Exercises

Exercise 6.1: Tricky Agents

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

Exercise 6.1.1

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".

```
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)
```

Exercise 6.1.2

You observe that Sally chooses `a`, and then `b`. How likely is it that she is deceptive?

What if you instead observed that she chose `b` and then `b` again?

Explain how deceptiveness and preferences interact to produce her actions.

Hint: Try conditioning on (not) deceive and visualize the possible action-pairs Sally might take.

Exercise 6.2: 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 will explore why this is the case.

For this problem, we will assume (condition on) the fact that we observe Monty opening the door that is neither Alice's door nor the prize door.

Exercise 6.2.1

The decision to 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?

```
///fold:
var removeBadItems = function(l, badItems) {
  return reduce(
    function(badItem, remainingL) {
      return remove(badItem, remainingL)
    },
    l,
```

```
    badItems
  )
}

const doors = [1, 2, 3]
///

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

  condition(montyDoor != prizeDoor)
  condition(montyDoor != aliceDoor)

  return ...
}

display("P(win) if Alice doesn't switch:")
viz.auto(
  Infer(
    {method: 'enumerate'},
    function() {
      model(false)
    }
  )
)
display("P(win) if Alice does switch:")
viz.auto(
  Infer(
    {method: 'enumerate'},
    function() {
      model(true)
    }
```

```
  )
)
```

Exercise 6.2.2

This time, fill in the code so that Monty behaves according to the original Monty Hall problem, i.e., picks the door that is neither the prize door nor Alice's door.

For both-avoiding Monty, you'll find that Alice *should* switch:

```
///fold:
var removeBadItems = function(l, badItems) {
  return reduce(
    function(badItem, remainingL) {
      return remove(badItem, remainingL)
    },
    l,
    badItems
  )
}

const doors = [1, 2, 3]
///

var montyAvoidBoth = function(aliceDoor, prizeDoor) {
  return Infer(
    {method: 'enumerate'},
    function() {
      return ...
    }
  )
}
var model = function(switch_cond) {
  var aliceDoor = ...
  var prizeDoor = ...
  var montyDoor = ...

  condition(montyDoor != prizeDoor)
  condition(montyDoor != aliceDoor)

  return ...
}
```

```
display("P(win) if Alice doesn't switch:")
viz.auto(
  Infer(
    {method: 'enumerate'},
    function() {
      model(false)
    }
  )
)
display("P(win) if Alice does switch:")
viz.auto(
  Infer(
    {method: 'enumerate'},
    function() {
      model(true)
    }
  )
)
```

Exercise 6.2.3

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, complete the probability table for Pr(AliceDoor, PrizeDoor, MontyDoor) under both `montyRandom` and `montyAvoidBoth`.

Your tables should look like the following:

Alice's Door	Prize Door	Monty's Door	Pr(AliceDoor, PrizeDoor, MontyDoor)
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.

Exercise 6.2.4

This time, fill in the code so that Monty randomly chooses between the two doors that aren't Alice's door. What should Alice do now?

```
///fold:
var removeBadItems = function(l, badItems) {
  return reduce(
    function(badItem, remainingL) {
      return remove(badItem, remainingL)
    },
    l,
    badItems
  )
}

const doors = [1, 2, 3]
///

var montyAvoidAlice = function(aliceDoor, prizeDoor) {
  return Infer(
    {method: 'enumerate'},
    function() {
      return ...
    }
  )
}
var model = function(switch_cond) {
  var aliceDoor = ...
  var prizeDoor = ...
  var montyDoor = ...

  condition(montyDoor != prizeDoor)
  condition(montyDoor != aliceDoor)

  return ...
}

display("P(win) if Alice doesn't switch:");
viz.auto(
  Infer(
    {method: 'enumerate'},
    function() {
```

```
      model(false)
    }
  )
)
display("P(win) if Alice does switch:")
viz.auto(
  Infer(
    {method: 'enumerate'},
    function() {
      model(true)
    }
  )
)
```

Exercise 6.2.5

This time, fill in the code so that Monty randomly chooses between the two doors that aren't the prize door. What should Alice do now?

```
///fold:
var removeBadItems = function(l, badItems) {
  return reduce(
    function(badItem, remainingL) {
      return remove(badItem, remainingL)
    },
    l,
    badItems
  )
}

const doors = [1, 2, 3]
///

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

```
var model = function(switch_cond) {
  var aliceDoor = ...
  var prizeDoor = ...
  var montyDoor = ...

  condition(montyDoor != prizeDoor)
  condition(montyDoor != aliceDoor)

  return ...
}

display("P(win) if Alice doesn't switch:")
viz.auto(
  Infer(
    {method: 'enumerate'},
    function() {
      model(false)
    }
  )
)
display("P(win) if Alice does switch:")
viz.auto(
  Infer(
    {method: 'enumerate'},
    function() {
      model(true)
    }
  )
)
```

Exercise 6.2.6

The psychological [meta-]question is, why do people have the initial intuition that switching shouldn't matter? Given your explorations, propose a hypothesis. Can you think of an experiment that would **test** this hypothesis?