1 ball, 2 ball, red ball, blue ball

By Melissa Dalis
Professor Susan Rodger
Duke University
June 2011
Overview

• In this tutorial, you’ll create a simulation for sampling red and blue marbles from two boxes.
• We’ll use 2 kinds of sampling: simple random sampling and stratified random sampling.
• Programming concepts included are visual and nonvisual arrays, variables, random numbers, shuffling, and other basic Alice animation.
Starting world

• Make sure you’ve downloaded the starting world from the website.
• Make sure you understand the objects in the object tree:
  – redArray and blueArray will display the red and blue balls chosen from the boxes in 2 lines at the top next to “Red:” (redText) and “Blue:” (blueText)
  – The box1Balls and box2Balls folders contain all the balls in their respective boxes.
Displaying the instructions

• Make sure you’ve downloaded the instructions image from the website.
• We want these instructions to display at the beginning of the game.
• At the top of Alice, click on File, select Make Billboard, and choose the instructions image.
• Click on Add objects, and rotate, move, and resize the instructions so that they cover most of the screen, and click Done.
• Make sure you understand the instructions.
• If the player wants to see a **stratified random sample**, we’ll randomly choose one of the two boxes and *then* choose the number of balls that the player specifies *from that one box*.
• If the player wants to see a **simple random sample**, we’ll randomly choose that certain number of balls, *regardless of box*.
• Let’s hide the instructions for now so we can see everything behind it while we’re coding.
• Once you understand the instructions, click on **instructions** in the object tree, and go to its properties.
• Change **isShowing** to false.
Creating events

• When the player types S, we want the instructions to disappear and to show a simple random sample.
• In the top right corner, click create new event, and select When a key is typed.
• Click any key, and select letters, S.
• From the bottom, drag a Do in order onto Nothing.
• From instructions’ properties, drag isShowing into the Do in order, and select false.
• We want to keep track of if the user chose a simple or stratified sample.
• Click on **world** in the **object tree**, and go to its **properties** tab.
• Click **create new variable**, name it **simple**, and select type **Boolean**.
• Drag **simple** into the **Do in order**, and select **set value, true**.
• Using the **clipboard**, make a copy of this event to modify it for a stratified random sample.
• The changes are: change **S** to **T**, and **true** to **false**.
Create new method: chooseBalls

• Click on world in the object tree, and go to its methods tab.
• Click create new method, and name it chooseBalls.
• This single method will work for both kinds of samples and for any number of balls we want to choose.
• So create 2 parameters of type Number: numChosenBalls and numTotalBalls.
• Create a parameter of type Object named allBalls, check off make a List, and change List to Array.
• Make sure that the ordering is the same as in the picture because this will be important later.
Player’s guess

• To make this a little more interactive, let’s have the player guess how many red and blue balls we’ll choose.

• Create two new variables inside the method, of type Number: redGuess and blueGuess.

• Drag redGuess into the method, and select set value, 1, for now.

• From world’s functions tab, drag ask user for a number onto the 1, select other, and type “How many red balls do you think we’ll choose?”

• Right click on that line to make a copy, and change redGuess to blueGuess and red to blue.
Controlling the number of balls

• We need the player to choose a positive number of balls, less than or equal to the total number of balls.
• Let’s keep prompting the user until he gives a reasonable answer.
• Drag a While into the method, and select true.
• From world’s functions, drag either a or b, or both onto the true, and select true.
• Drag numChosenBalls onto the first true, and select >, expressions, numTotalBalls.
• Drag numChosenBalls onto the second true, and select <, 1.
• Drag an If/Else into the While, and select true.
• Drag `numChosenBalls` onto the true, and select `>`, expressions, `numTotalBalls`.
• Drag `numChosenBalls` into the first `Do Nothing`, and select set value, 1, for now.
• From world’s functions, drag ask user for a number onto the 1, select other, and type “There aren’t even that many balls in the box! Choose another number.”
• Drag an If/Else into the Else, and select true.
• Drag numChosenBalls onto the true, and select \(<, 1\).
• Drag numChosenBalls into the first Do Nothing, and select set value, 1, for now.
• From world’s functions, drag ask user for a number onto the 1, select other, and type “Come on, try something positive!”
Shuffling the balls

• Drag a **Loop** into the bottom of the method, and select expressions, `numTotalBalls`, and click **show complication version**.

• In order to get a random sample of balls, we will shuffle all the balls in `allBalls` and choose the first `numChosenBalls` of them.

• In the method, **create a Number variable** named `rand`.

• Drag `rand` into the **Loop**, select **set value**, **1**, for now.

• From **world’s functions**, drag **random number** onto **1**.

• Click the purple **more 3 times** to set the **minimum** to **0**, **maximum** to `numTotalBalls`, and **integerOnly** to **true**.
Theory behind shuffling

• Suppose we want to make this change:

```
[Green] [Purple] [Green]  
```

• We move the green to a temp spot, put the purple in its place, then put the green in the purple’s former place.

```
[Green] [Purple] [Green]  
```

• Notice that this temp spot is necessary to store one of the items while the other moves places.
So let’s code that

- In the method, create a new Object variable named `tempBall`.
- Drag `tempBall` into the Loop, and select set value, camera, for now.
- Drag `allBalls` onto camera, and select ith item from array, expressions, rand.

- Drag `allBalls` into the Loop, and select set item `<index>` to `<item>`, expressions, rand, camera.
- Drag `allBalls` onto camera, select ith item from array, expressions, index, and set the duration to 0.
- Drag `allBalls` into the `Loop`, and select `set item <index> to <item>`, `expressions`, `index`, `expressions`, `tempBall`.
- Set the `duration` to 0 seconds.

```
set item index to tempBall in allBalls duration = 0 seconds
```

- Now we want to move the first `numChosenBalls` to the top of the screen, where we have 2 visual (yet hidden) arrays to keep track of the red and blue balls we choose.
- Drag another `Loop` into the bottom of the method, and select `expressions`, `numChosenBalls`, and click `show complicated version`.
- In this new `Loop`, we’ll use `tempBall` as the ball that we are moving.
- Drag `tempBall` into the `Loop` and select `set value`, `camera`, for now.
Showing our sample of balls

• Drag allBalls onto camera, and select ith item from array, expressions, index.
• We want the ball to move to redArray if it’s red and blueArray if it’s blue.
• Below that line, drag an If/Else and select true.
• To access the color property, we’ll first have to get it from an item in the object tree.
• From box1’s properties in the object tree, drag color onto the true, and select ==, red.
• Drag tempBall onto box1.
Create two new number variables in the method: redCount and blueCount, and set their starting values to 0.

These will keep track of how many red and blue balls we have chosen, respectively.

In the object tree, go to redArray’s properties.

We want the red ball we just chose to move to the spot for item redCount in redArray.

Drag elements into the first Do Nothing, and select set item <index> to <item>, expressions, redCount, expressions, tempBall, and set duration to 0.5.
• Below that line, drag **redCount** and select increment by 1.

• From **blueArray’s** properties tab, drag elements into the **Else**, select set item `<index>` to `<item>`, expressions, **blueCount**, expressions, **tempBall**, and set duration to 0.5.

• Below that line, drag in **blueCount**, and select increment by 1.
Finding the difference

• Now we want to see how close redGuess and blueGuess were from redCount and blueCount, so create two new number variables in the method, named redDiff and blueDiff.

• Drag redDiff into the bottom of the method, and select set value, expressions, redCount.

• Click the white arrow, and select math, -, expressions, redGuess.

• From world’s functions, drag absolute value of a onto (redCount – redGuess).

• Do the same for blueDiff with blueCount and blueGuess.
Astronaut say...

• From astronaut’s methods tab, drag say into the bottom of the method and type “Your guess for the number of red balls was only”.

• Drag a joined with b onto that text, select default string, and put another a joined with b onto default string.

• Drag what as a string onto the first default string, and select camera, for now.

• Also from world’s functions, drag int a as a string onto redDiff, and select expressions, redDiff.
• Change the last **default string** to say “balls off from the number we sampled!”
• Change the **duration** to 5 seconds.
• Do the same for the blue balls.

• Now we’ll use `redDiff` and `blueDiff` as variables to calculate the percentages of red and blue balls.
• Drag `redDiff` into the method, and select **set value**, **expressions**, `redCount`.
• Click **math** next to **redCount**, and select /, **numChosenBalls**.

• Click the arrow after (**redCount**/ **numChosenBalls**), and select **math**, *, **100**.

• Do the same for **blue**.

• Click the arrow after (**redCount**/ **numChosenBalls**), and select **math**, *, **100**.

• Do the same for **blue**.
Analyzing the data

• Statistically speaking, our guesses (if they’re reasonable) are usually closer to the true answer when the sample size is larger.

• Drag astronaut say into the method, and type “If we had sampled more balls, your guess might have been even closer to the true answer!”

• Change the duration to 5 seconds.

• Drag astronaut say into the method, and add a joined with b’s 4 times until you have 5 places to enter text.
• Set the **duration** to 5 seconds.
• Change the text fields so that the line looks like this:

```
Set the duraFon to 5 seconds.
```

• If there were half red and have blue balls, for example, the astronaut would say, “We chose 50% red balls and 50% blue balls.”
• Next have the astronaut say “Press restart to see another sample” for 3 seconds.
my first method

- Go to `world.my first method`.
- Drag in a `While`, and select `true`.
- Find instructions in the object tree, and go to its properties tab.
- Drag `isShowing` onto the `true`, and leave the `Do Nothing` as it is.
- Drag in an `If/Else` at the bottom and select `true`.
- From `world’s properties`, drag `simple` onto the `true`. 
• For 2 seconds, have the astronaut say “Great! You chose a simple random sample.”
• Create a number variable named numBalls.
• Drag numBalls below the astronaut say line, and select set value, 1.
• From world’s functions, drag ask user for a number onto the string and type “There are 25 total balls. How many do you want to randomly choose?”
Keeping track of the balls

- For a **simple random sample**, we choose balls without caring which box they are from, so we want an array of all of the balls.

- Under **world’s properties**, create a new Object variable.

- Name the variable **bothBoxBalls**, select make a List, and change List to Array.

- Click new item until you have items 0-24 filled with red1, ..., red15 and blue1, ..., blue 10 in any order, but make sure that you include all of them.
• From world’s methods, drag choose below the numBalls set value line, and select expressions numBalls, expressions, numBalls, expressions, world.bothBoxBalls.

• Change the second numBalls to 25 since there are 25 total balls.

• Find instructions’ properties in the object tree.

• Set isShowing to true.

• Play your world—it should work for a simple random sample! (Not stratified yet, though.)
For a stratified sample

• In the Else, have the astronaut say, “Great, you chose a stratified random sample!” for 2 seconds then “First, we’ll choose a box.” for 2 seconds.

• We want to choose each box with equal probability.

• Create a new boolean variable in the method named firstBox.

• Drag firstBox below the astronaut say lines, and select set value, true for now.

• From world’s functions, drag choose true probabilityOfTrue onto true, and select 50%.
• Below that line, still in the Else, drag in another If/Else and select true.
• Drag firstBox onto the true.
• Drag Box1Text into the first Do Nothing, and select set color to, green.
• Below that line drag numBalls, and select set value, 1.
• Drag ask user for a number onto the 1, and type “How many balls should we pick from box 1?”
• Under the Else, do the same for box2.
allBox1Balls and allBox2Balls

- For a stratified sample, we are either sampling from box1 or box2, so we need an array for each case.

- Create a **10-item object array** named `onlyBox1Balls`, and insert each of the 10 items from the `box1Balls` folder (red1,...,red5, blue1,...,blue5) into the spaces, in any order.

- Similarly, create a **15-object object array** named `onlyBox2Balls` for each of the 15 items in the `box2Balls` folder (red6,...,red15, blue6,...blue10).
Finishing up

• Go back to my first method, and scroll to the bottom.
• From world’s methods, drag choose below the first numBalls set value line, and have the 3 fields be numBalls, 10, world.onlyBox1Balls, in that order.
• Below the second numBalls set value line, drag in choose and have the fields be numBalls, 15, world.onlyBox2Balls, in that order.
Check your code:

```plaintext
numBalls = 1, firstBox = true

While instructions .isShowing
  Do Nothing

If world.simple
  astronaut say Great! You chose a simple random sample. duration = 2 seconds
  numBalls set value to ask user for a number question = There are 25 total balls. How many
  numChosenBalls = numBalls numTotalBalls = 25 allBalls = world.bothBoxes
```
astronaut \_\_ say Great, you chose a stratified random sample! \_ \_ duration = 2 seconds \_ \_ more...

astronaut \_\_ say First, we'll choose a box. \_ \_ duration = 2 seconds \_ \_ more...

firstBox \_\_ set value to choose true 0.5 (50%) \_ \_ of the time \_ \_ more...

If firstBox

Box1Text \_\_ set color to \_ \_ more...

numBalls \_\_ set value to ask user for a number question = How many balls should we pick?

world.chooseBalls numChosenBalls = numBalls \_ \_ numTotalBalls = 10 \_ \_ allBalls = world.ones(numBalls)

Else

Box2Text \_\_ set color to \_ \_ more...

numBalls \_\_ set value to ask user for a number question = How many balls should we pick?

world.chooseBalls numChosenBalls = numBalls \_ \_ numTotalBalls = 15 \_ \_ allBalls = world.ones(numBalls)
world.chooseBalls(numChosenBalls, numTotalBalls, allBalls)

redGuess = 1,
blueGuess = 1,
rand = 1,
tempBall = <None>,
redCount = 0,
blueCount = 0,
redDiff = 1,
blueDiff = 1

redGuess set value to ask user for a number question = How many red balls do you think we'll choose?

blueGuess set value to ask user for a number question = How many blue balls do you think we'll choose?

While either numChosenBalls > numTotalBalls or numChosenBalls < 1

If numChosenBalls > numTotalBalls

numChosenBalls set value to ask user for a number question = There aren't even that many balls

Else

If numChosenBalls < 1

numChosenBalls set value to ask user for a number question = Come on, try something

Else

Do Nothing
redDiff set value to absolute value of (redCount - redGuess)

blueDiff set value to absolute value of (blueCount - blueGuess)

astronaut say Your guess for the number of red balls was only joined with

astronaut say Your guess for the number of blue balls was only joined with

redDiff set value to (redCount / numChosenBalls) * 100

blueDiff set value to (blueCount / numChosenBalls) * 100

astronaut say If we had sampled more balls, your guess might have been even closer to the

astronaut say We chose joined with int redDiff as a String as a string

astronaut say Click restart to see another sample. duration = 2 seconds more...
Good job!!
Play your world!