal, transition, state, deceive) {
```

```
  return Infer({method: 'enumerate'}, function() {
```

```
    var action = sample(actionPrior);
```

```
    condition(...)
```

```
    return action;

  })

};


var goalPosterior = Infer({method: 'enumerate'}, function() {

  var deceive = flip();

  var goalFood = sample(foodPrior);

  var goal = function(outcome) {return outcome == goalFood};

  var sallyActionDist = chooseAction(goal, vendingMachine, 'state', deceive);

  condition(...)

  return goalFood;

});


viz.auto(goalPosterior);
```

run ▼

b) What happens if you don't know Sally is deceptive and she chooses "b" and then "b". What if she chooses "a" and then "b." Show the models and describe the difference in behavior. Is she deceptive in each case?

Exercise 2: Monty Hall.

Here, we will use the tools of Bayesian inference to explore a classic statistical puzzle – the Monty Hall problem. Here is one statement of the problem:

> *Alice is on a game show and she's given the choice of three doors. Behind one door is a car; behind the others, goats. She picks door 1. The host, Monty, knows what's behind the doors and opens another door, say No. 3, revealing a goat. He then asks Alice if she wants to switch doors. Should she switch?*

Intuitively, it may seem like switching doesn't matter. However, the canonical solution is

that you *should* switch doors. We'll explore (a) the intuition that switching doesn't matter, (b) the canonical solution, and more. This is the starter code you'll be working with:

```
// Here's a function that might be handy: it removes some set of badItems from a list l
// e.g. removeBadItems(['nut', 'cake', 'nut', 'bagel'], ['cake', 'bagel']) => ['nut', 'nut
var removeBadItems = function(l, badItems) {
  return reduce(function(badItem, remainingL) {
    return remove(badItem, remainingL)
  }, l, badItems);
}

var doors = [1,2,3]

var montyRandom = function(aliceDoor, prizeDoor) {
  return Infer({method: 'enumerate'}, function() {
    ...
  };
};

// var montyAvoidBoth = function(aliceDoor, prizeDoor) {
//   return Infer({method: 'enumerate'}, function() {
//     ...
//   };
```

```
// };

// var montyAvoidAlice = function(aliceDoor, prizeDoor) {
//   return Infer({method: 'enumerate'}, function() {
//     ...
//   };
// };

// var montyAvoidPrize = function(aliceDoor, prizeDoor) {
//   return Infer({method: 'enumerate'}, function() {
//     ...
//   };
// };

Infer({method: 'enumerate'}, function() {
  var aliceDoor = ...
  var prizeDoor = ...
  var montyFunction = montyRandom;

  var montyDoorDist = montyFunction(aliceDoor, prizeDoor);
```

```
  factor(...)

  return ...

});
```

run ▼

a) Whether you should switch depends crucially on how you believe Monty chooses doors to pick. First, write the model such that the host *randomly* picks doors (for this, fill in `montyRandom`). In this setting, should Alice switch? Or does it not matter? Hint: it is useful to condition on the exact doors that we discussed in the problem description.

b) Now, fill in `montyAvoidBoth` (make sure you switch your `var montyFunction = ...` alias to use `montyAvoidBoth`). Here, Monty randomly picks a door that is *neither* the prize door *nor* Alice's door. For both-avoiding Monty, you'll find that Alice *should* switch. This is unintuitive – we know that Monty picked door 3, so why should the process he used to arrive at this choice matter? By hand, compute the probability table for $P(\text{Prize} \mid \text{Alice picks door 1}, \text{Monty picks door 3}, \text{Door 3 is not the prize})$ $\mathrm{P}(\text{Prize} \mid \text{Alice picks door 1,Monty picks door 3,Door 3 is not the prize})$ under both `montyRandom` and `montyAvoidBoth`. Your tables should look like:

Alice's door	Prize door	Monty's Door	P(Alice, Prize, Monty)
1	1	1	...
1	1	2	...
...	...	...	...

Using these tables, explain why Alice should switch for both-avoiding Monty but why switching doesn't matter for random Monty. Hint: you will want to compare particular *rows* of these tables.

c) Fill in `montyAvoidAlice`. Here, Monty randomly picks a door that is simply not Alice's door. Should Alice switch here?

d) Fill in `montyAvoidPrize`. Here, Monty randomly picks a door that is simply not the prize door. Should Alice switch here?

e) An interesting cognitive question is: why do we have the initial intuition that switching shouldn't matter? Given your explorations, propose an answer.