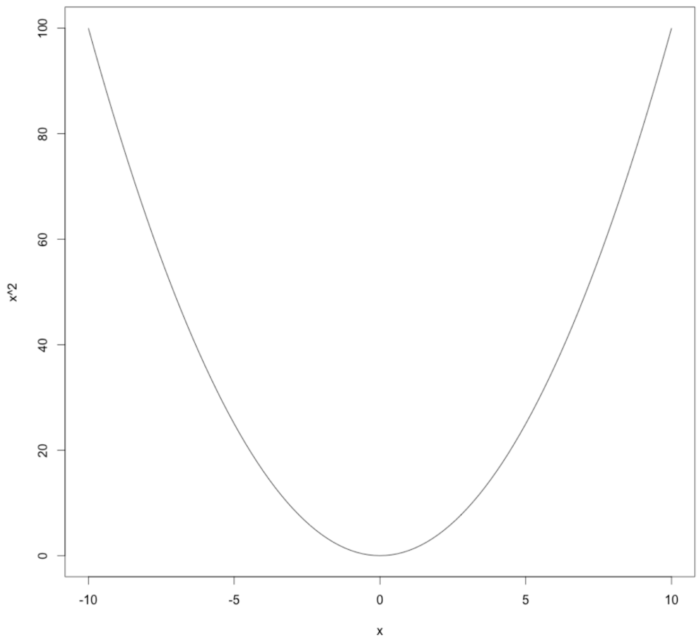
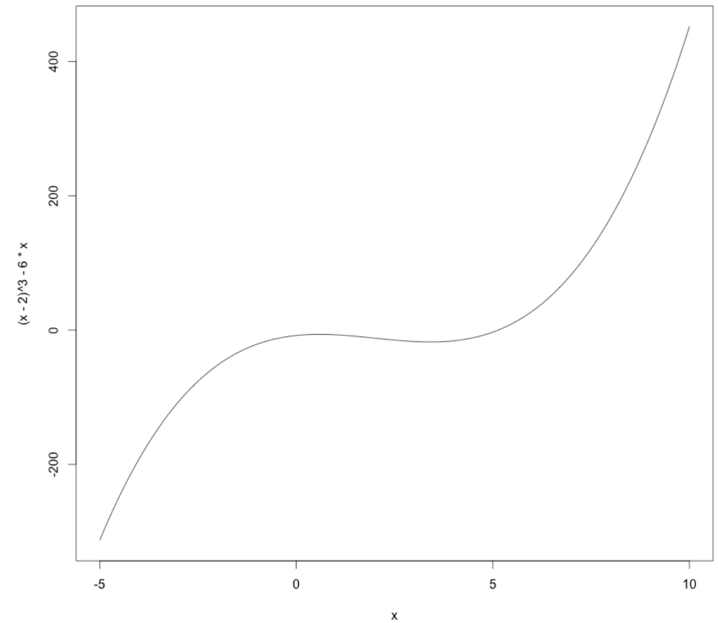
Lecture 4: Monte Carlo simulation - Urn Models

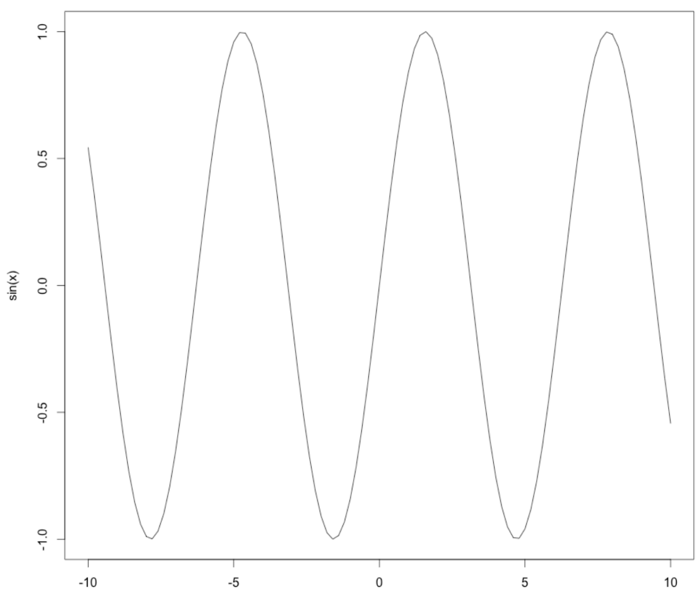
- Assignment 1 due today
- Last week's lab tasks…

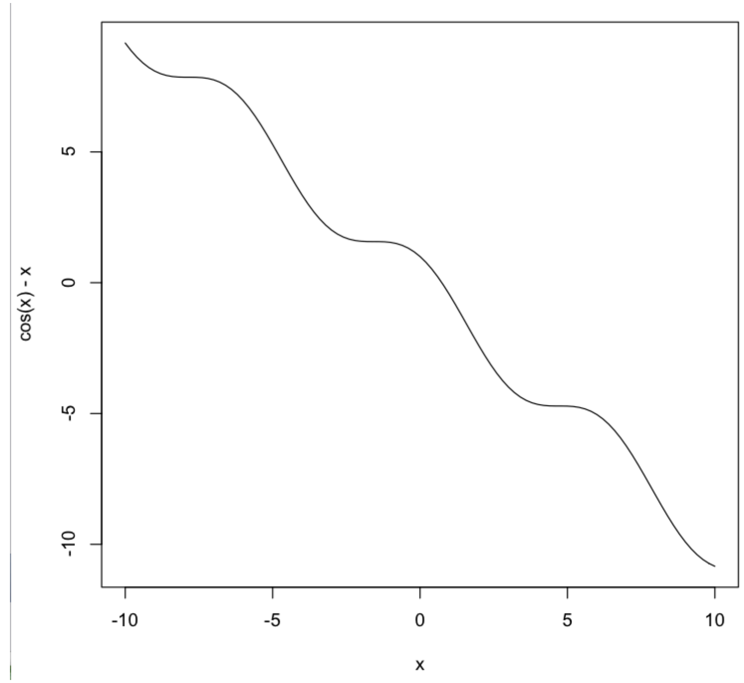
Lab task (30 minutes)

- Implement Newton-Raphson in R
- Test it on the following functions:
  - $f(x)=x^2$
  - $f(x)=(x-2)^3-6x$
  - $f(x)=\sin(x)$
  - $f(x)=\cos(x)-x$
- Draw plots showing progress of the algorithm
- When you have something that works, or that is giving you problems, push it to GitHub with @pmarjora in your commit message.


x^2
x
-10 -5 0 5 10
0 20 40 60 80 100


(x - 2)^3 - 6 * x
x
-5 0 5 10
-200 0 200 400


sin(x)
x
-10 -5 0 5 10
-1.0 -0.5 0.0 0.5 1.0


cos(x) - x
x
-10 -5 0 5 10
-10 -5 0 5

Examinable assignment 2a: The Art Show

- Take the code for Newton-Raphson on the complex plain from Github:
  - Basic [slow] code: https://github.com/PM520-Spring-2020/Week3-NewtonRaphsonFractals. [has pretty graphics].
  - Better [faster] code: https://github.com/PM520-Spring-2020/Week3-FasterFractals
- Use it to draw some fractals for some functions **other than those we saw in today's class**. [Alternatively, Google "Julia Sets"]
- Write a report in Rmarkdown including your R code, and describing the functions you tried, but also including some pretty pictures for display at the start of class in three week's time. **They must be drawn by your R code**. (R's ggplot package is your friend here.)
- Each person should also commit one picture. You are strongly encouraged to give it a pretentious/artsy name. We will vote on who has the best (and I will provide some sort of prize for the best picture).
- **Deadline: 3 weeks from today.**

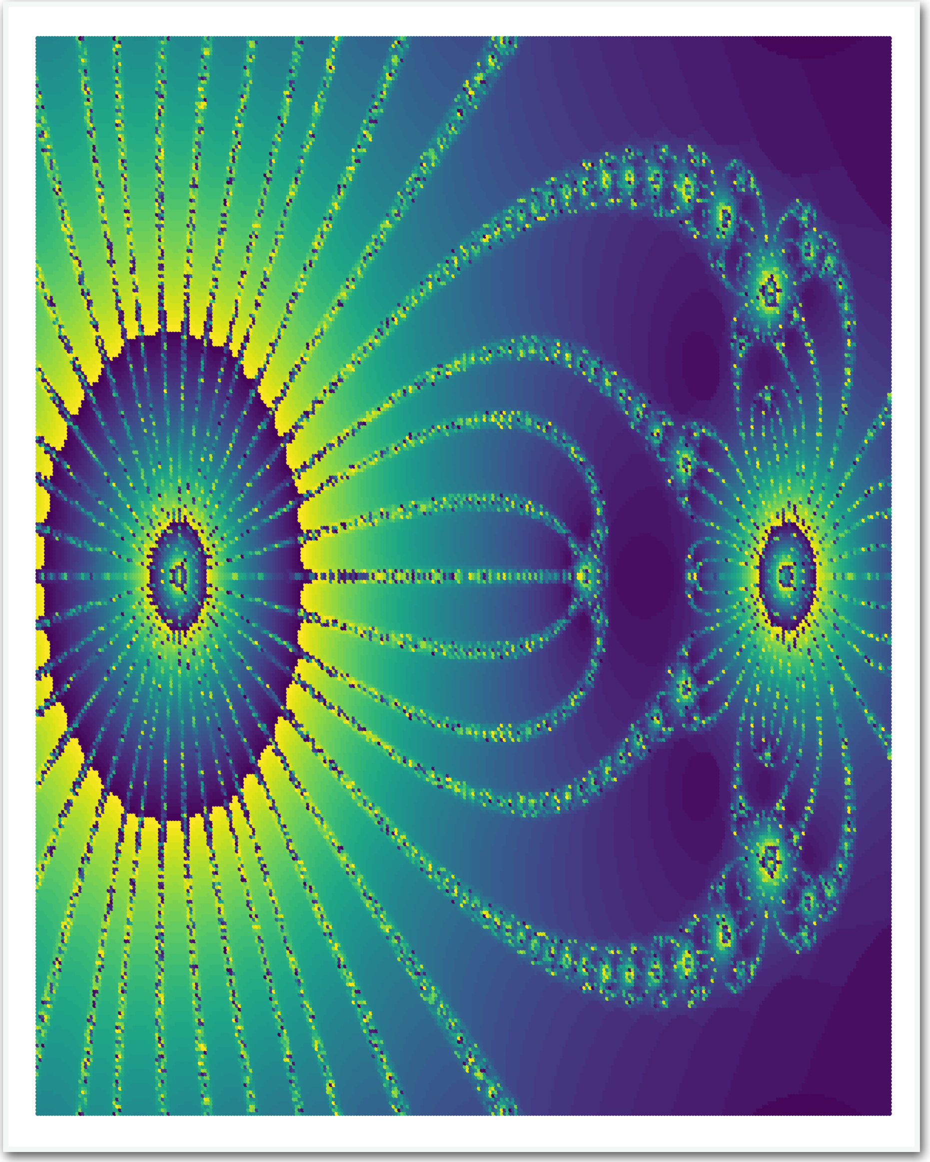
Examinable Assignment 2b

1. Write a program to implement the Secant method.
2. Test it, using $x_0=1$ and $x_1=2$ on:
    1. cos(x)-x
    2. log(x)-exp(-x)

   Compare it with the performance of Newton-Raphson on the same functions.
3. Write a function to show a plot of the iterations of the algorithm.
4. Turn it in as a knitted Rmd document.

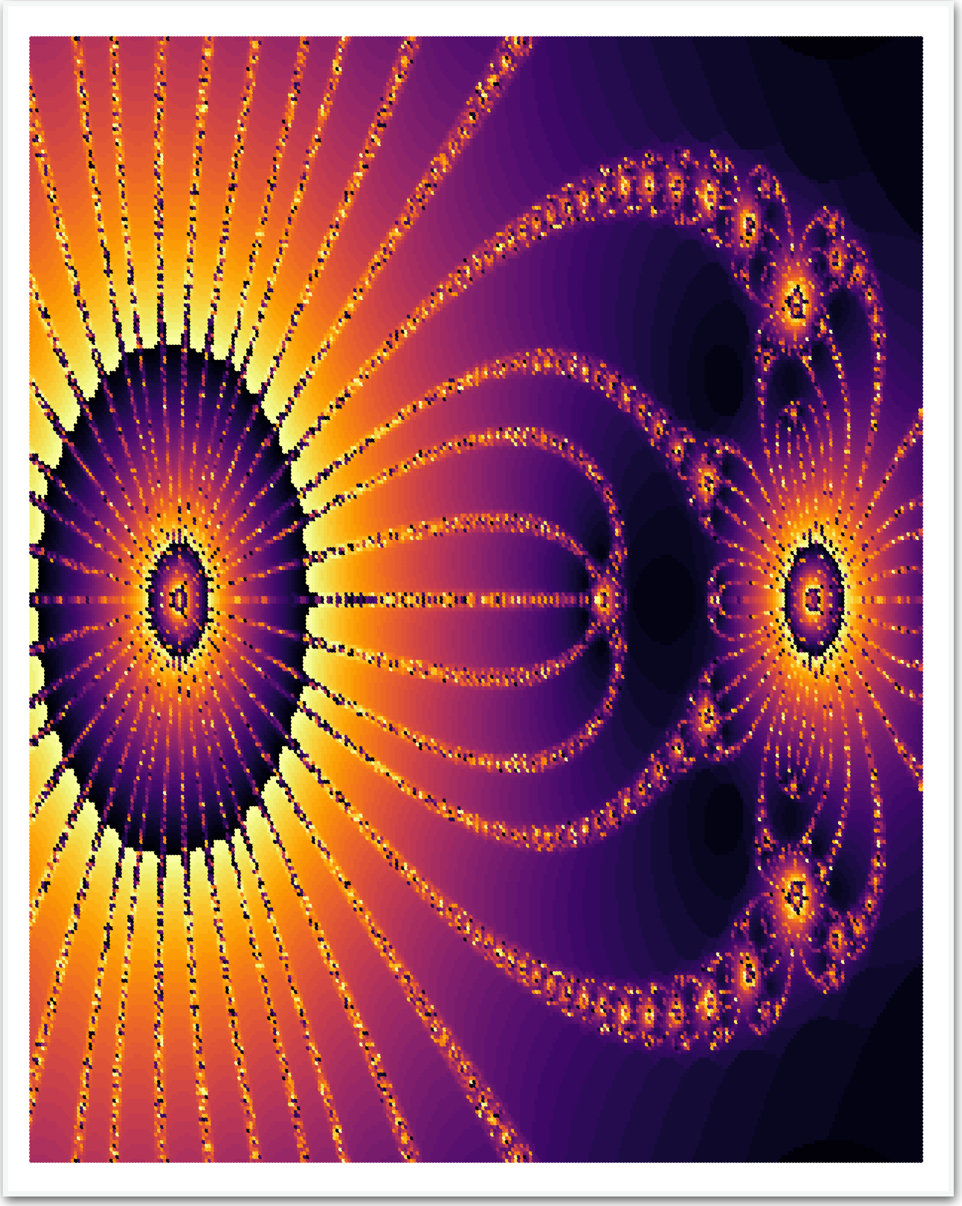
Non-examinable lab task

- Implement the bisection method
- Test it on the following functions:
  - $f(x)=x^3$
  - $f(x)=x^3-2x^2+x$
  - $f(x)=\sin(x)$



```
library(viridis)
palette(viridis(256))
```

This is the function 35z^9 - 180z^7 + 378z^5 - 420z^3 + 315z   (taken from http://www.chiark.greenend.org.uk/~sgtatham/newton/)



```
library(viridis)
palette(inferno(256))
```

This is the function 35z^9 - 180z^7 + 378z^5 - 420z^3 + 315z (taken from http://www.chiark.greenend.org.uk/~sgtatham/newton/)

Examinable Assignment 2b

1. Write a program to implement the Secant method.
2. Test it, using $x_0=1$ and $x_1=2$ on:
    1. cos(x)-x
    2. log(x)-exp(-x)

   Compare it with the performance of Newton-Raphson on the same functions.
3. Write a function to show a plot of the iterations of the algorithm.
4. Turn it in as an Rmd document.

This week's 538.com "Riddler" Questions - The hard one

A grasshopper lands somewhere randomly on your lawn, which has an area of 1 square meter.

As soon as it lands, it jumps 30 centimeters.

What shape should your lawn be to maximize the chances that the grasshopper will still be on the lawn after the 30-centimeter jump?

This week's 538.com "Riddler" Questions - The hard one

A grasshopper lands somewhere randomly on your lawn, which has an area of 1 square meter.

As soon as it lands, it jumps 30 centimeters.

What shape should your lawn be to maximize the chances that the grasshopper will still be on the lawn after the 30-centimeter jump?

(Hint: It's not a circle.)

Extra credit: What if the grasshopper jumps *X* centimeters instead?

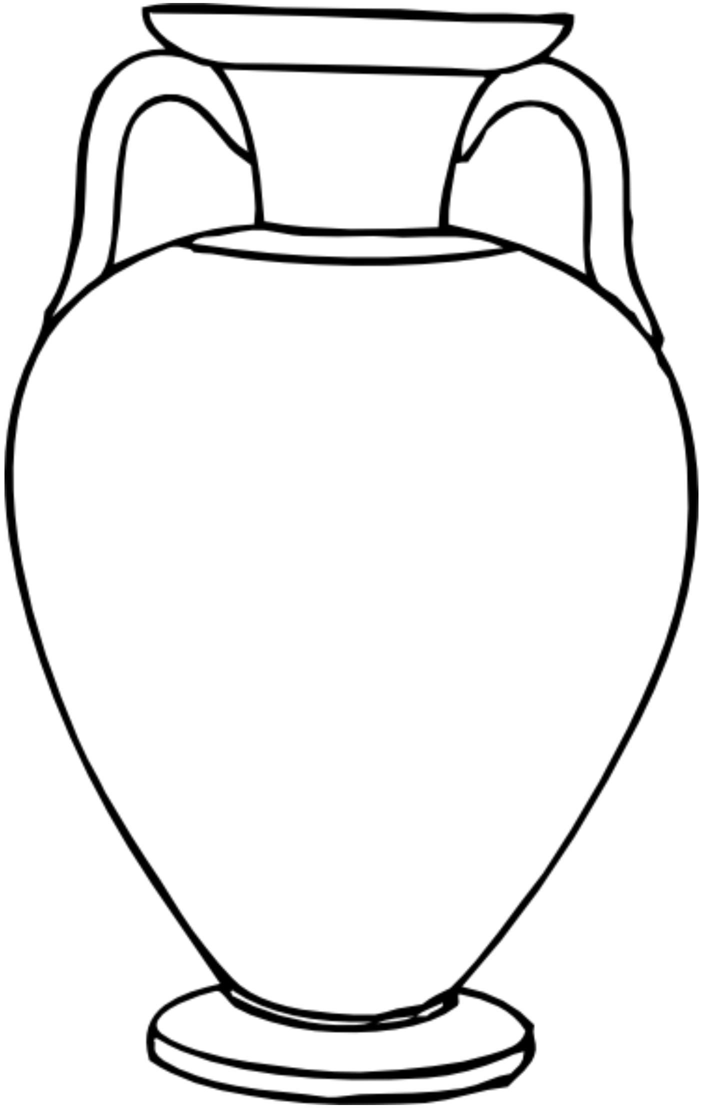
Extended Monte Carlo Simulation example: Urn models

www.hiwtc.com

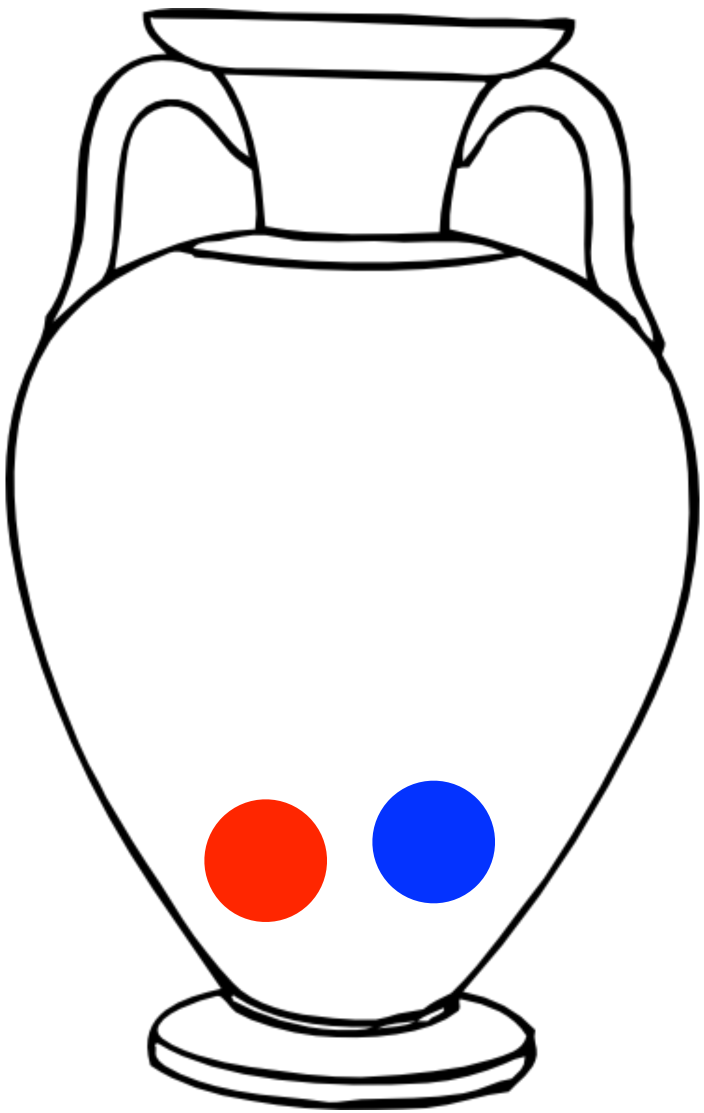
1. Draw balls from an urn
2. Take actions that depend upon what you drew
3. Ask questions about what happens after *n* draws, say.

Used as very early model of epidemics.

In pictures

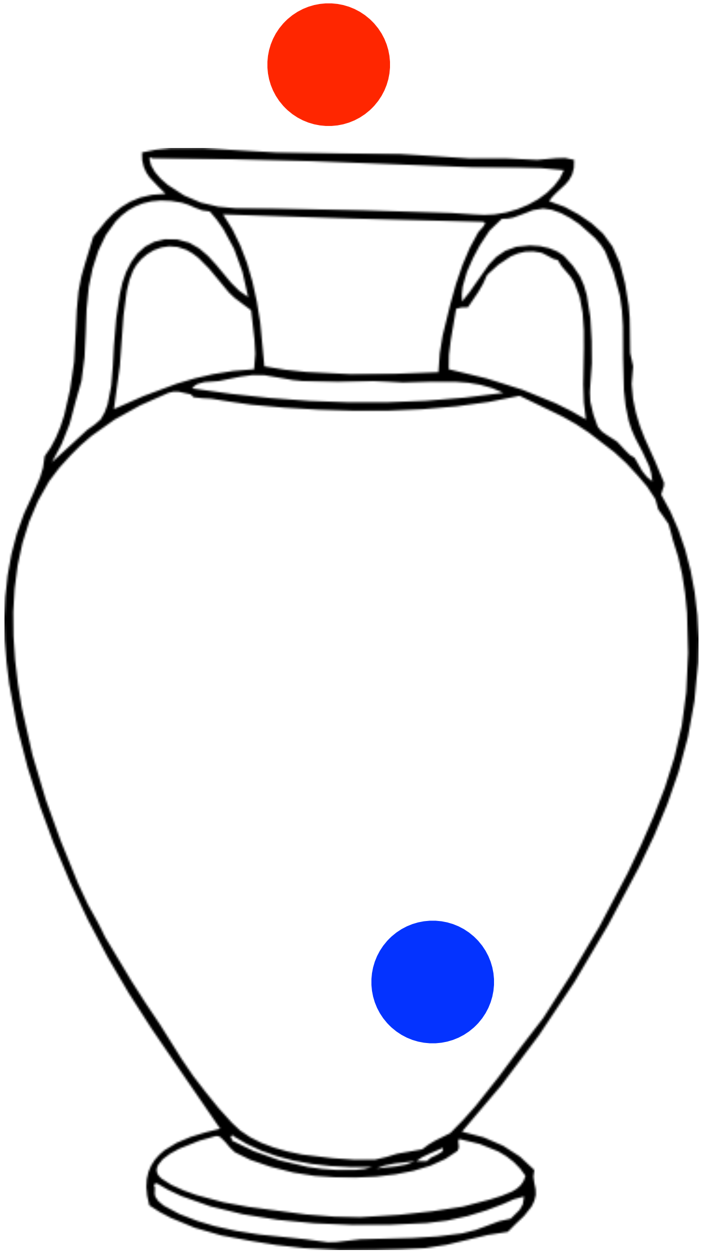


Simplest form



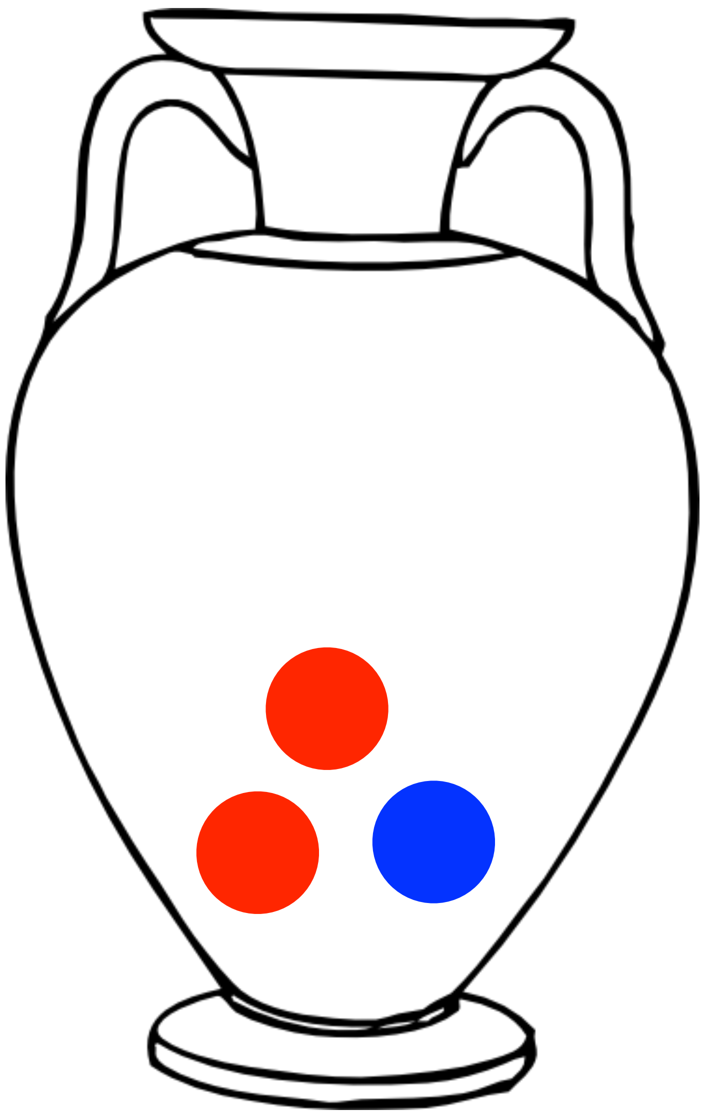
1. Start with two balls, of different colors.

Simplest form



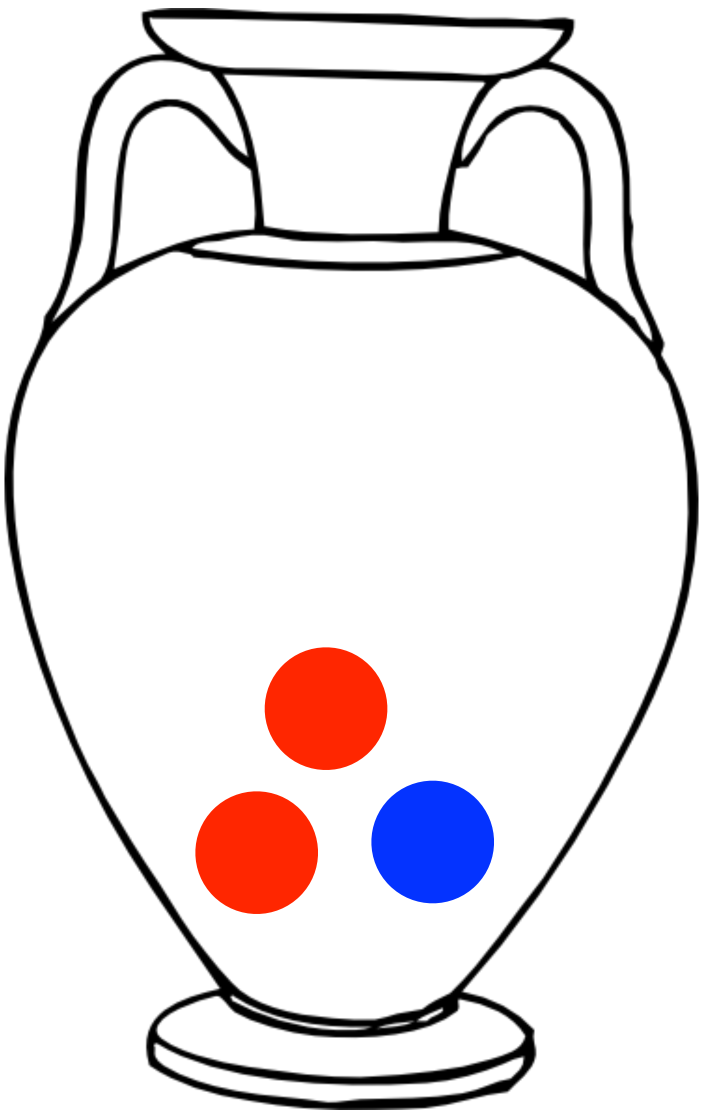
1. Start with two balls, of different colors.
2. Draw a ball from the urn.

Simplest form



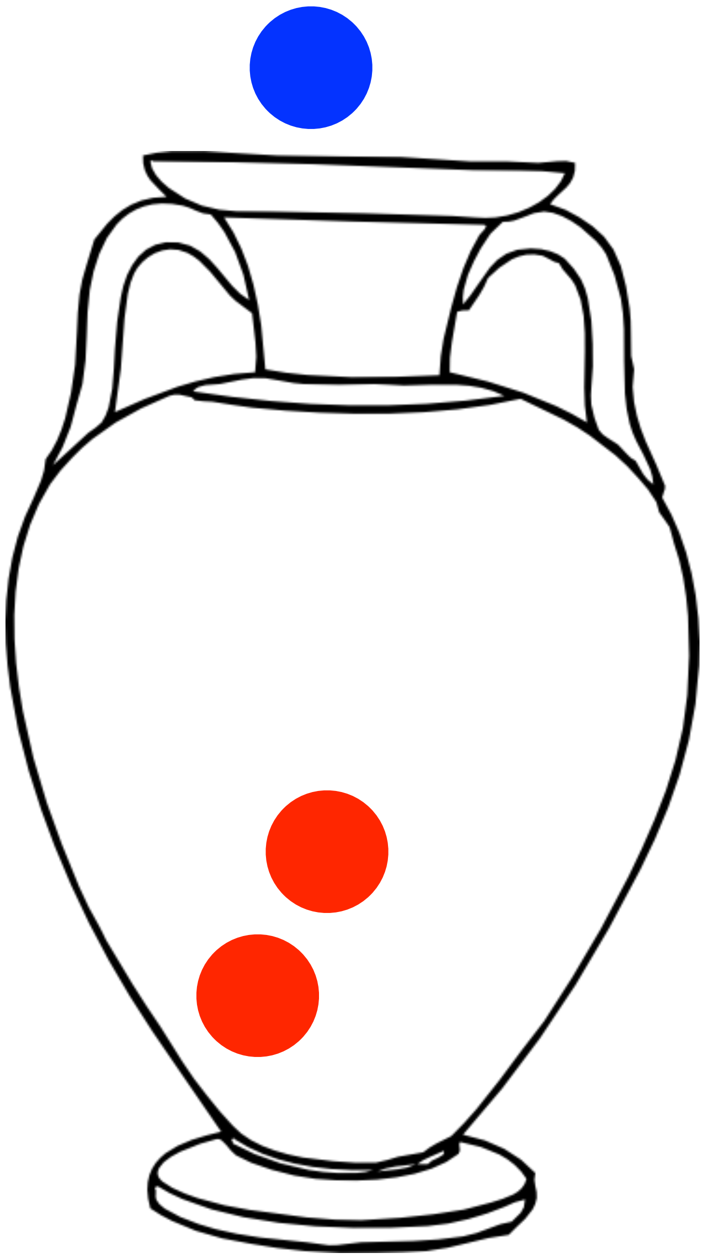
1. Start with two balls, of different colors.
2. Draw a ball from the urn.
3. Put the ball back into the urn *and add another ball of the same color.*

Simplest form



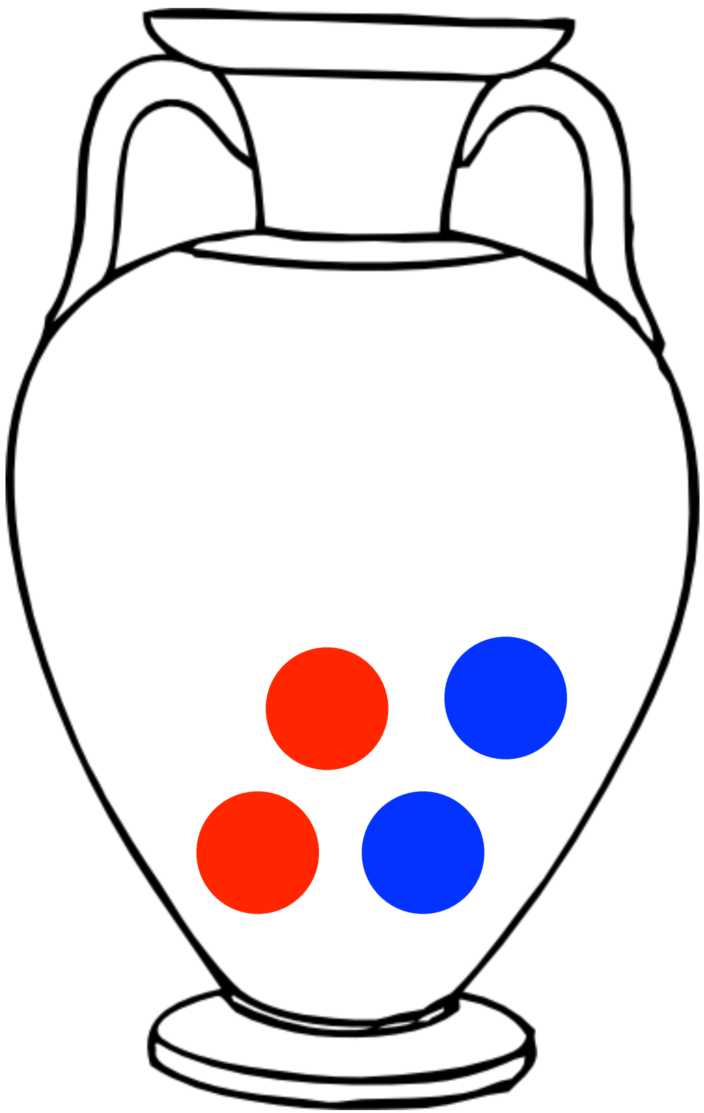
1. Start with two balls, of different colors.
2. Draw a ball from the urn.
3. Put the ball back into the urn *and add another ball of the same color.*
4. Iterate.

Simplest form



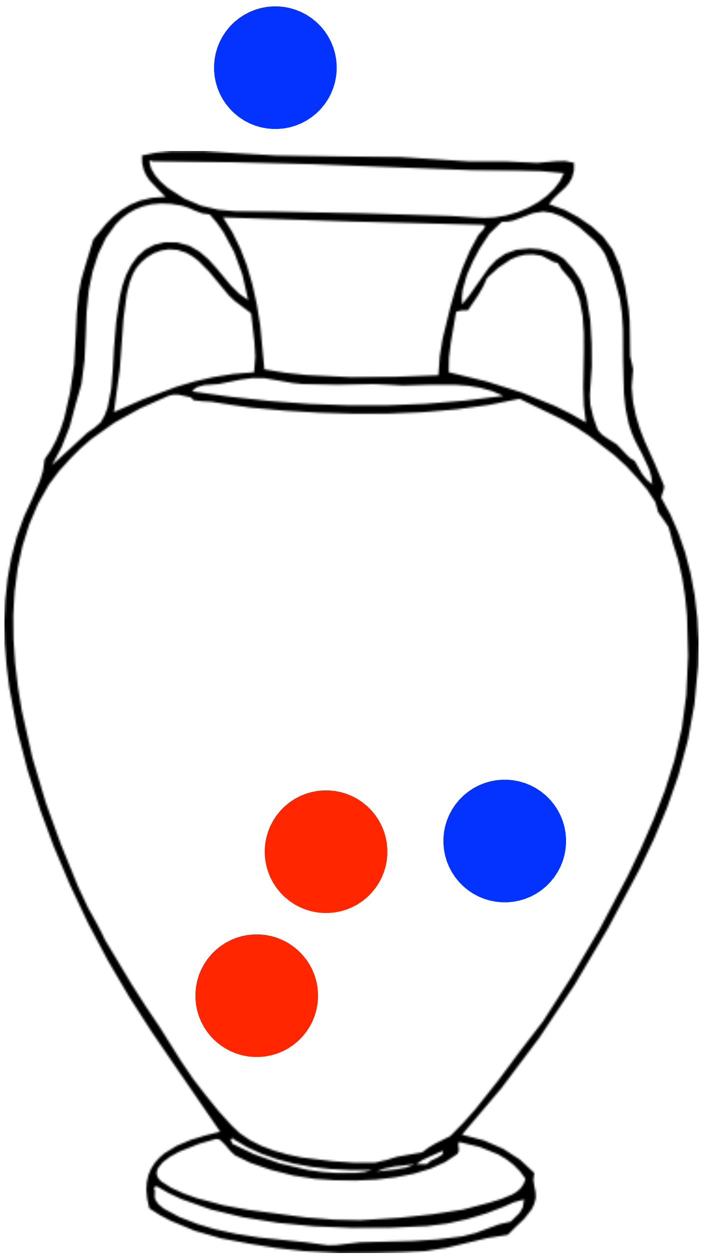
1. Start with two balls, of different colors.
2. Draw a ball from the urn.
3. Put the ball back into the urn *and add another ball of the same color.*
4. Iterate.

Simplest form



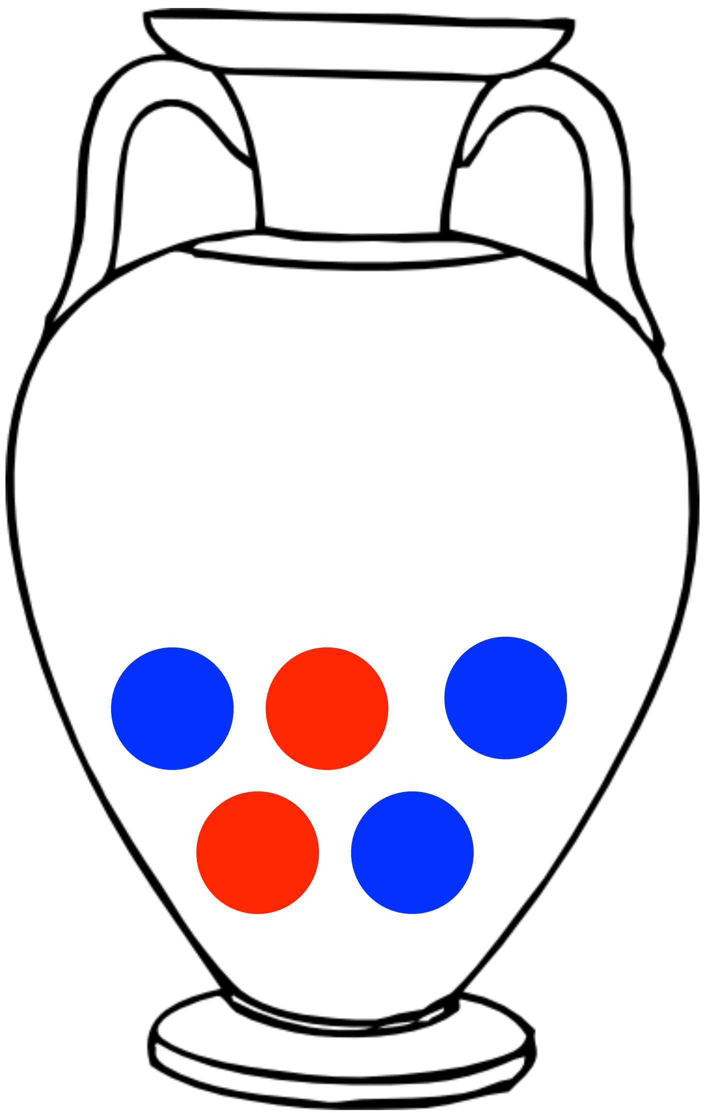
1. Start with two balls, of different colors.
2. Draw a ball from the urn.
3. Put the ball back into the urn *and add another ball of the same color.*
4. Iterate.

Simplest form



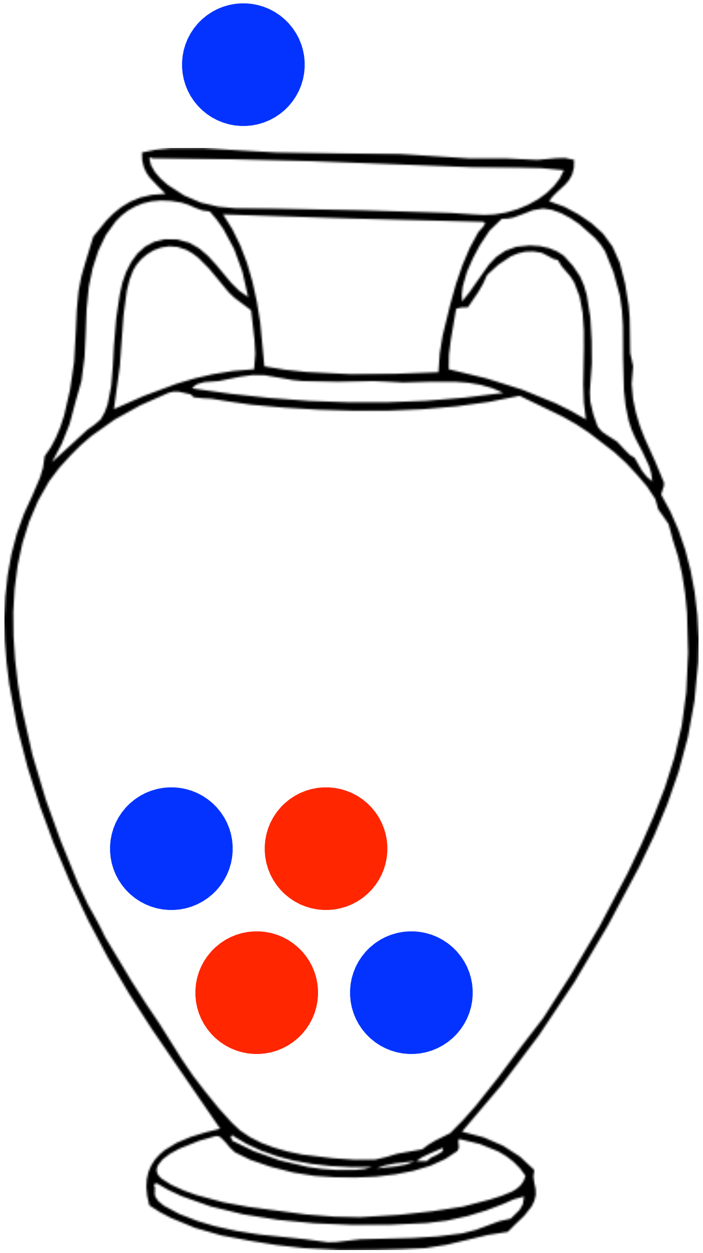
1. Start with two balls, of different colors.
2. Draw a ball from the urn.
3. Put the ball back into the urn *and add another ball of the same color.*
4. Iterate.

Simplest form



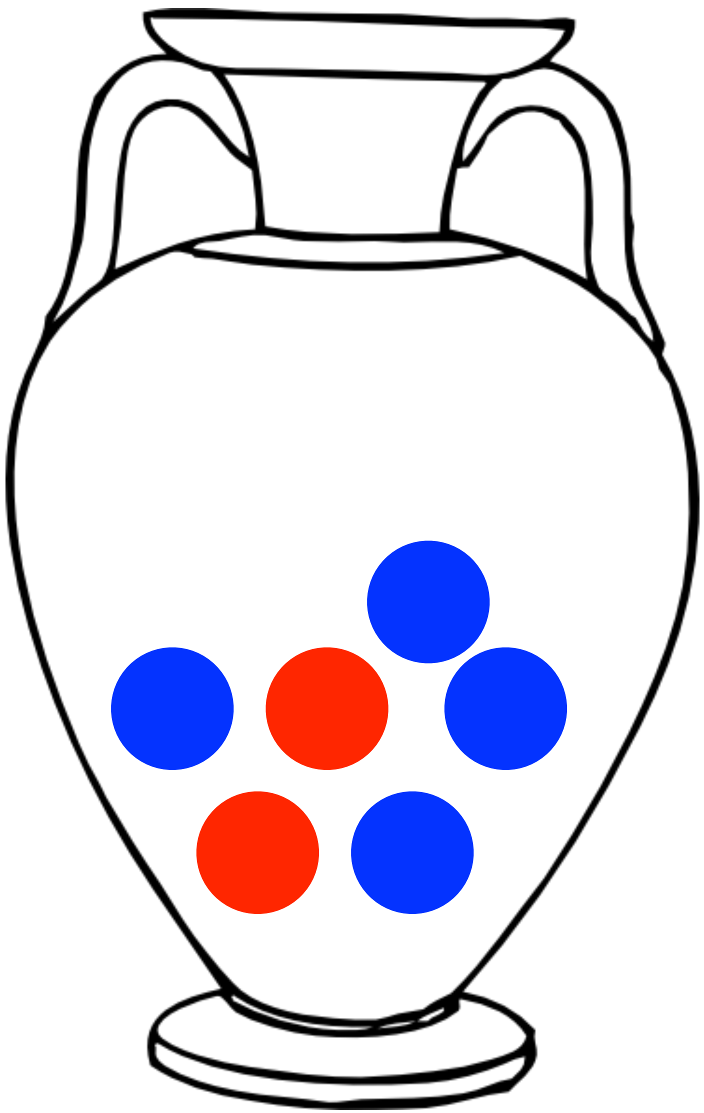
1. Start with two balls, of different colors.
2. Draw a ball from the urn.
3. Put the ball back into the urn *and add another ball of the same color.*
4. Iterate.

Simplest form



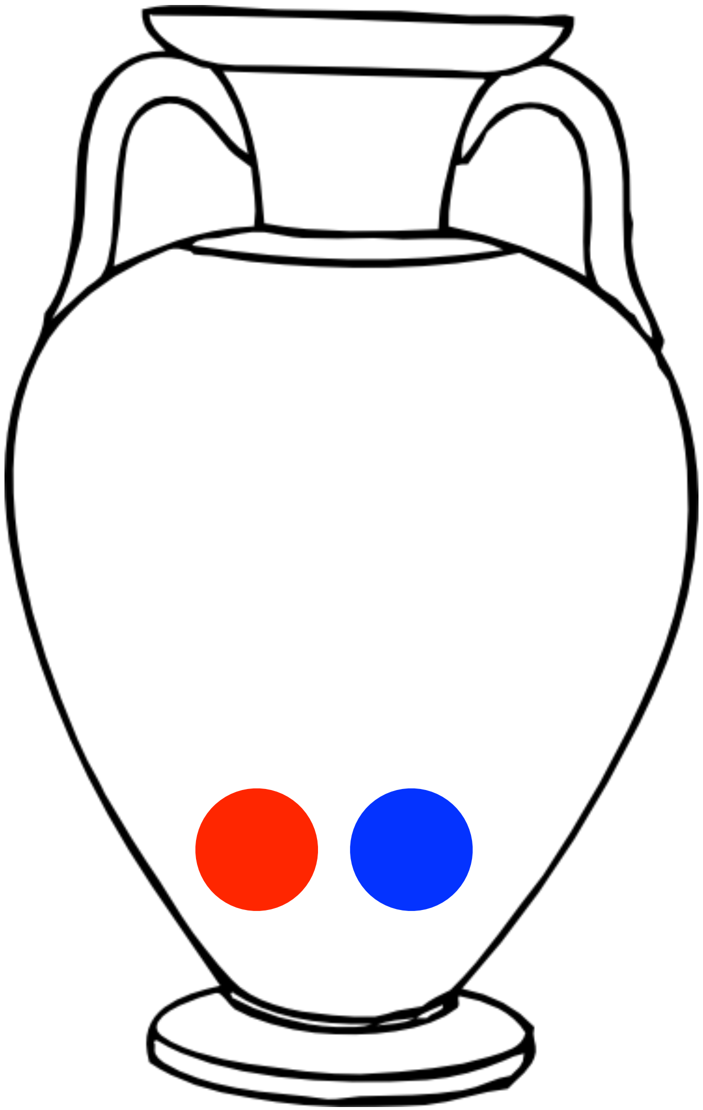
1. Start with two balls, of different colors.
2. Draw a ball from the urn.
3. Put the ball back into the urn *and add another ball of the same color.*
4. Iterate.

Simplest form



1. Start with two balls, of different colors.
2. Draw a ball from the urn.
3. Put the ball back into the urn *and add another ball of the same color.*
4. Iterate.

Full statement:



1. Start with two balls, of different colors.
2. While there are less than n balls in the urn, repeat the following:
    1. Draw a ball from the urn.
    2. Put the ball back into the urn *and add another ball of the same color.*

Pseudocode for Polya Urn model -Version 1

[UrnPseudoCode.R in Week4-UrnModels]

```
 # set the random number seed
set.seed(16)

# define your variables
# How many balls do we start with
InitialNumberOfBalls<-2
# How many balls do we need eventually
TargetNumberOfBalls<-10

# Now write a function to simulate the Urn model
UrnSim <- function(InitialNBalls, TargetNBalls){
  # set up the initial state of the urn
  Urn<-mat.or.vec(1,TargetNBalls)

  # we will start with two balls of different colors: "red" and "blue"
  # The first element of the urn vector should be set to "red", the second element should be set to "blue" (say)

  # set up a counter (NumberOfBalls) to keep track of how many balls we have
  NumberOfBalls<-sum(Urn=="red")+sum(Urn=="blue")

  # set-up a loop that pulls a ball from the urn and takes the appropriate action
  while (NumberOfBalls<TargetNBalls){
                  # draw a ball (WhichBall)
                  # return the ball and add another one like it
                  # increase the counter of how many balls we have in the urn
  }

  # return some summary of what is in the urn when we are done (e.g. number of red balls)
}

# now run the function
NumTrials <- 1000   # how many urns to simulate
TrialResults <- rep(0,NumTrials) # somewhere to put the results
for(i in 1:length(TrialResults)){
  TrialResults[i] <- UrnSim(InitialNumberOfBalls,TargetNumberOfBalls)
}
```

How to draw a ball...

```
# Here’s how I declared my urn and set its initial state
Urn<-mat.or.vec(1,TargetNumberOfBalls)
# What is the initial state
Urn[1]<-"blue"
Urn[2]<-"red"

# choose the ball
WhichBall<-sample(1:NumberOfBalls,1) #NumberOfBalls is how many #balls there are now, so it starts at 2 and increases by one each time

# what color is that ball?
WhatColorIsIt<-Urn[WhichBall]
# add another ball of the same color
Urn[NumberOfBalls+1]<-WhatColorIsIt
```

Lab Task #1

- Working in groups - code up the Urn model.
- Start with 1 red and one blue ball
- Continue until you have 50 balls
    1. What is the expected number of red balls at the end? (Intuition?)
    2. What is the distribution of the number of red balls? (Plot a histogram - what properties do you think the distribution will have?)
- Suggest: Use **at least** 10000 replicates.
- We will then look at the results.
    - Empirical estimate of expected value of a distribution is just the mean of the observed values (i.e. mean number of red balls across the 10000(say) reps.)
    - Empirical estimate of a distribution is the histogram.

Function to count the red balls

```
CountBalls<-function(Urn,ColorToCount)
{
   # works with the urn "Urn", and returns the number of balls of color ColorToCount
   # we use the fact that the function "==" will be applied to each element of the
    vector Urn
   return (sum(Urn==ColorToCount))
}
```

Lab Task #2

- Code up the Urn model
- Start with any number, n, of red and blue balls
- Assume n-1 of the starting balls are red (and so 1 is blue), for 1<k<n.
- Continue until you have 50 balls

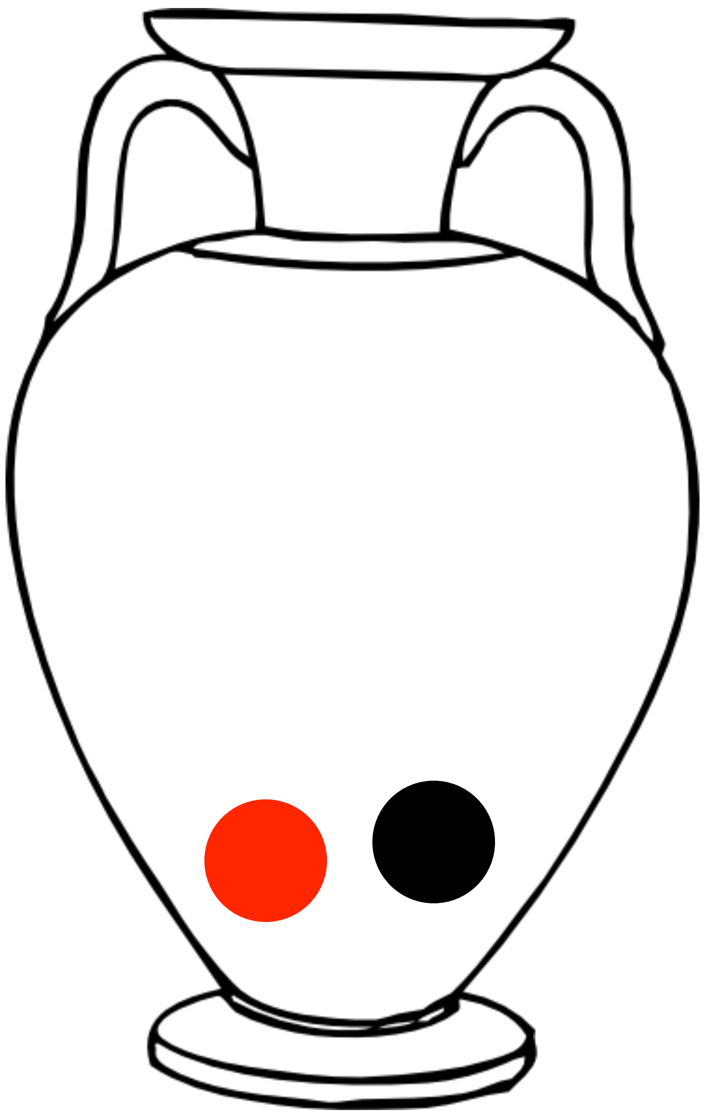
1. What is the expected number of red balls at the end, as a function of the number we start with? (Draw a plot of mean # of reds, versus #reds at start). (Also draw a plot of mean #reds versus proportion of reds at start.)
2. Is this described by a simple relationship?

Lab Task #3 - Generalized Urn

- Use the urn model in which we start with any number of balls, and in which when we remove a ball, instead of returning it and adding one more ball of the same color, we return it and add B balls of the same color, for some B.
- Use two starting balls (one red, one blue).
- Try B=1,2,3,4,5,...10, and drawing until you have at least 100 balls each time.
- What does this do to:

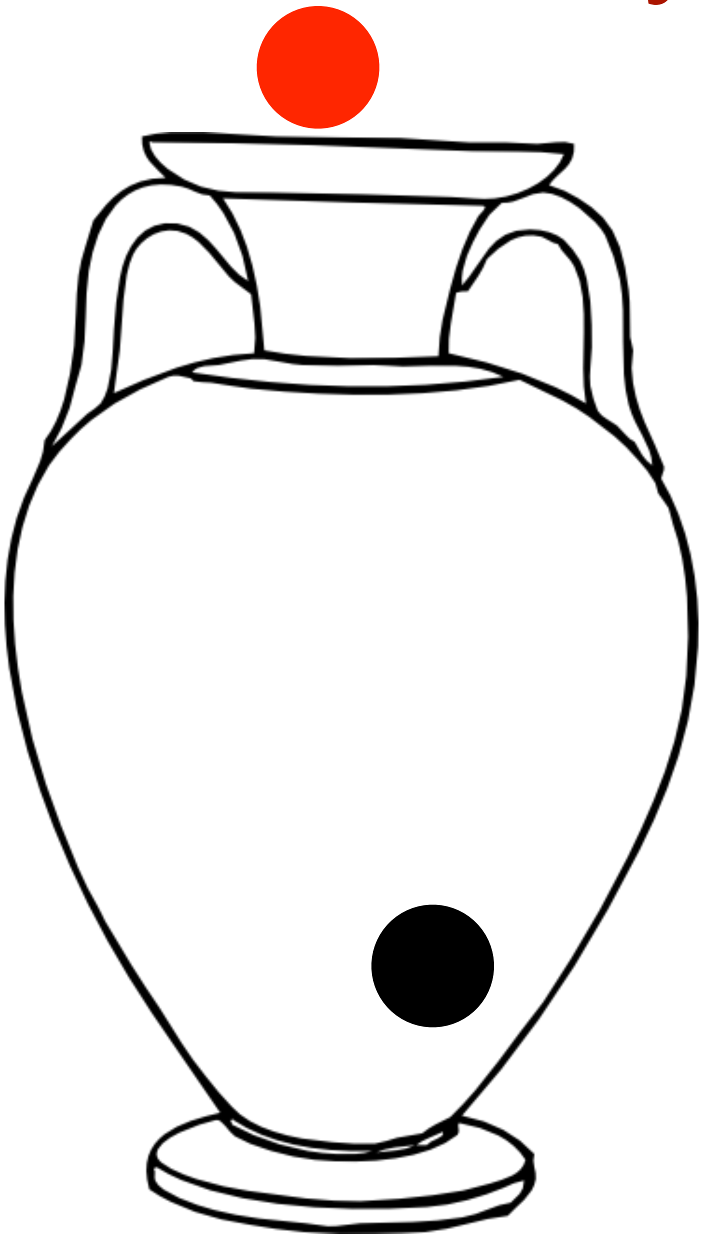
1. The Expected number/fraction of red (say) balls at the end?
2. The distribution of the number/fraction of red balls at the end?

The Polya Urn version 2



Add a black (special color) ball.
Start with two balls, one of which is black.

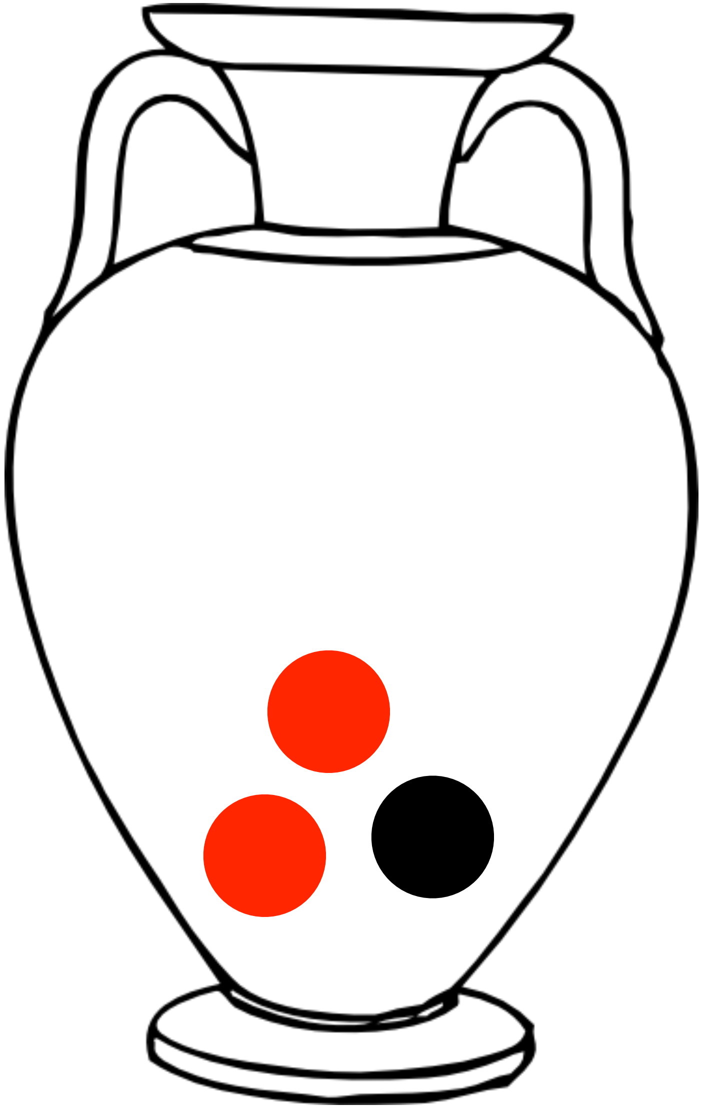
The Polya Urn version 2



Add a black (special color) ball.

If you draw a non-black ball, proceed as before (add an additional ball of the same color)

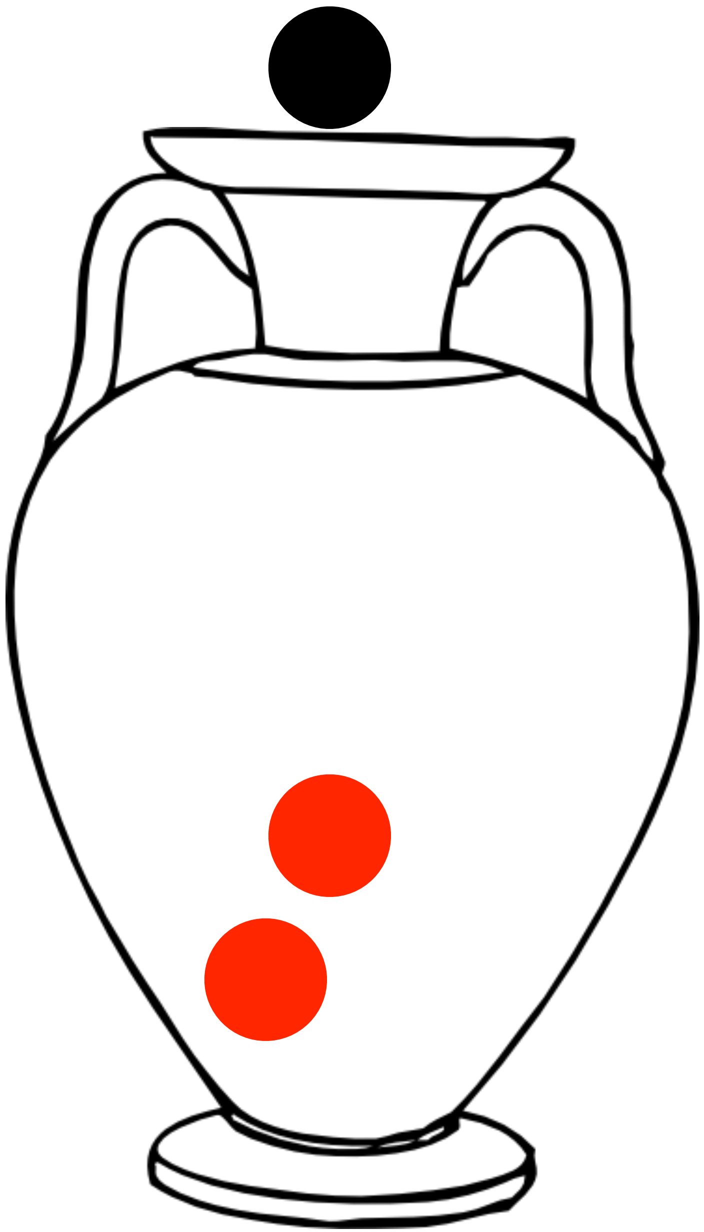
The Polya Urn version 2



Add a black (special color) ball.

If you draw a non-black ball, proceed as before (add an additional ball of the same color)

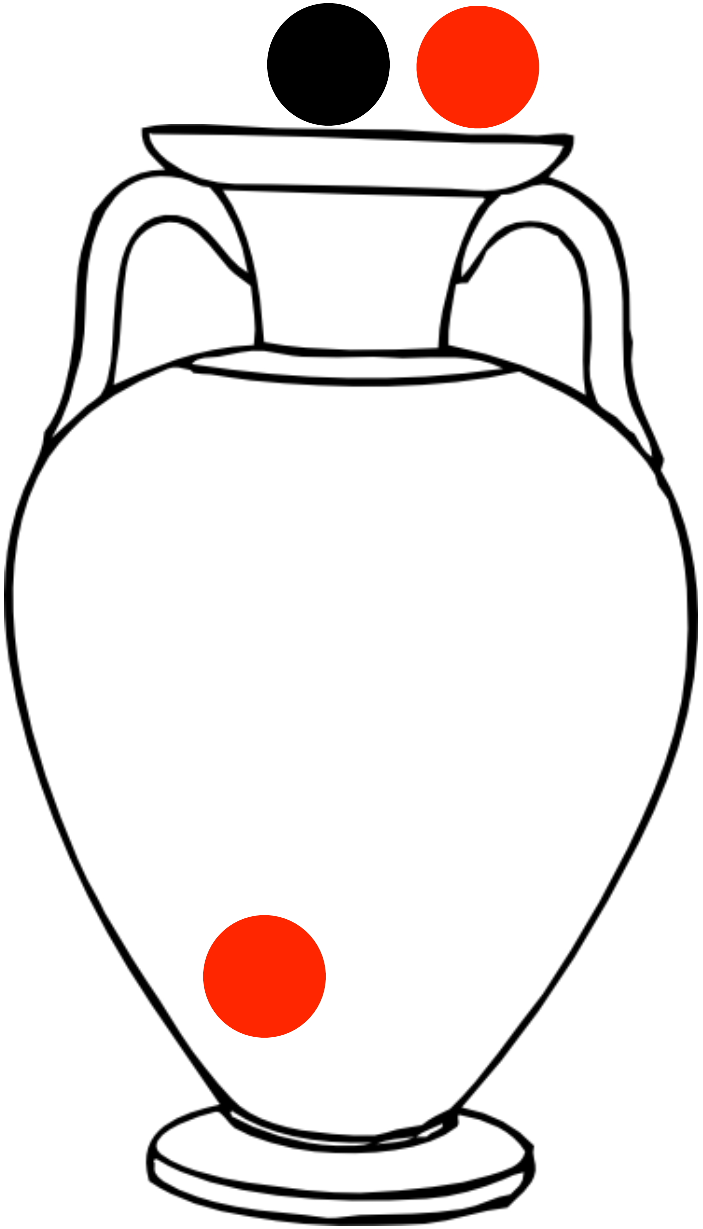
The Polya Urn version 2



Add a black (special color) ball.

If you draw a black ball do the following:
1.Keep the black ball for now.

The Polya Urn version 2

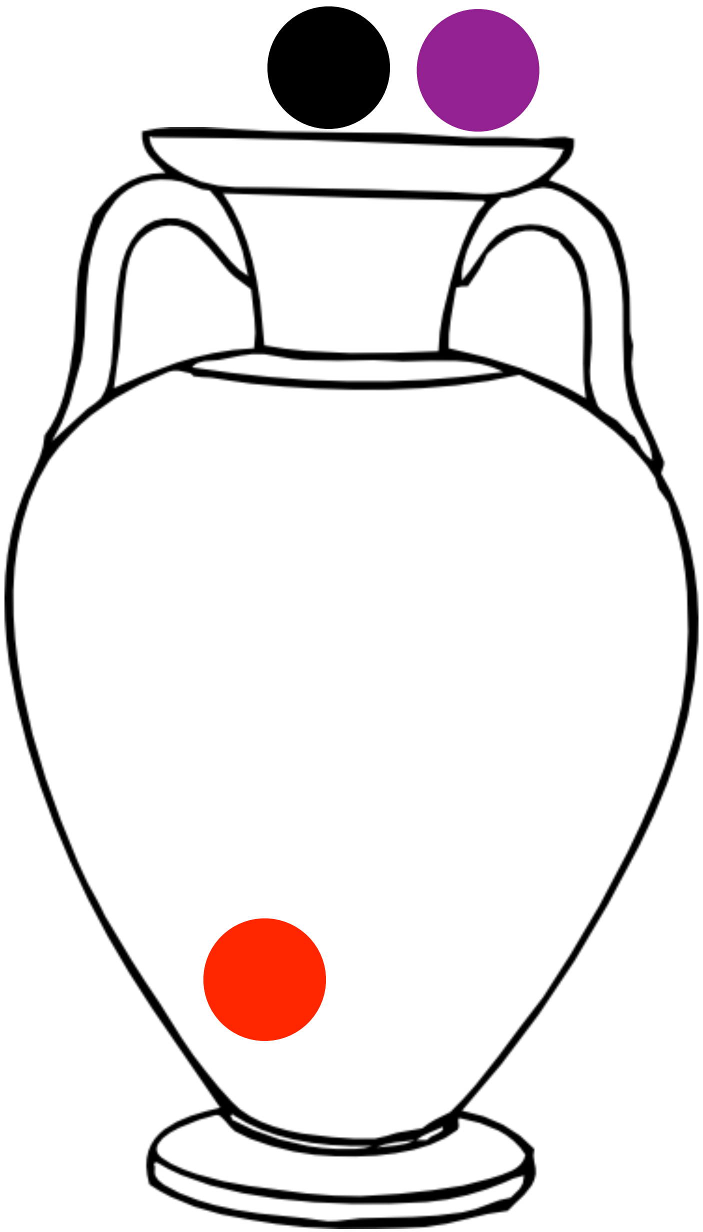


Add a black (special color) ball.

If you draw a black ball do the following:

1. Keep the black ball for now.
2. Draw another ball from the urn

The Polya Urn version 2

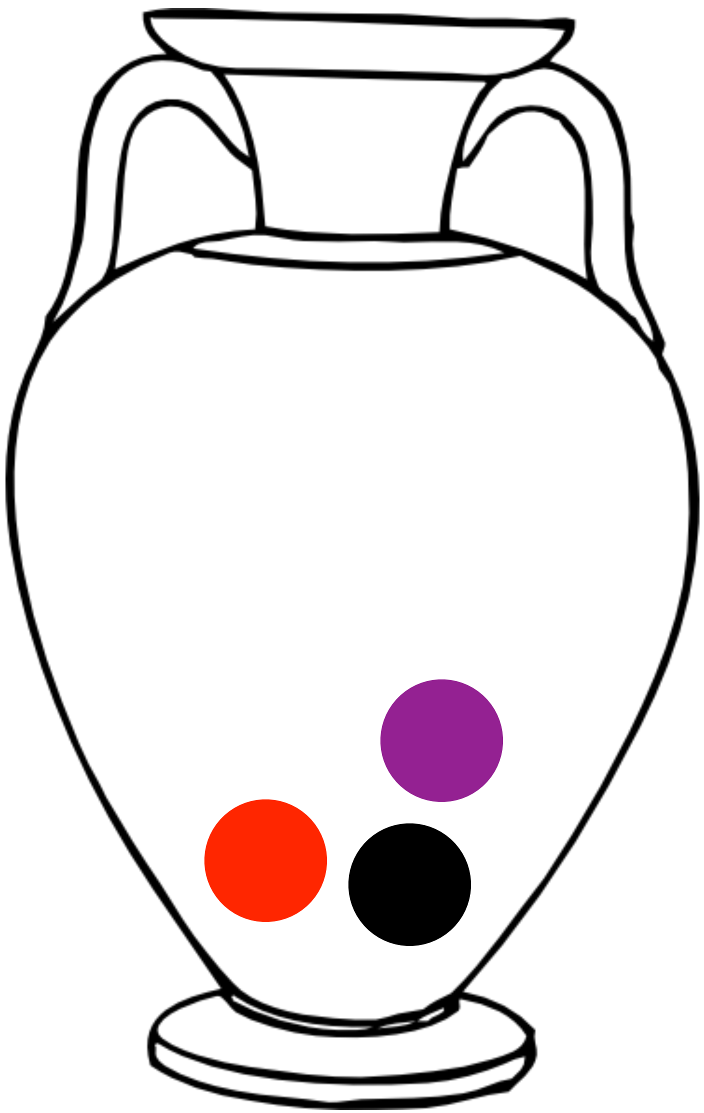


Add a black (special color) ball.

If you draw a black ball do the following:

1. Keep the black ball for now.
2. Draw another ball from the urn.
3. Change the color of the second ball, making it a color that is not currently in the urn.

The Polya Urn version 2

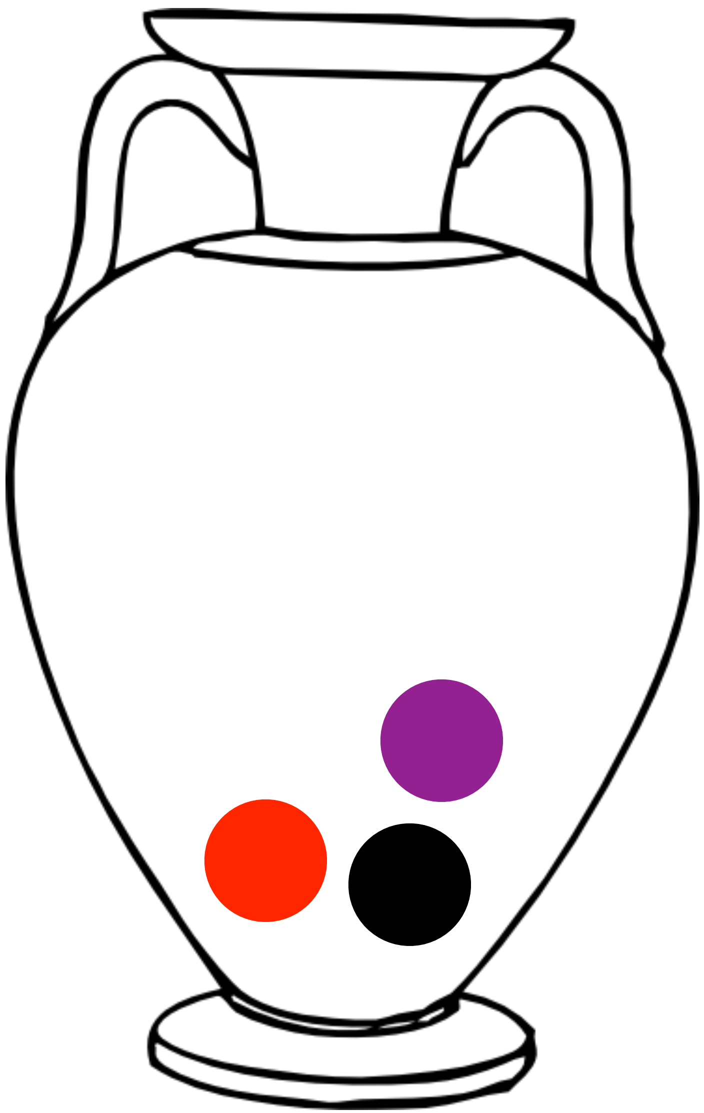


Add a black (special color) ball.

If you draw a black ball do the following:

1. Keep the black ball for now.
2. Draw another ball from the urn.
3. Change the color of the second ball, making it a color that is not currently in the urn.
4. Put both balls back in the urn.

The Polya Urn version 2

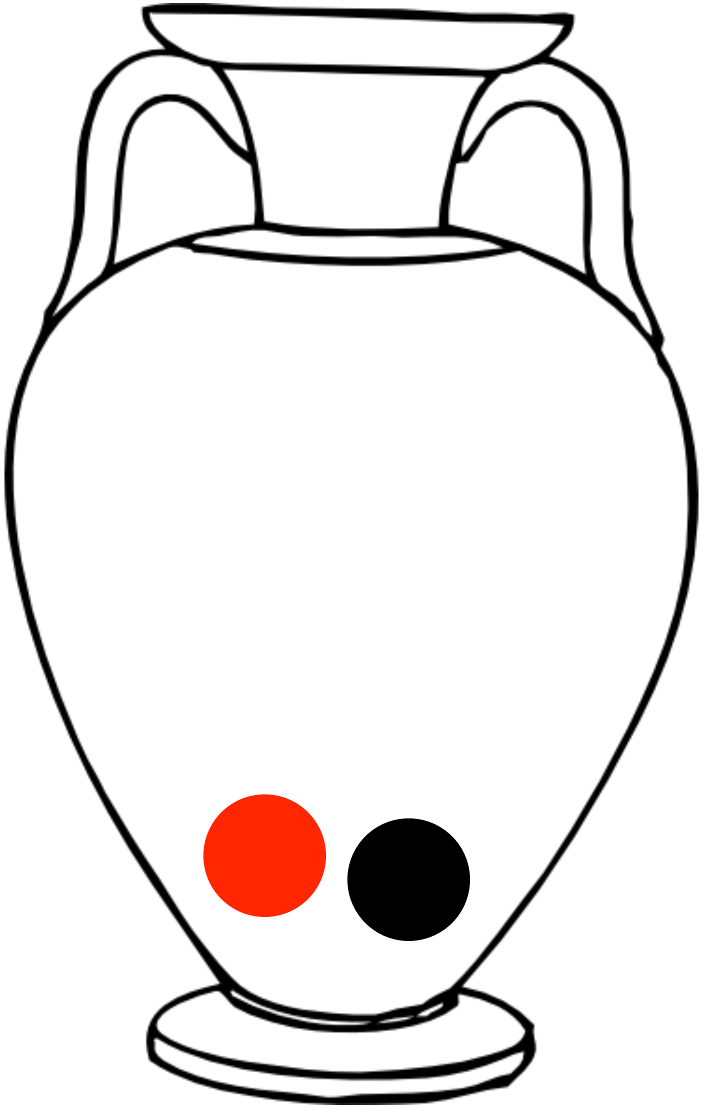


Add a black (special color) ball.

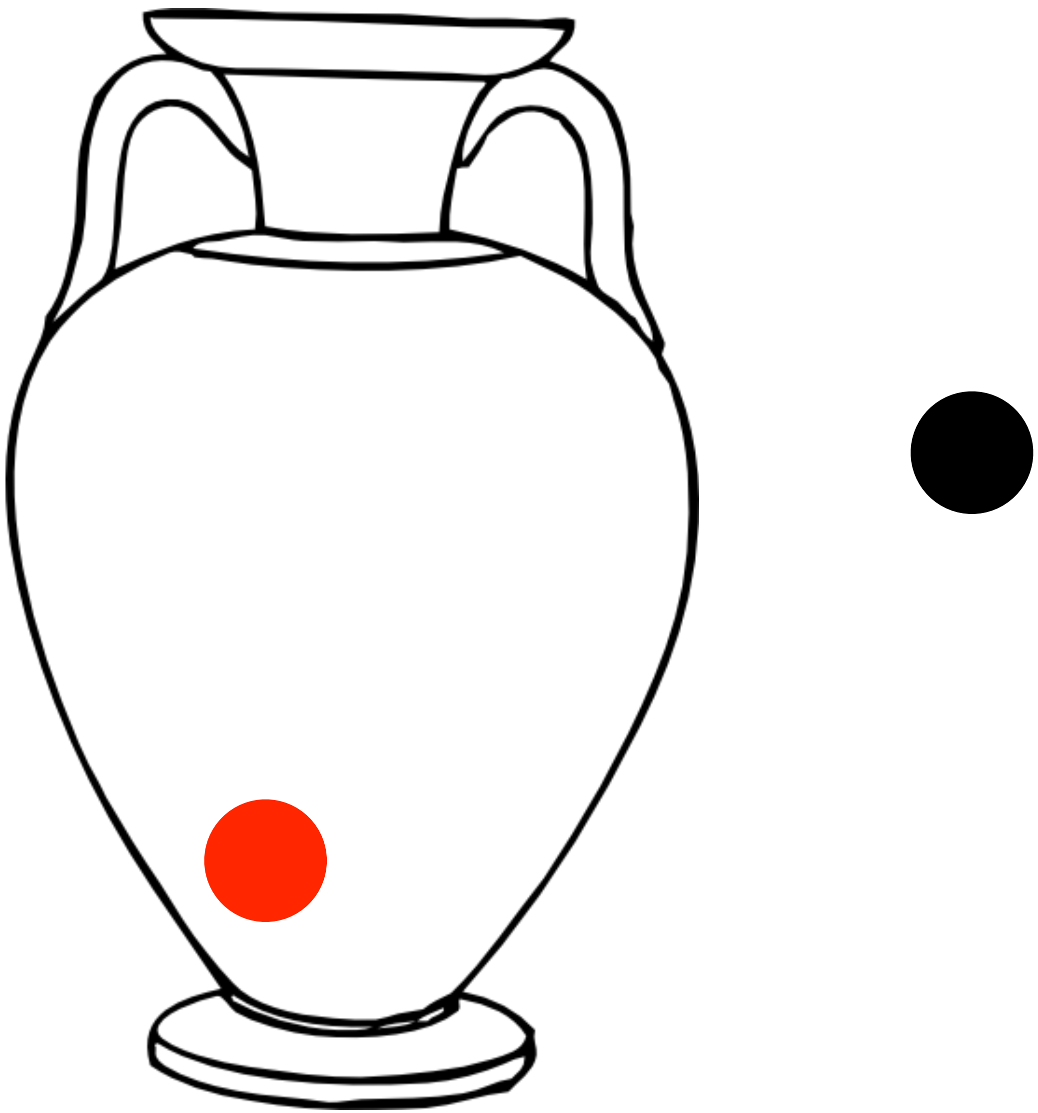
So:
black ball -> draw another ball, change its color, return both balls to the urn.
non-black ball -> return it to the urn along with another ball of the same color.

Notes:
The number of black balls never changes.
The number of iterations you need to reach N (non-black) balls in the urn is a random variable.
Colors can disappear.

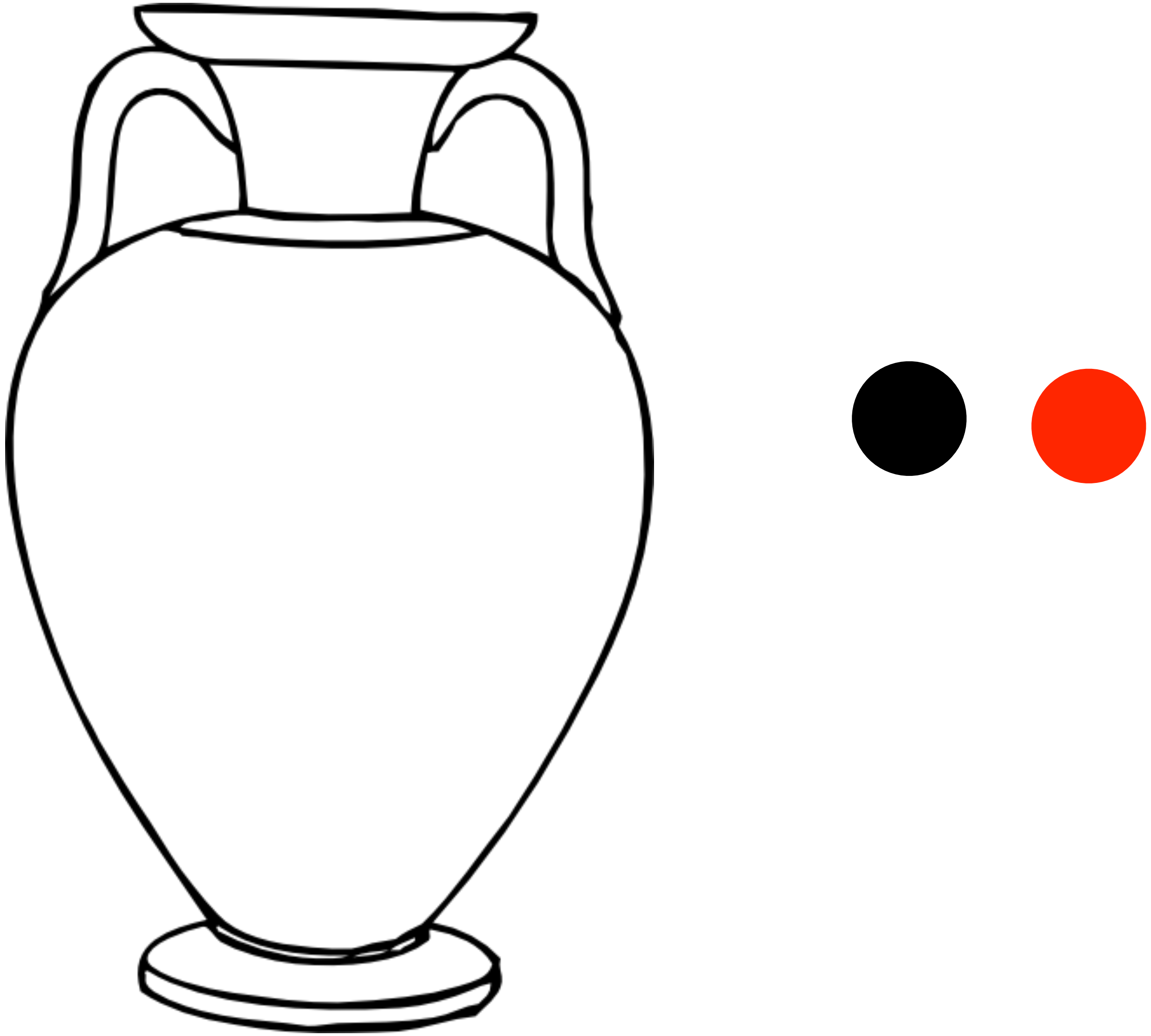
The Polya Urn version 2 - example



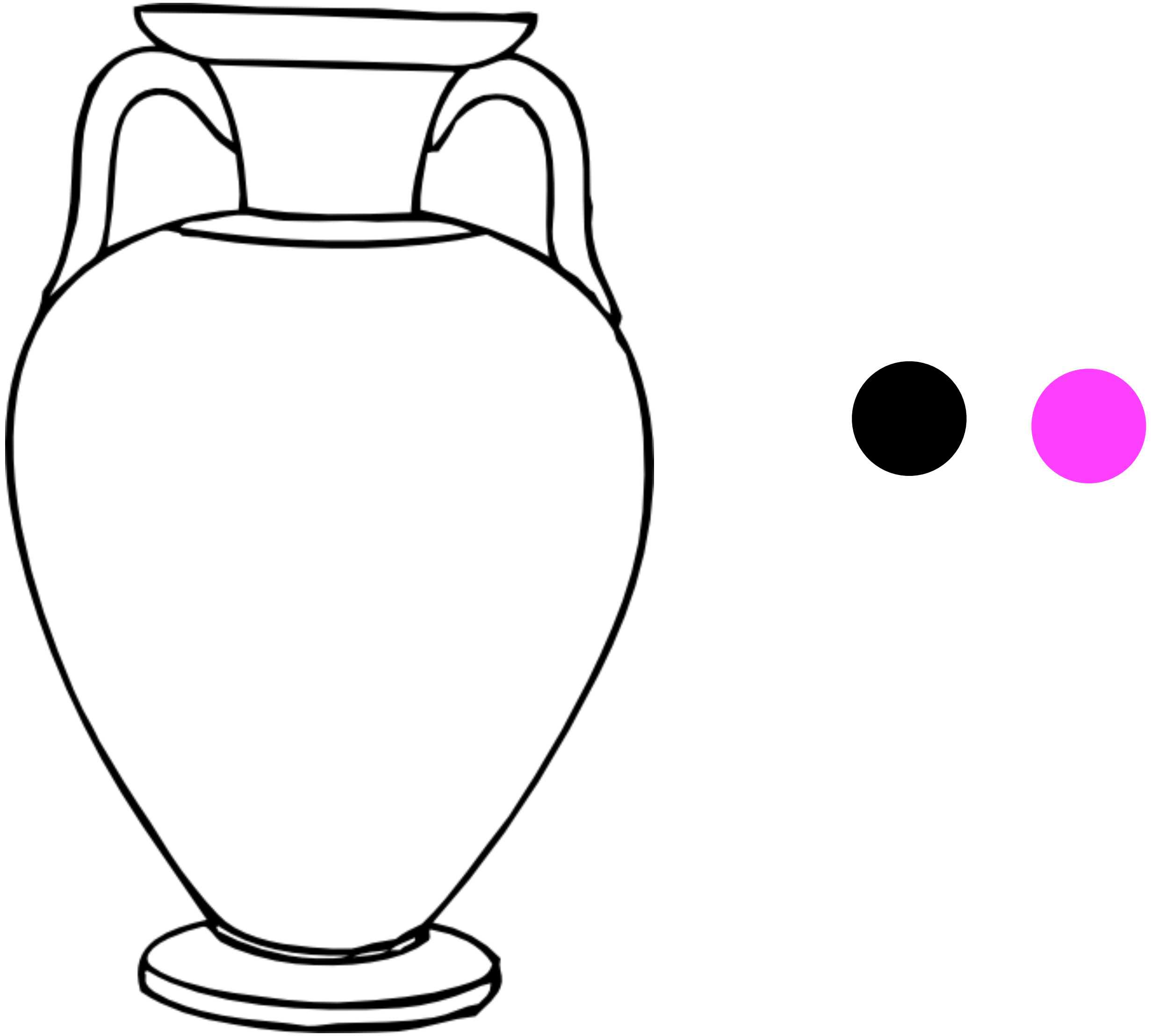
The Polya Urn version 2 - example



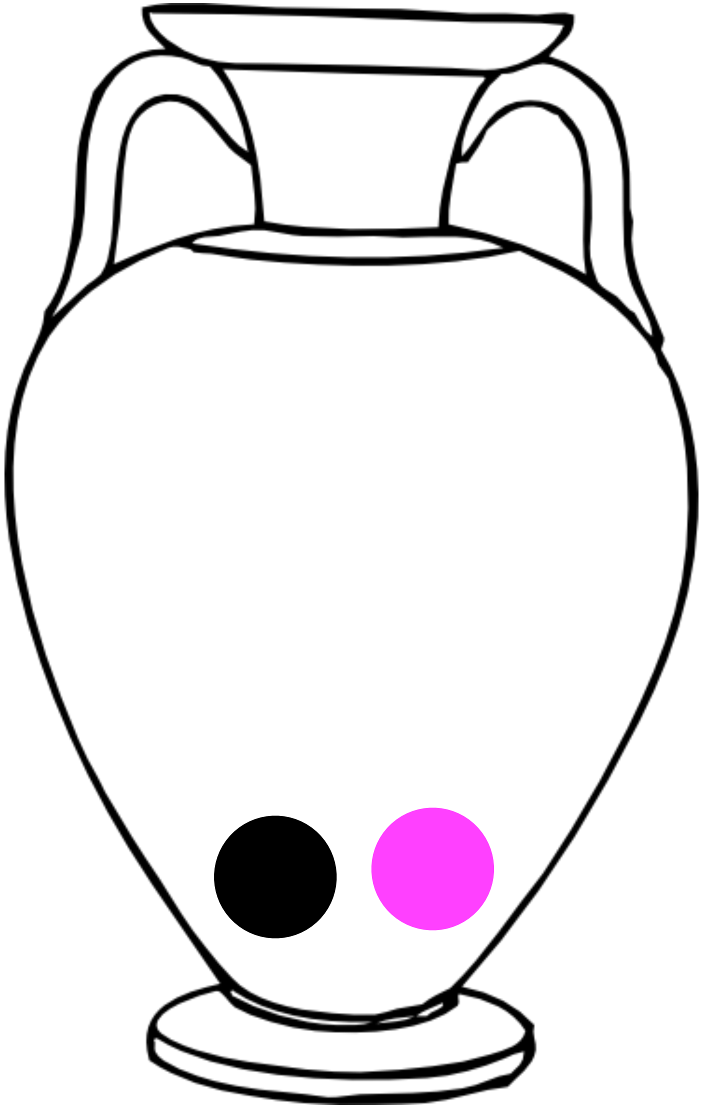
The Polya Urn version 2 - example



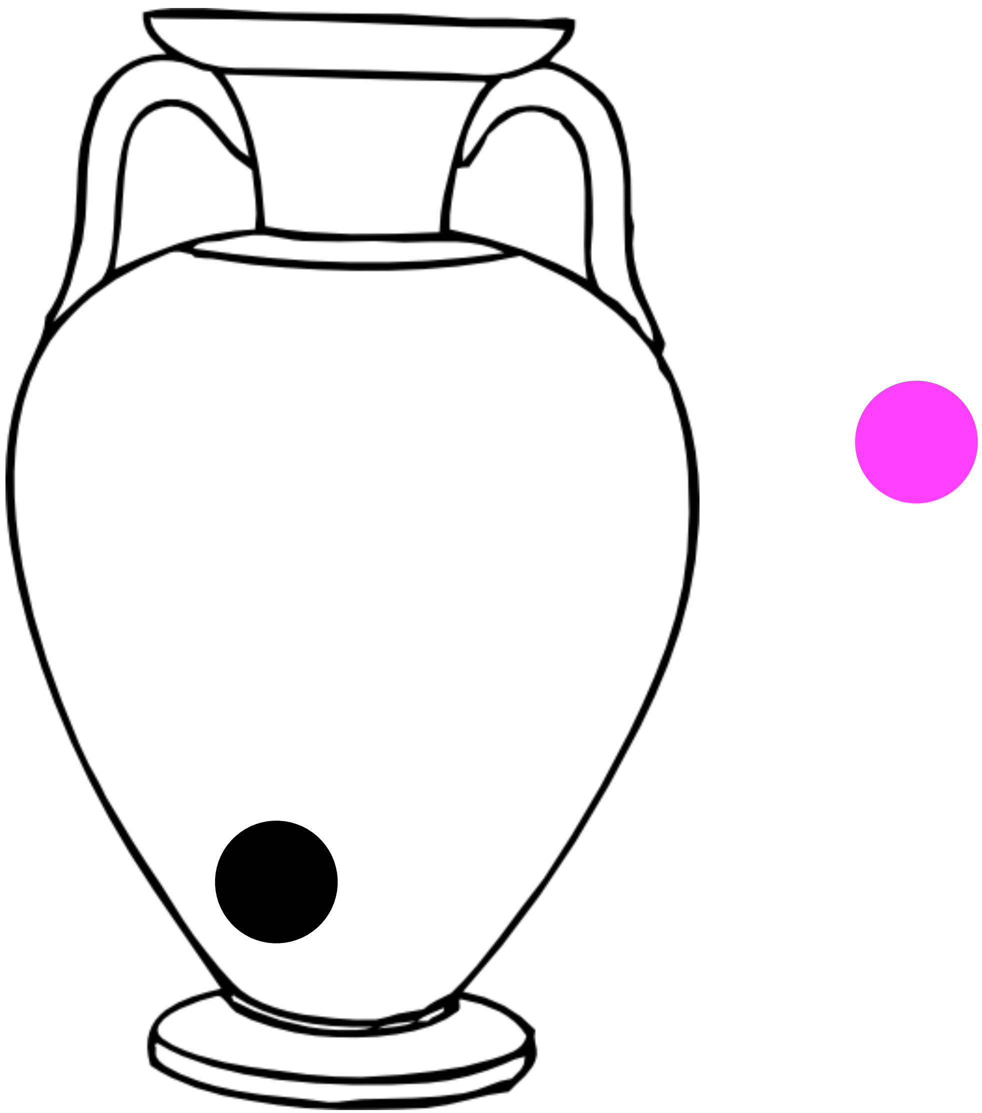
The Polya Urn version 2 - example



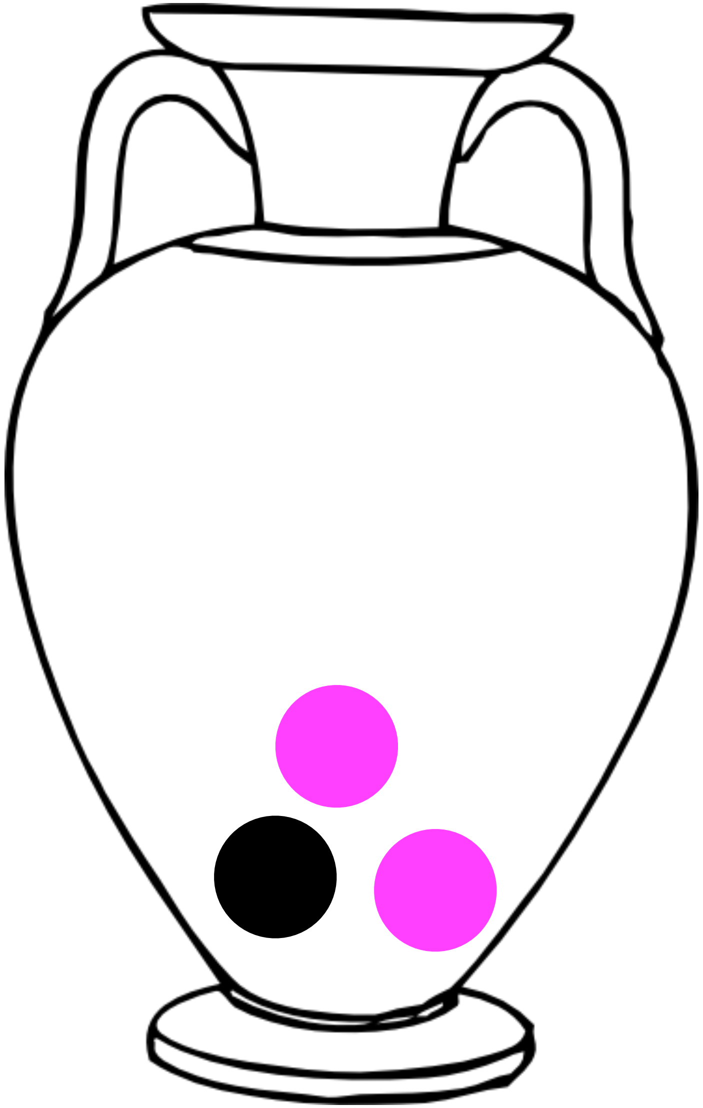
The Polya Urn version 2 - example



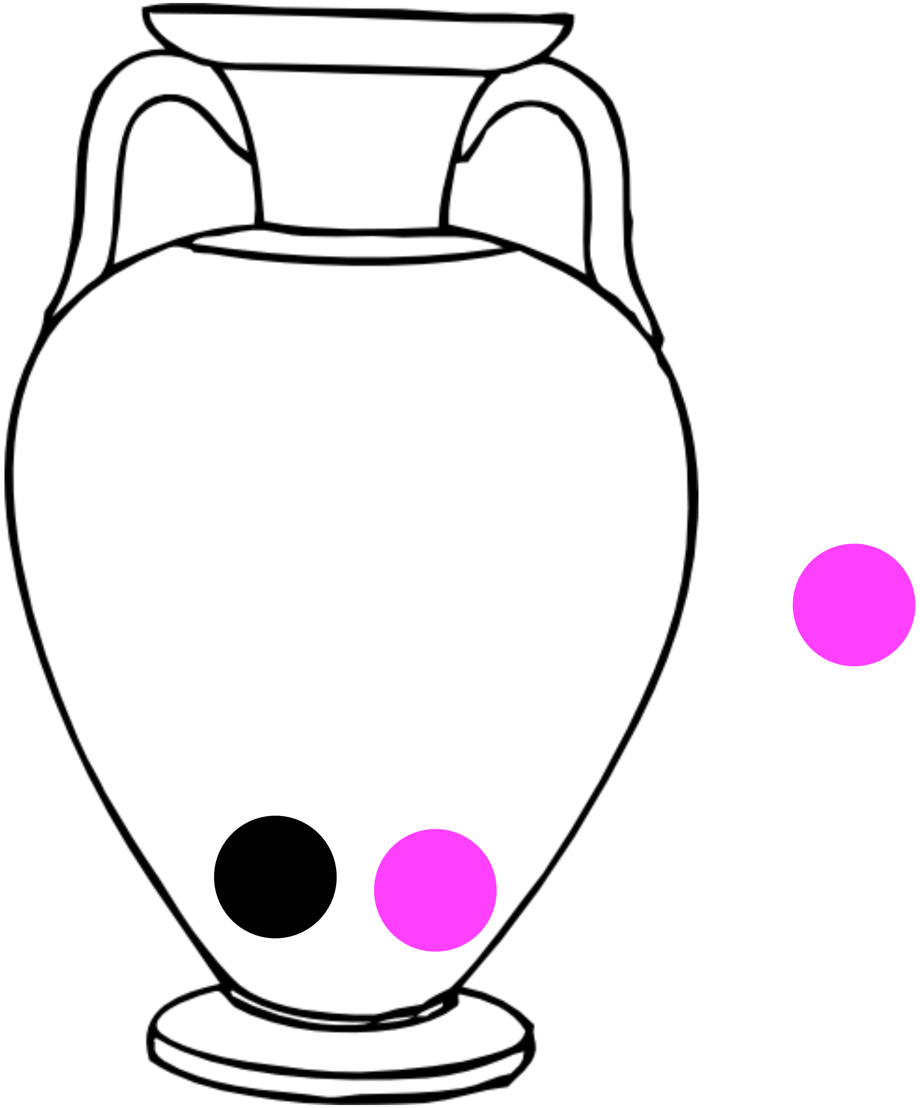
The Polya Urn version 2 - example



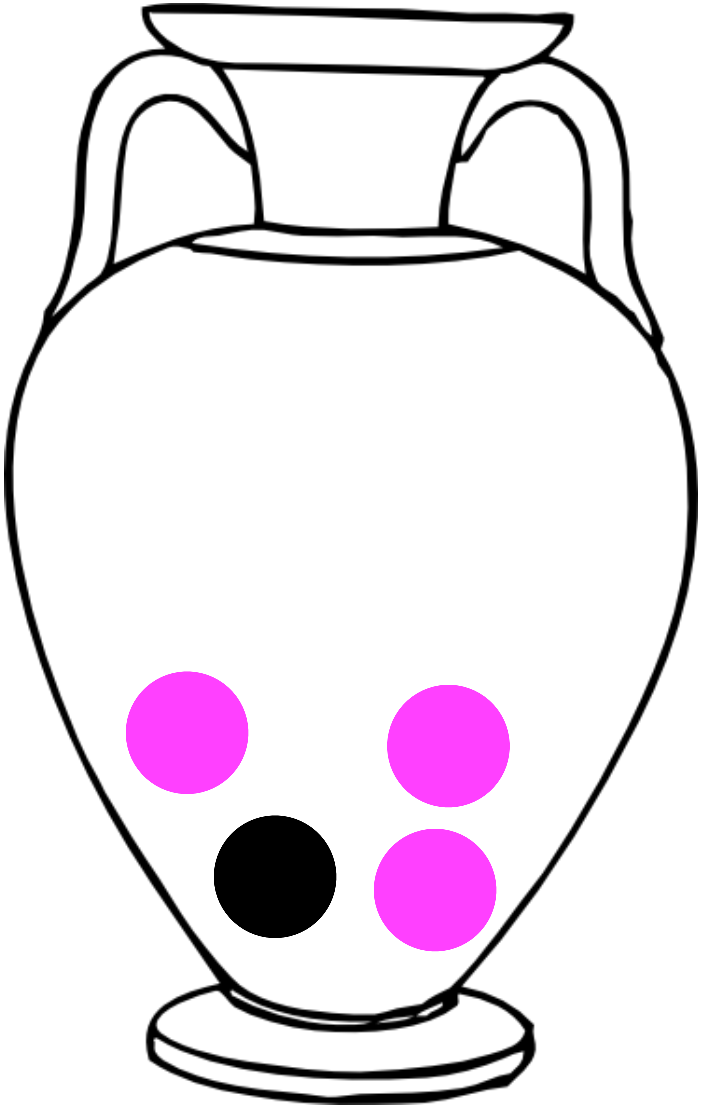
The Polya Urn version 2 - example



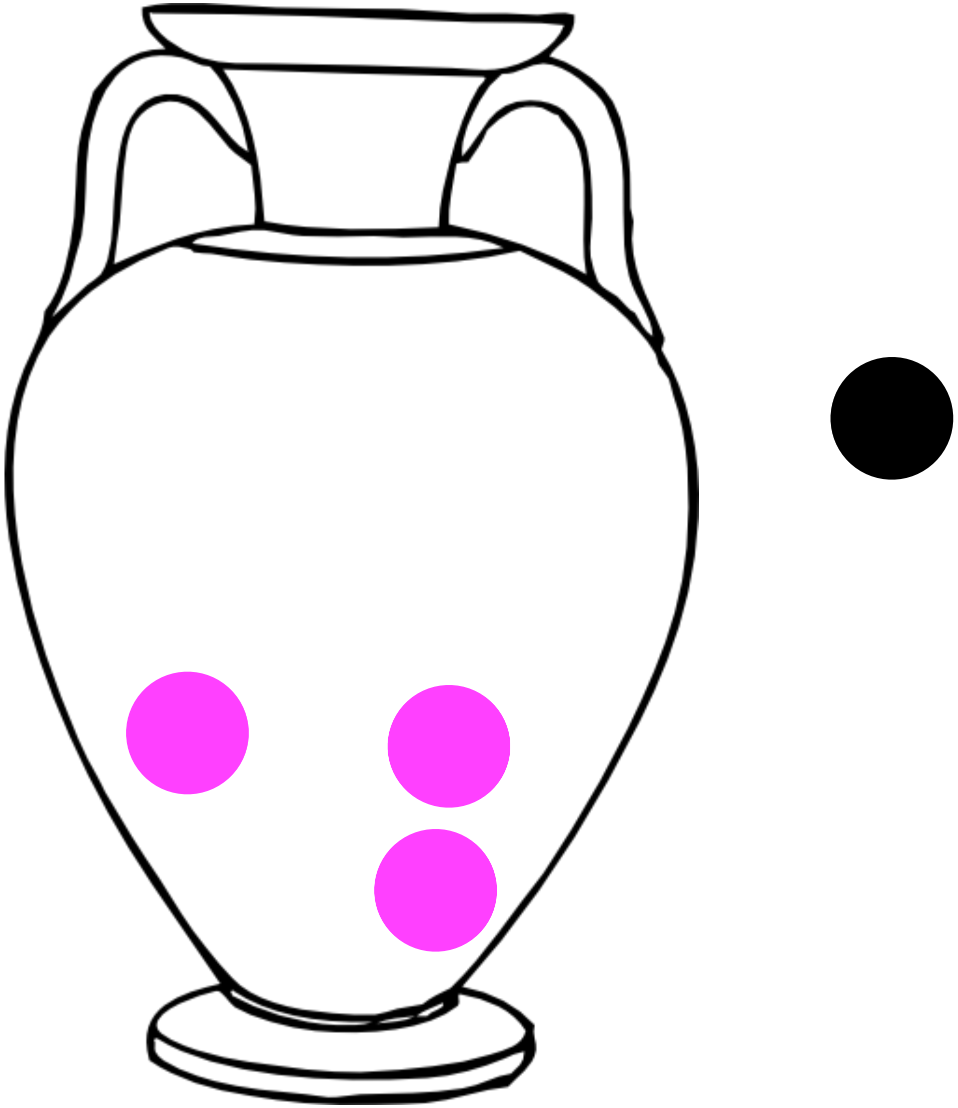
The Polya Urn version 2 - example



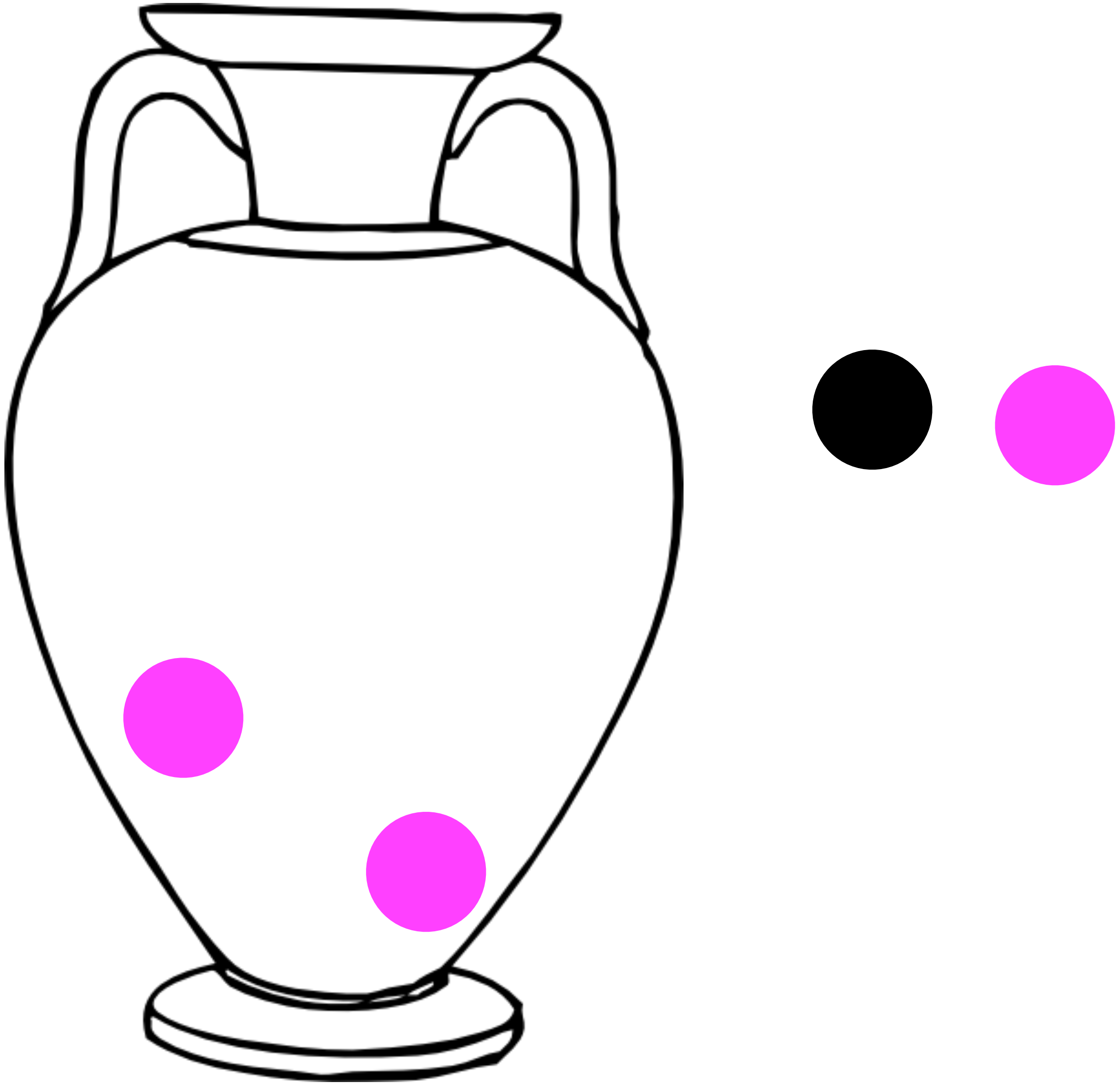
The Polya Urn version 2 - example



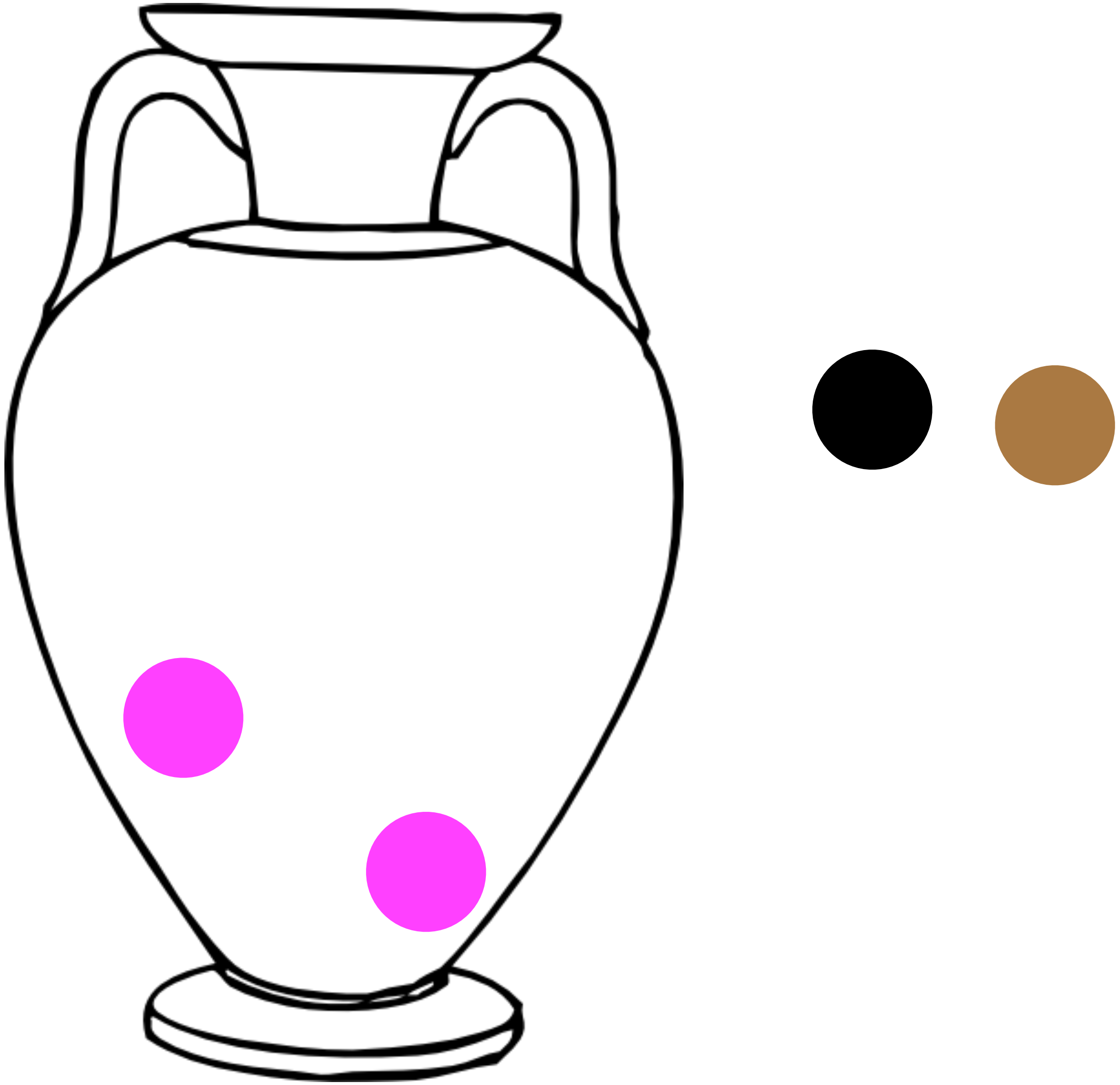
The Polya Urn version 2 - example



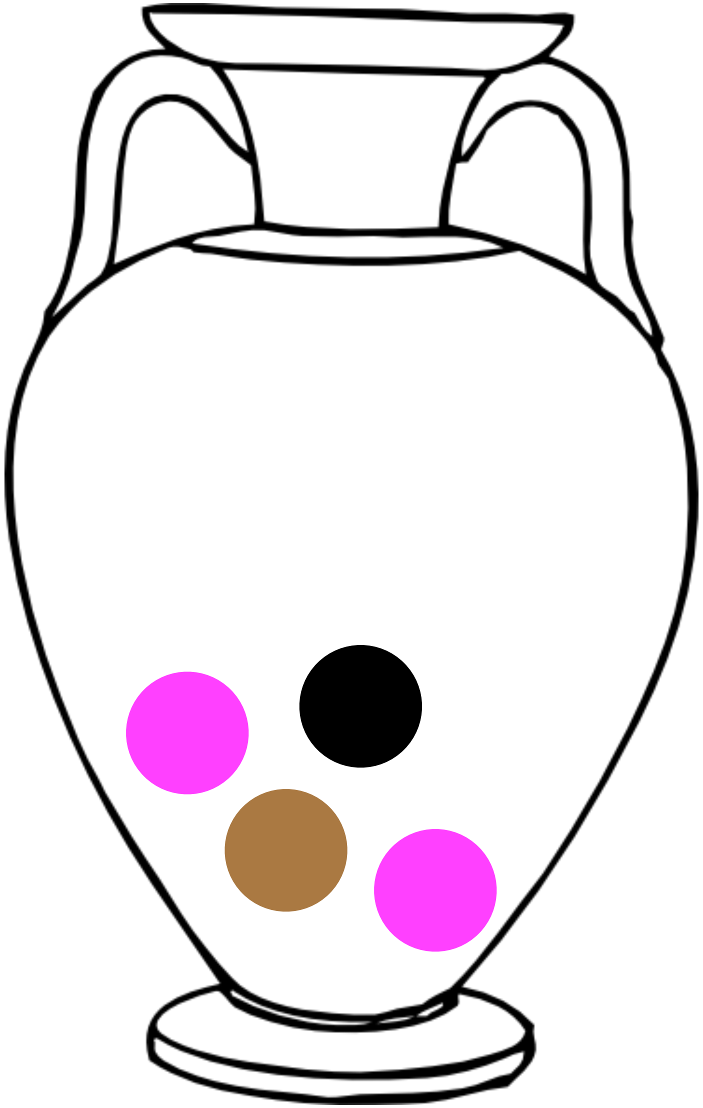
The Polya Urn version 2 - example



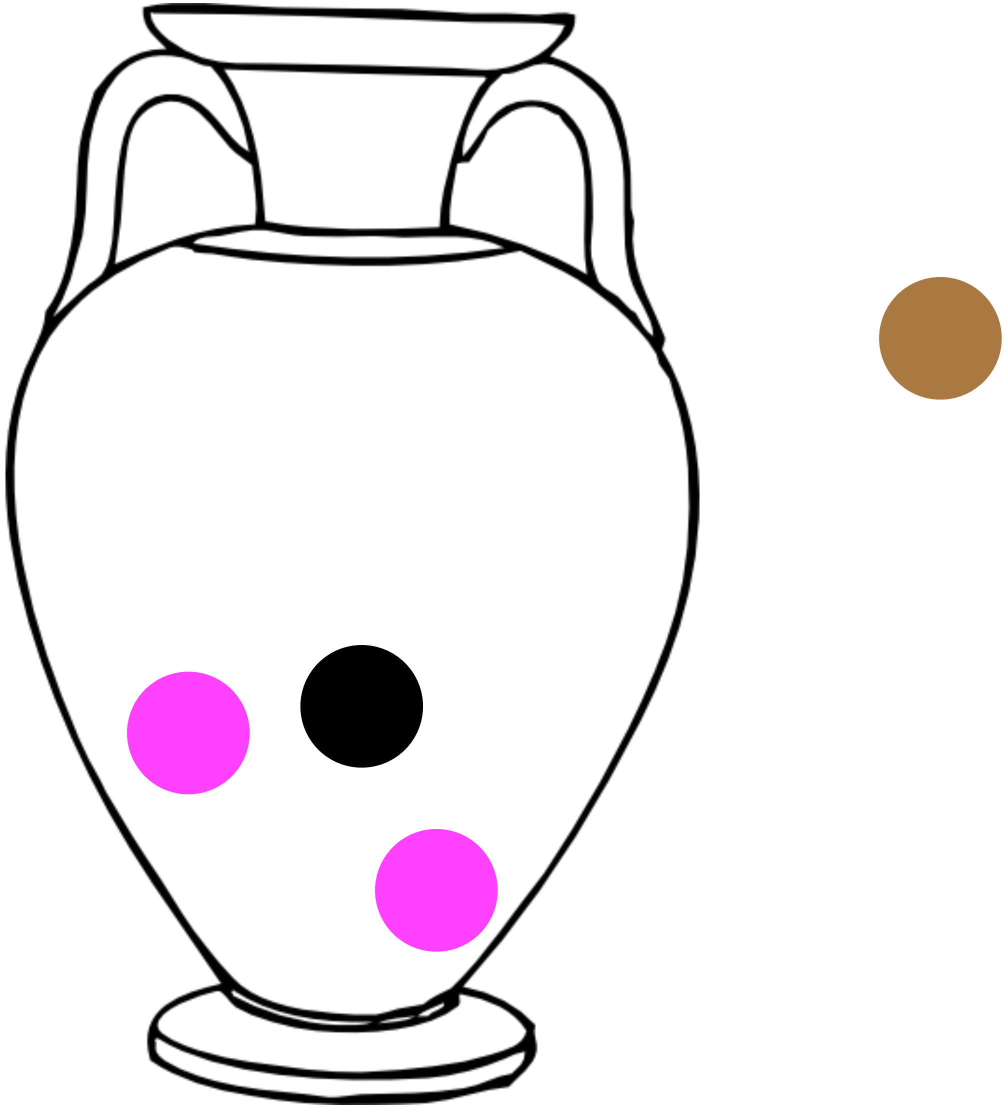
The Polya Urn version 2 - example



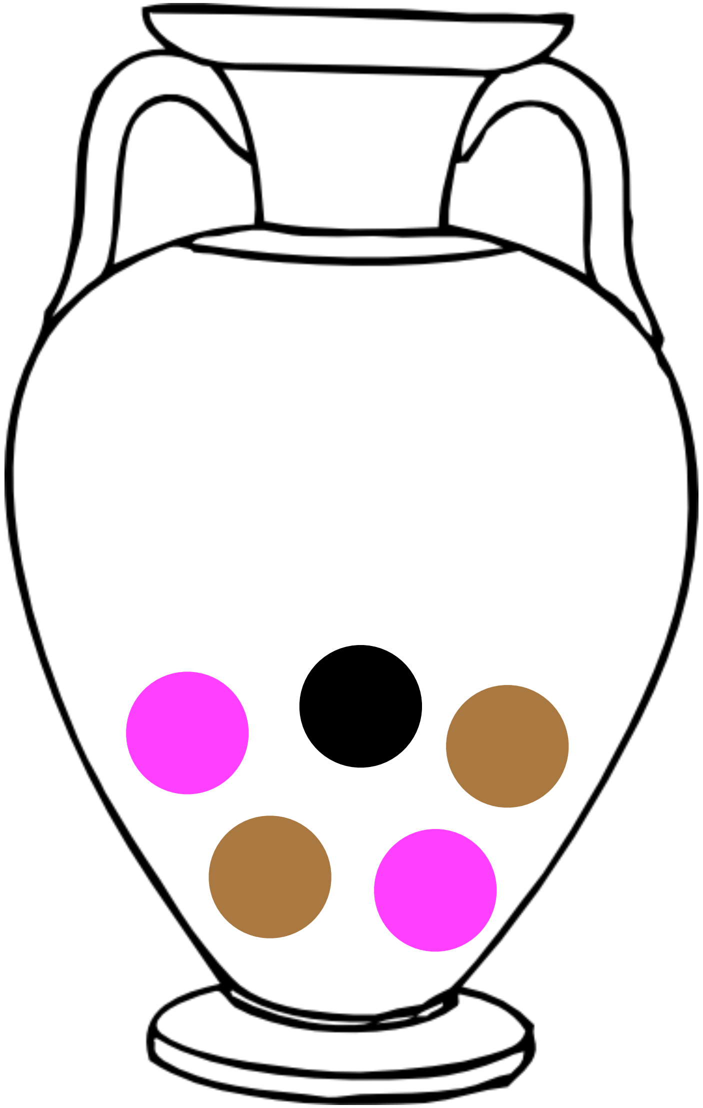
The Polya Urn version 2 - example



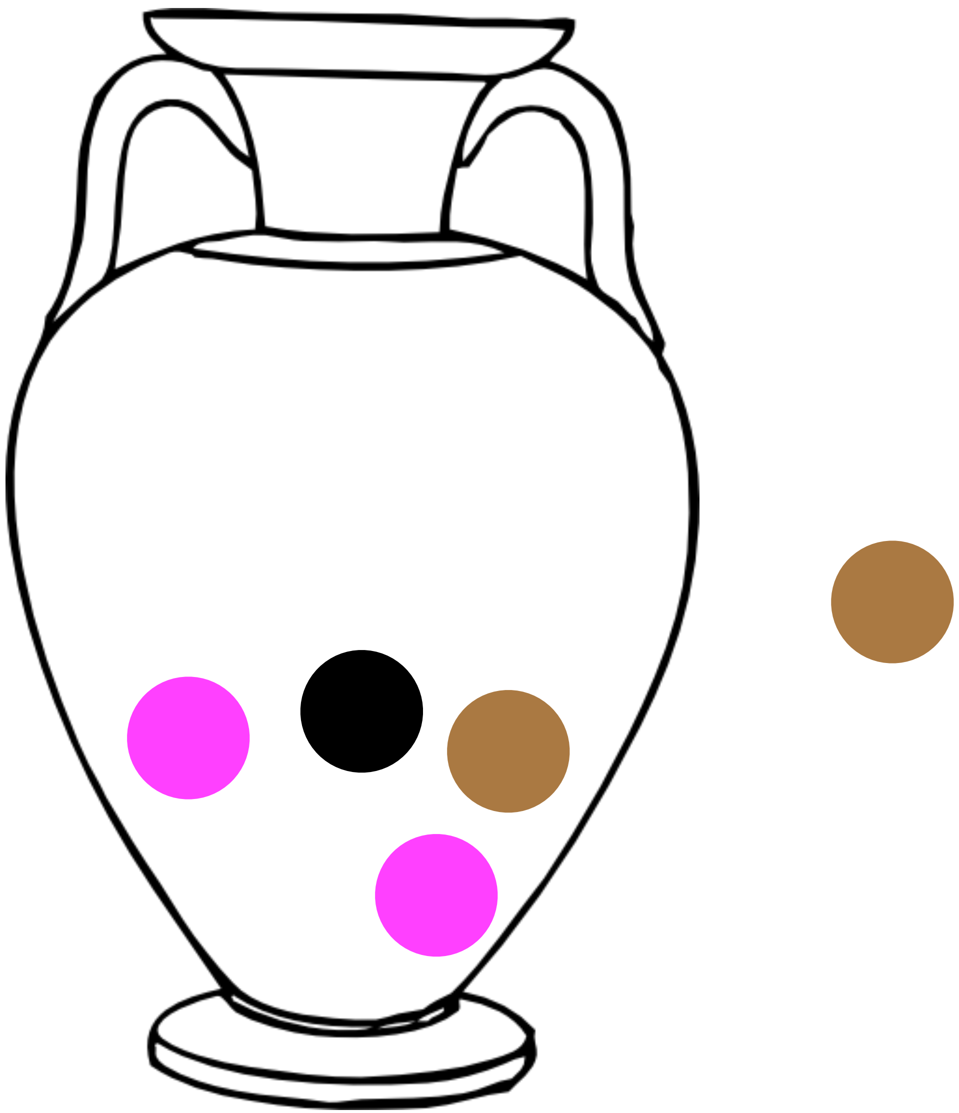
The Polya Urn version 2 - example



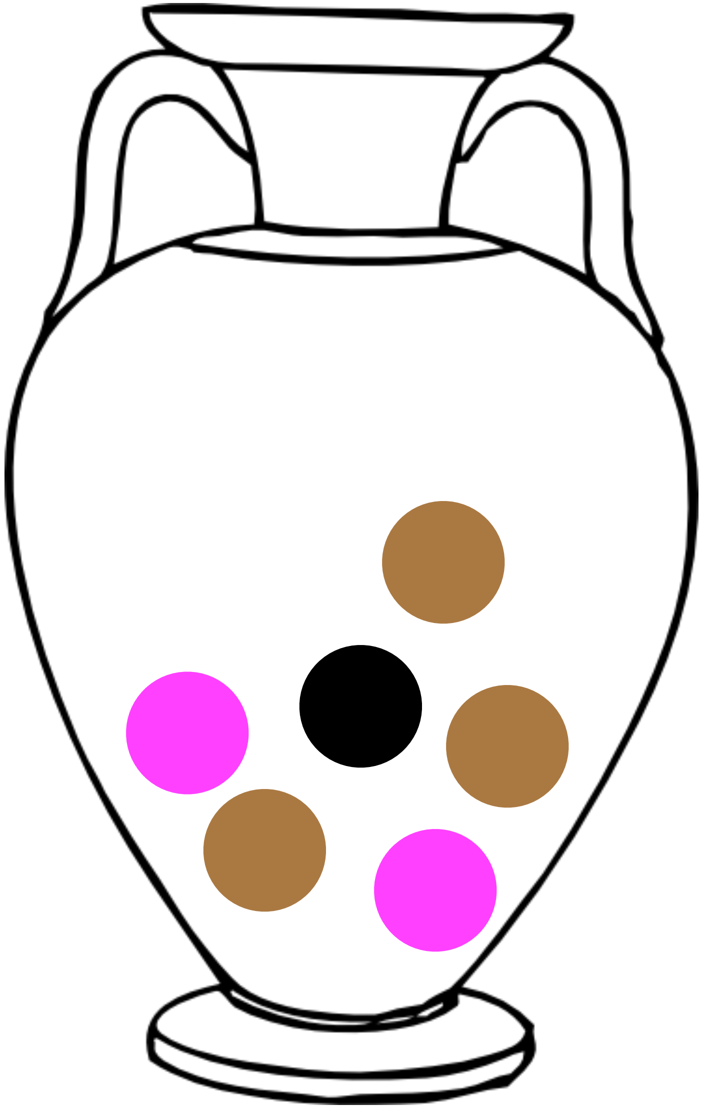
The Polya Urn version 2 - example



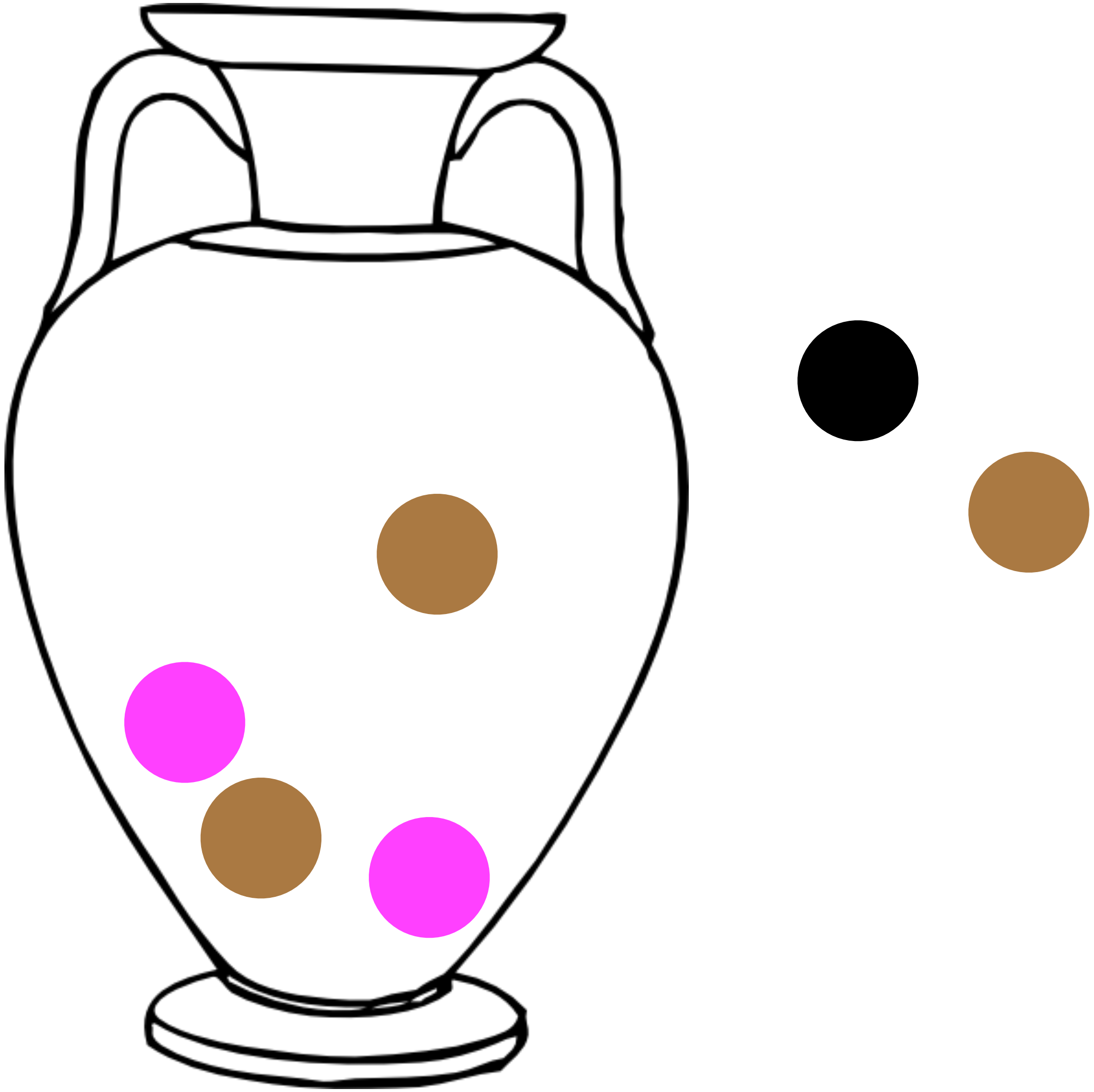
The Polya Urn version 2 - example



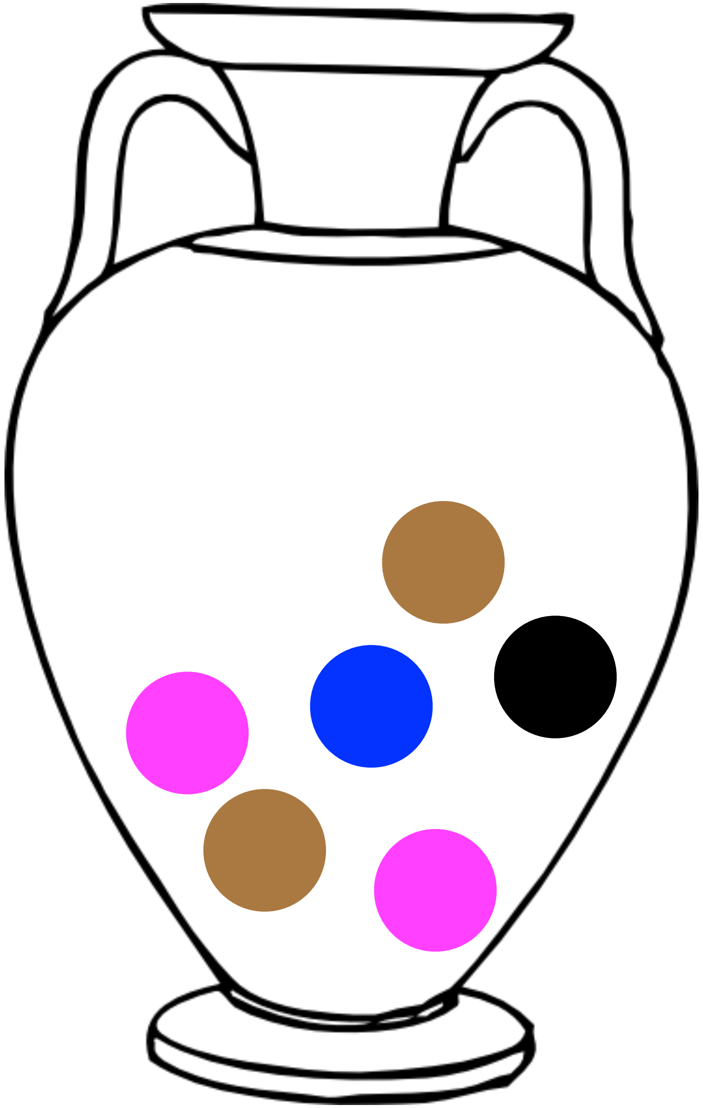
The Polya Urn version 2 - example



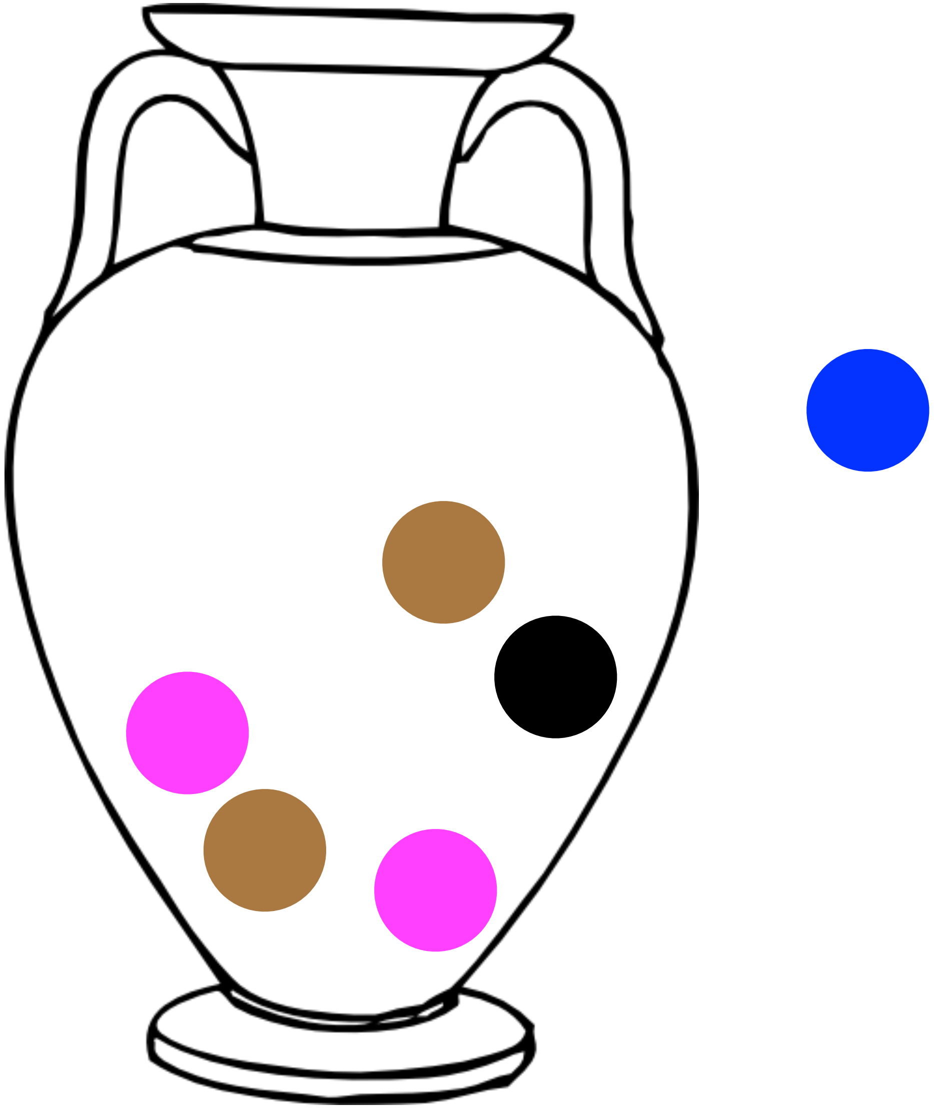
The Polya Urn version 2 - example



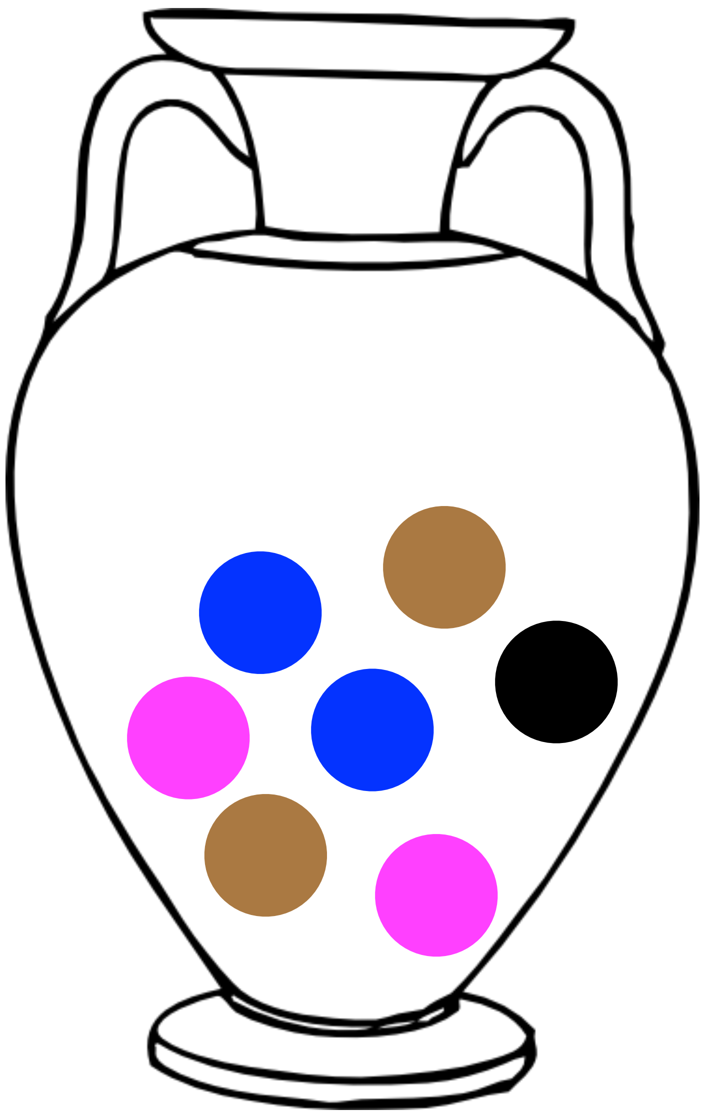
The Polya Urn version 2 - example



The Polya Urn version 2 - example



The Polya Urn version 2 - example



If we want to generate a sample of N non-black balls, we will do it two, slightly different ways:

1. Stop as soon as you have N non-black balls.

In your R-script, how are you going to represent the balls?

Pseudo-code

```
# Build it on your existing code (define variables, etc.)
# change the function for drawing a ball...
# notate colors with numbers, so black==0, other colors== 1,2,3...

# NumberOfColorsUsed<-NumberOfColorsInUrnAtStart
# while NumberOfBalls<(HowManyBallsWeNeed+1){
# draw a ball
# if (Ball is black){
        # pick another ball
        # change color of ball to new color (i.e. new number)
        NumberOfColorsUsed<-NumberOfColorsUsed+1
        # Set color of ball equal to  NumberOfColorsUsed
        # note that we don't increase the count of the number of balls
}else{
        # the ball is some other color
        # return the ball and add another one like it
        # increase the counter of how many balls we have in the urn:
        # NumberOfBalls<-NumberOfBalls+1
#}
# output summaries of what is in the urn when we are done
```

Lab Task #4 - The Polya Urn

- Code up this version of the Urn model. Then try the following:
- Start with one ‘red’ ball and the black ball (but code the colors as numbers, for simplicity’s sake, so Ball1 <- 1, Ball2 <- 2, say).
- Repeat until you have 50 non-black balls; replicate this process multiple times.

1. What is the distribution of the number of (non-black) colors at the end?
2. What is the distribution of the number of balls of the commonest color at the end?
3. How many replicates do you need to do to answer these questions? How did you decide upon this number?
4. Repeat 1. and 2., but start with two balls of a same color+black ball each time. Compare your results to those from 1. and 2. Can you explain what you see?

Lab Task 5 - The Generalized Polya Urn (note: you will need this version later in the course, so it better work!)

- Code up a further generalized version of the Polya Urn model
  - Imagine that each color has a 'weight'
  - When we draw a ball, a given ball is chosen with probability equal to (Weight of that ball)/(total weight of all balls in the Urn).
  - [So, our previous version had implicit weights of 1 for each color]
- Start with two 'red' balls and the black ball (but, again, code the colors as numbers, for simplicity's sake).
- Repeat until you have 50 non-black balls; replicate this process multiple times.
- Assume all, non-black colors have weight 1
  1. What is the expected number of different (non-black) colors in the Urn at the end as a function of the weight of the black ball ?
  2. What is the distribution of the number of (non-black) colors at the end, as a function of the weight of the black ball? (Try weight=1, or 2, or, ... , or 10.

And if we have time…

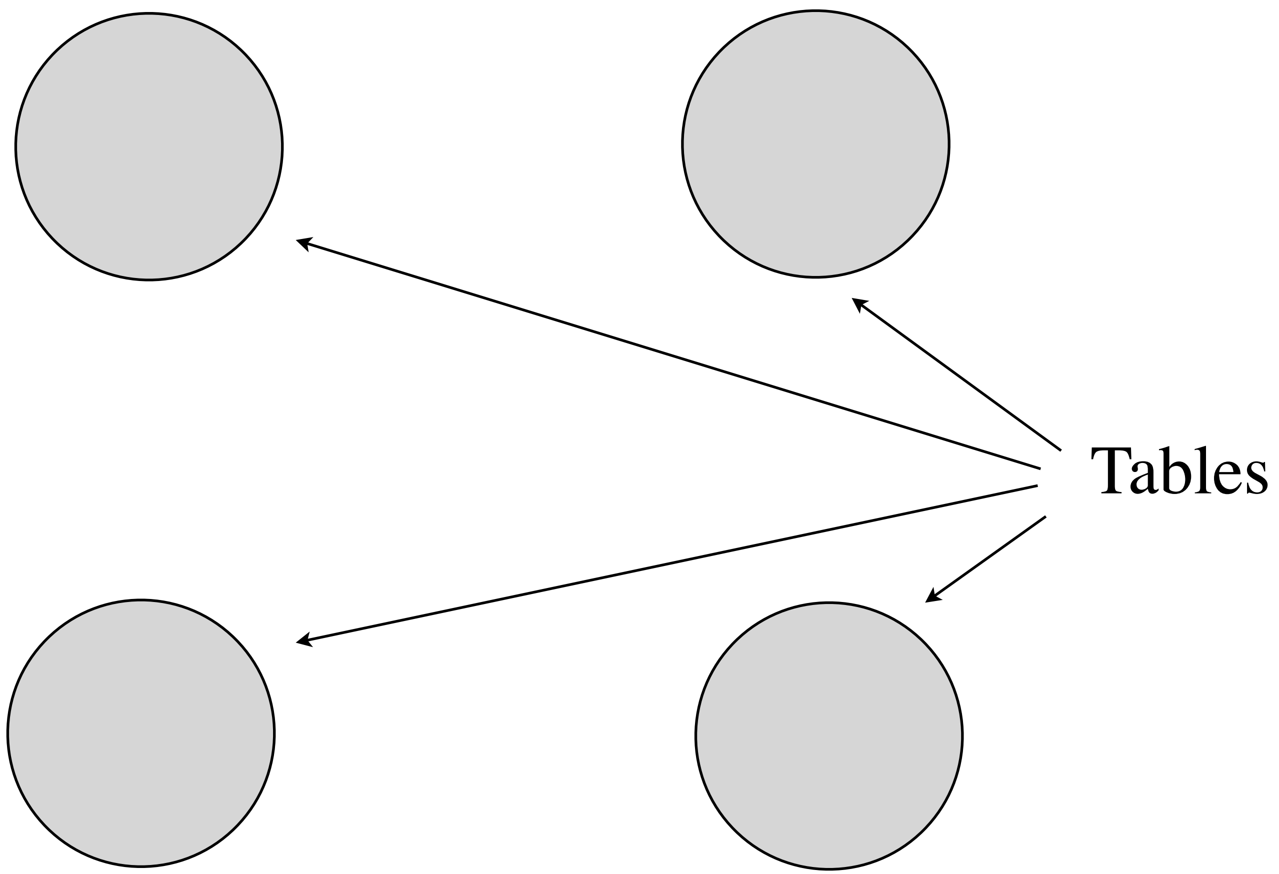
The Restaurant Process

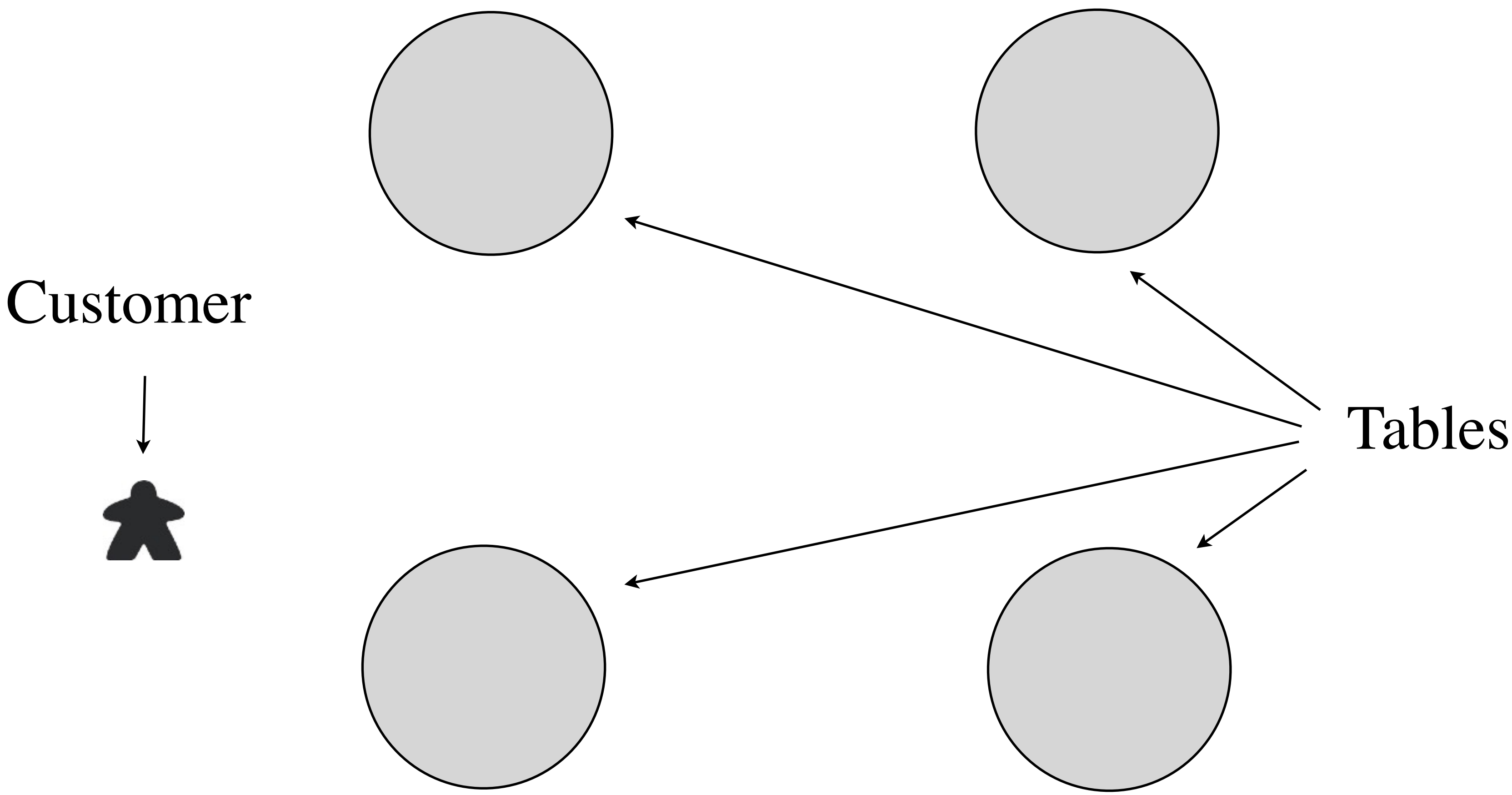
Customers arrive one by one in an empty restaurant which has an unlimited number of round tables.
Initially, customer 1 sits by herself.
When the $(n+1)^{th}$ customer arrives, she seats herself as follows:

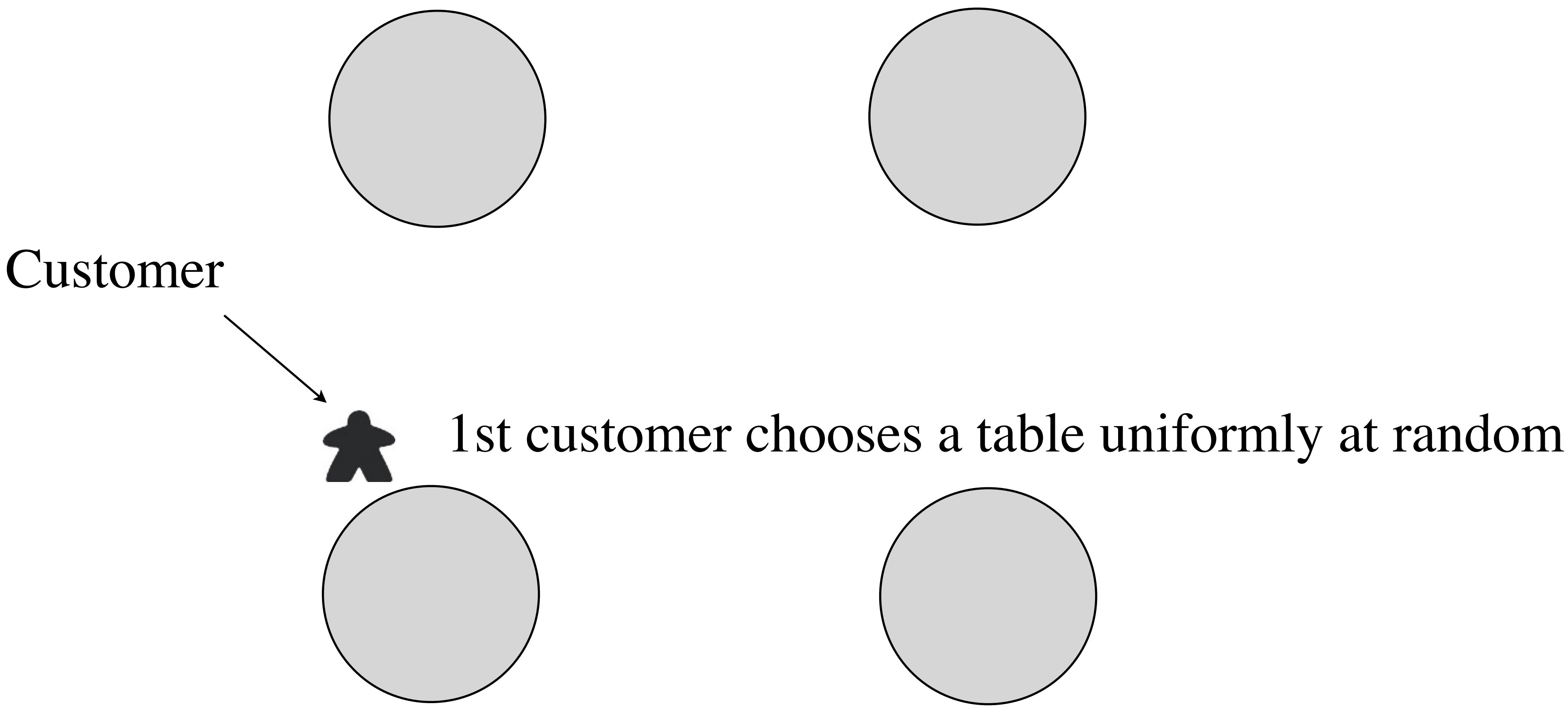
i) with probability $1/(n+1)$ she sits at a new table

ii) otherwise (i.e. with prob. $n/(n+1)$) she sits to the right of an existing diner; choosing to sit to the right of a **customer** chosen uniformly at random from among those already seated.

N.B. "customer", not "table"

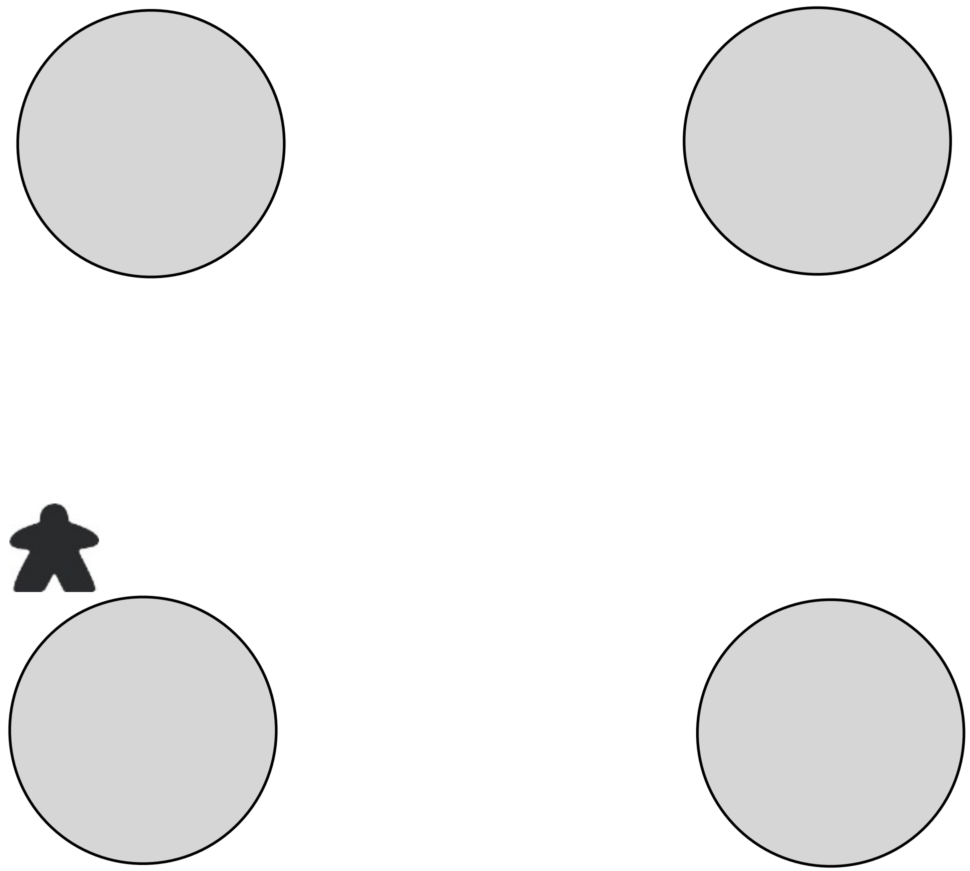

Tables


Customer
Tables


Customer
1st customer chooses a table uniformly at random

Customer n+1 does one of two things:

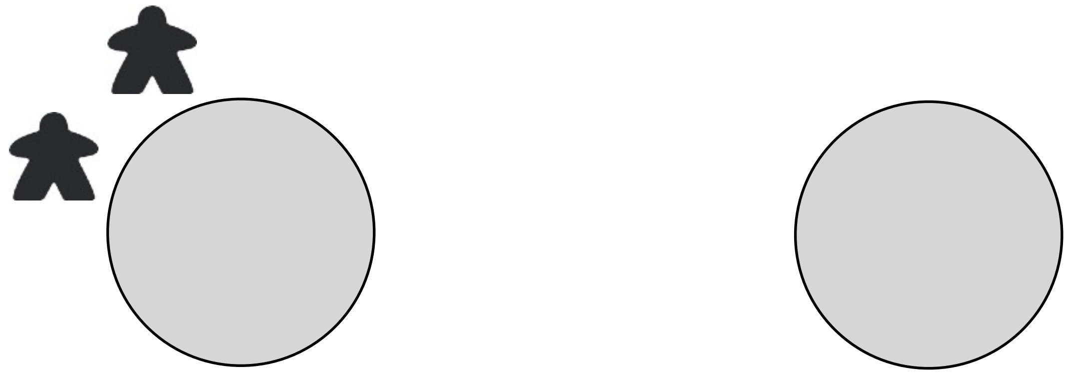
1. With probability 1/(n+1) she sits at an unoccupied table
2. Otherwise, she picks a person, uniformly at random, and sits at the same table, to their right.



Customer n+1 does one of two things:

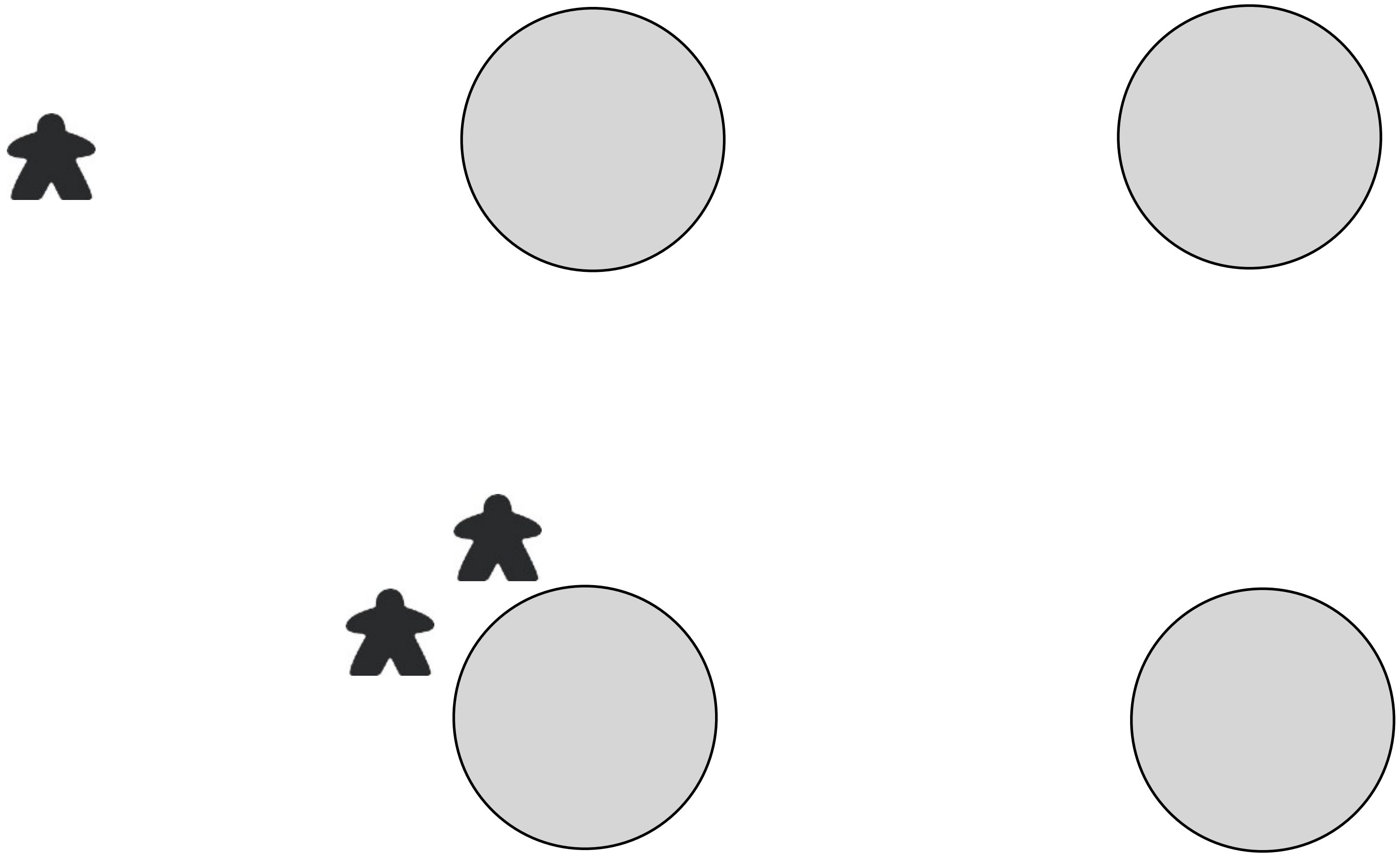
1. With probability 1/(n+1) she sits at an unoccupied table
2. Otherwise, she picks a person, uniformly at random, and sits at the same table, to their right.

Prob.= 1/2



Customer n+1 does one of two things:

1. With probability 1/(n+1) she sits at an unoccupied table
2. Otherwise, she picks a person, uniformly at random, and sits at the same table, to their right.



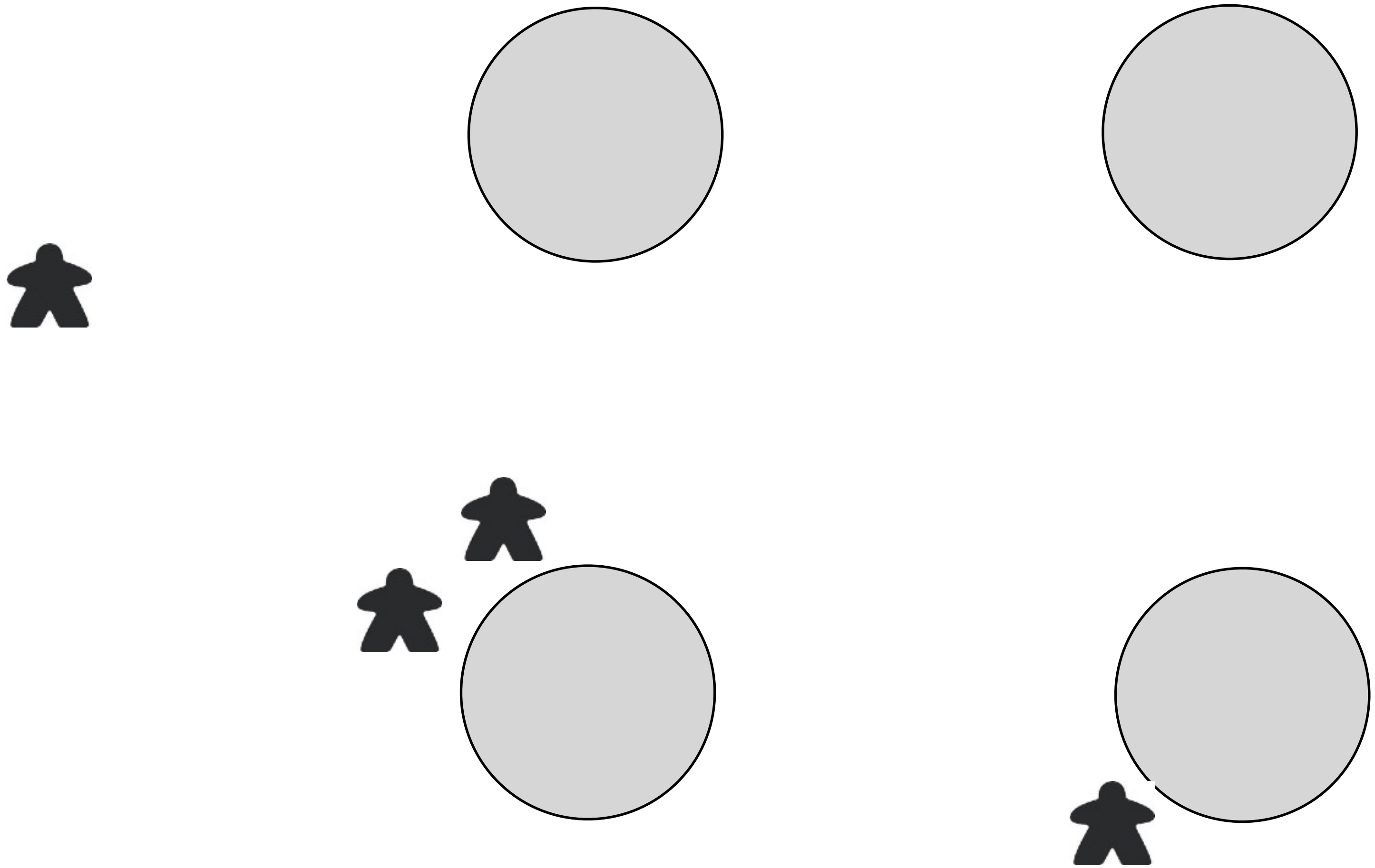
Customer n+1 does one of two things:

1. With probability 1/(n+1) she sits at an unoccupied table
2. Otherwise, she picks a person, uniformly at random, and sits at the same table, to their right.

Prob.= 1/3

Customer n+1 does one of two things:

1. With probability 1/(n+1) she sits at an unoccupied table
2. Otherwise, she picks a person, uniformly at random, and sits at the same table, to their right.

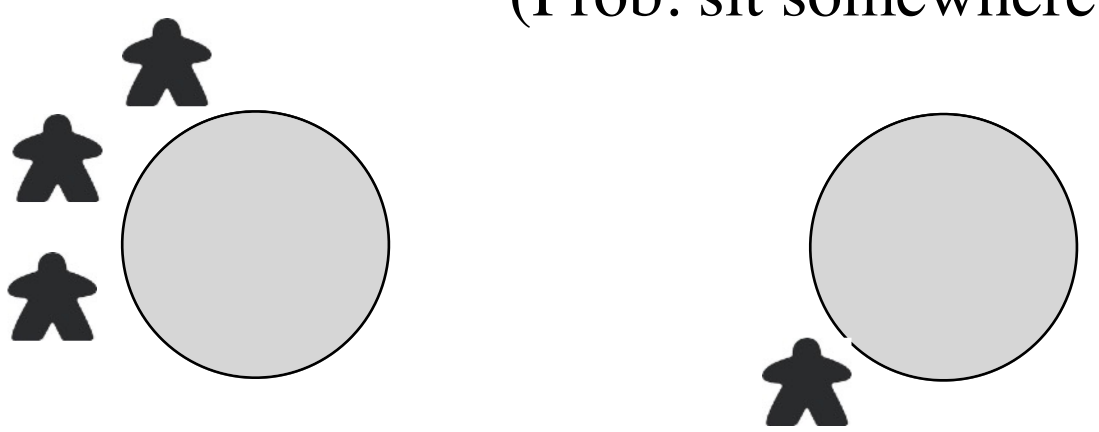


Customer n+1 does one of two things:

1. With probability 1/(n+1) she sits at an unoccupied table
2. Otherwise, she picks a person, uniformly at random, and sits at the same table, to their right.

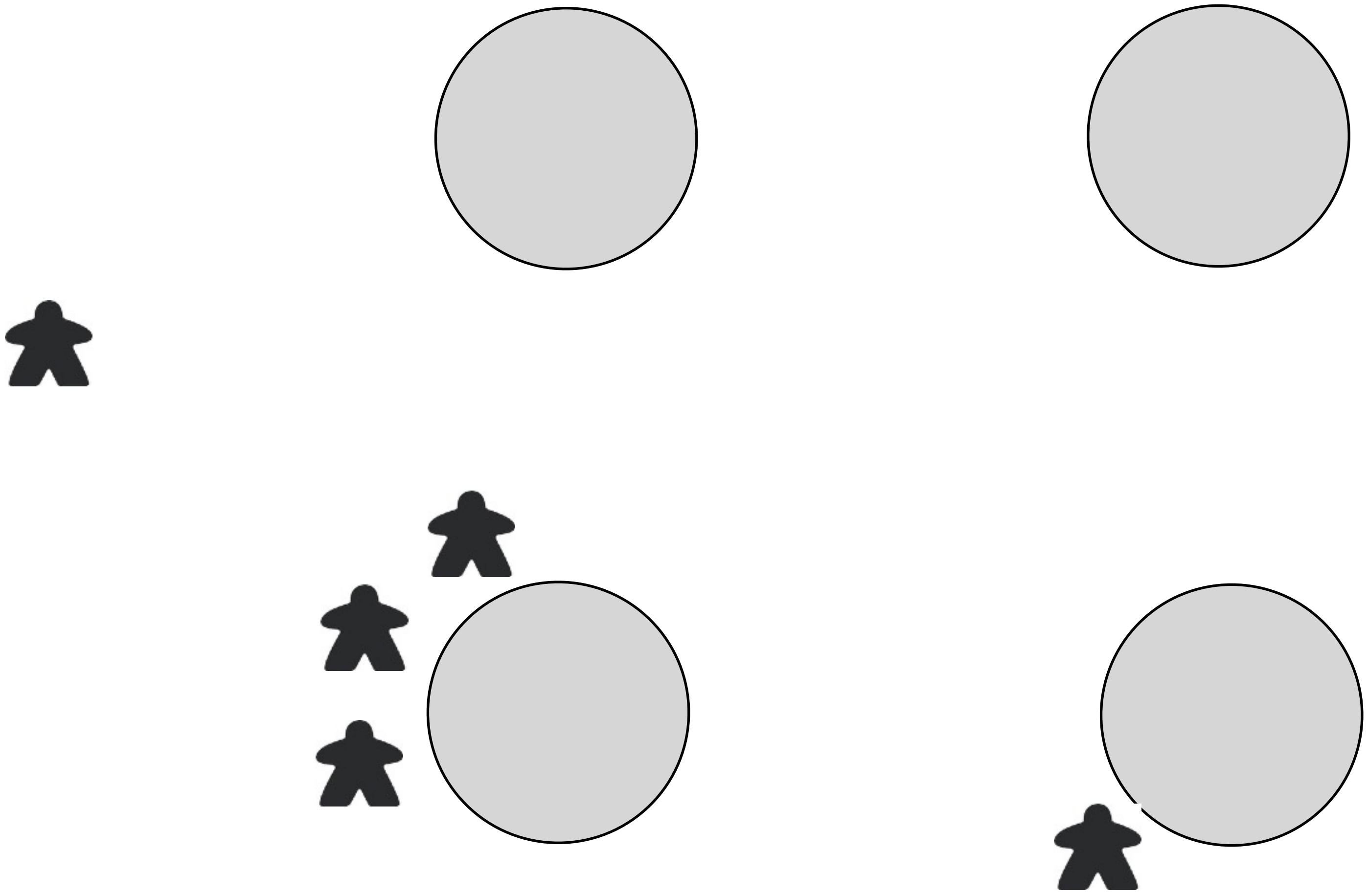
Prob.= 1/4

(Prob. sit somewhere at that table was 2/4)



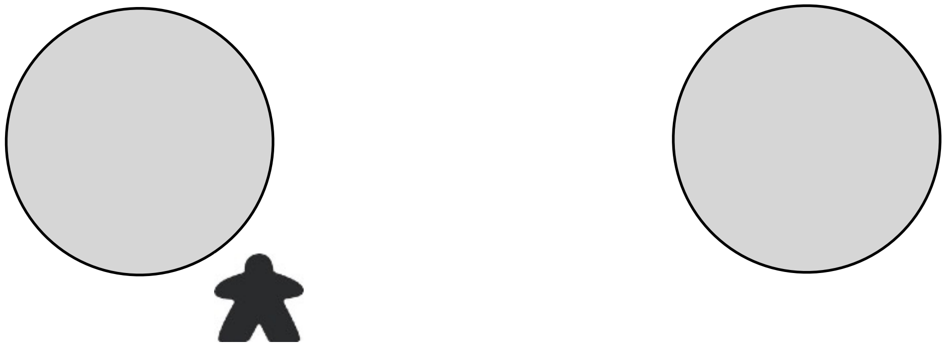
Customer n+1 does one of two things:

1. With probability 1/(n+1) she sits at an unoccupied table
2. Otherwise, she picks a person, uniformly at random, and sits at the same table, to their right.



Customer n+1 does one of two things:

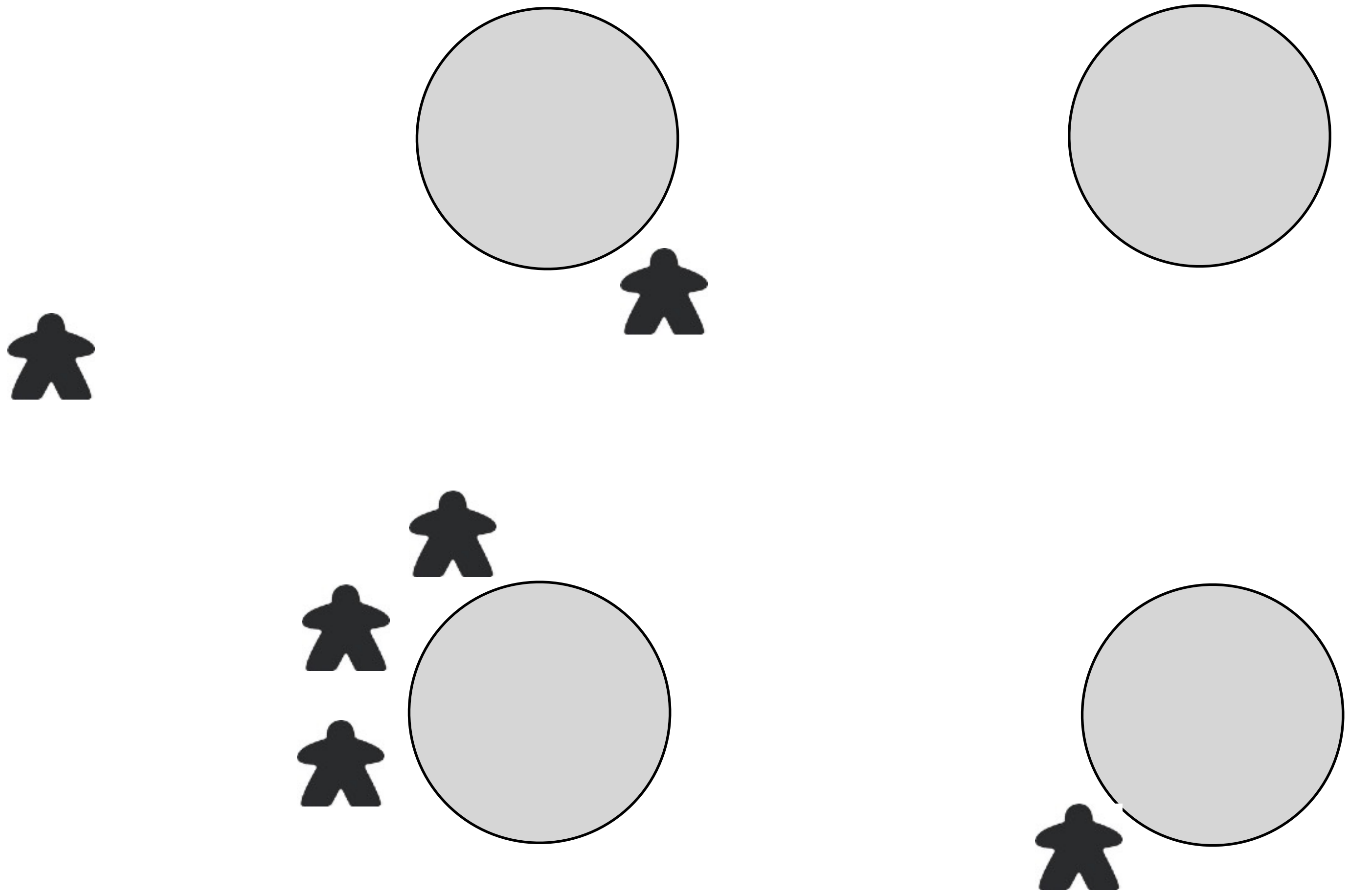
1. With probability 1/(n+1) she sits at an unoccupied table
2. Otherwise, she picks a person, uniformly at random, and sits at the same table, to their right.



Prob.= 1/5

Customer n+1 does one of two things:

1. With probability 1/(n+1) she sits at an unoccupied table
2. Otherwise, she picks a person, uniformly at random, and sits at the same table, to their right.



Customer n+1 does one of two things:

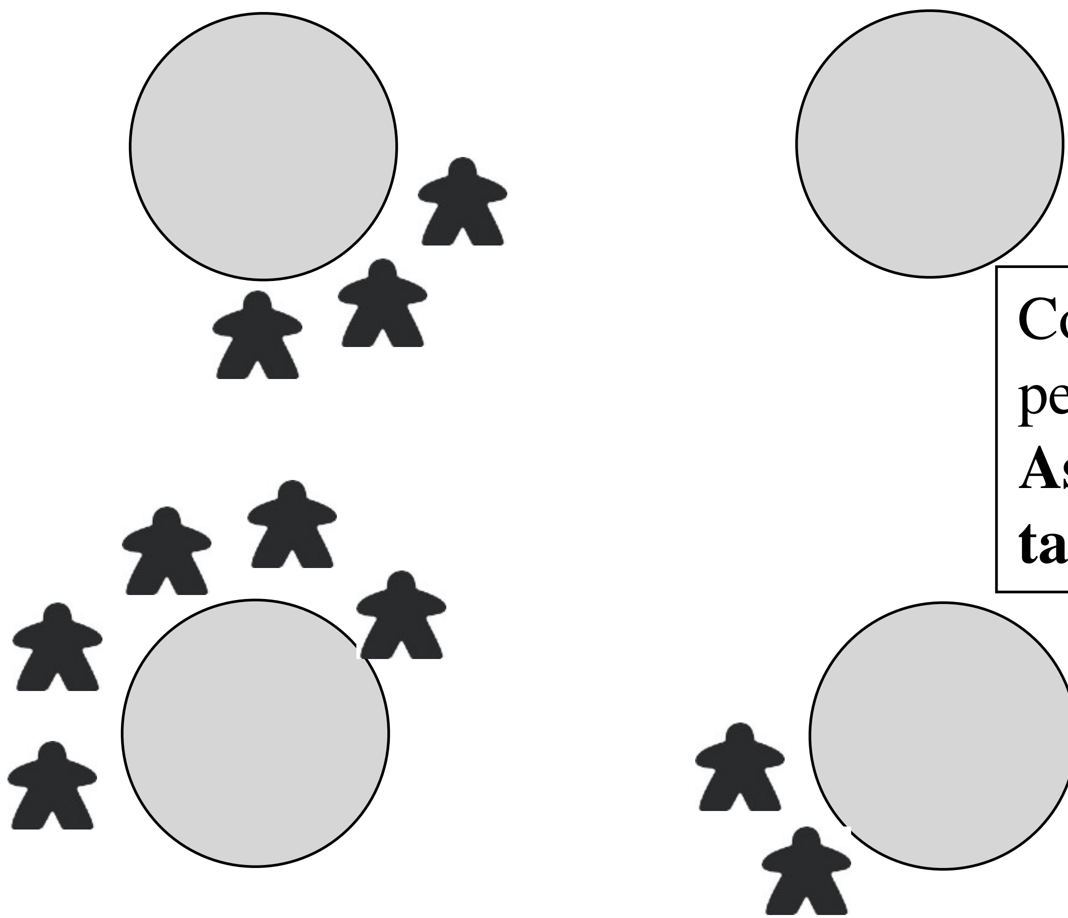
1. With probability 1/(n+1) she sits at an unoccupied table
2. Otherwise, she picks a person, uniformly at random, and sits at the same table, to their right.

Prob.= 1/6

Prob. sit at that table (anywhere) = 1/6

Customer n+1 does one of two things:

1. With probability 1/(n+1) she sits at an unoccupied table
2. Otherwise, she picks a person, uniformly at random, and sits at the same table, to their right.



Continue until we have n people seated.
Assume the supply of tables is unlimited.

Task - the Restaurant Process [RP] 1

- Write a function to simulate the RP.
- Write another function to replicate the RP many times.
- Suppose we eventually have 50 customers.
  - What is the expected number of occupied tables?
  - What is the distribution of the number of occupied tables?

Pseudocode for single RP

```
# define the number of customers (NumberToSeat)  and a vector of customers (CustVec <-
mat.or.vec(1,CustVec)
# Put customer 1 and table 1 (CustVec[1]<-1)
# initialize the count of occupied tables and customers (TablesOcc<-1, SeatedCustomers<-1)

# Now the customers arrive
while (SeatedCustomers < NumberToSeat){
   # does the next customer sit at a new table (with prob. 1/(SeatedCustomers+1). If so:
            # Increase number of seated customers and occupied tables by 1, and then
            CustVec[SeatedCustomers]<-TablesOcc
        # If not:
             # Increase number of seated customers by one
             # Choose a customer to sit to the right of (Neighbor)
             CustVec[SeatedCustomers]<-CustVec[Neighbor]
  }
```

[“Restaurant Process” repo on Github]

Task - the Restaurant Process 2

- Suppose that, conditional on there currently being N customers already seated, the prob. that the next customer sits at a new table is $\theta/(N+\theta)$, rather than $1/(N+1)$

- Suppose we eventually have 50 customers.
  - What is the expected number of occupied tables as a function of θ? [Suggest try θ=0.1,0.5,1,2,3,4,...,10,100]
  - How does the distribution of the number of occupied tables vary as a function of θ?
  - How does the run time vary as a function of θ?

- Do you see any connection with your answers to Urn Task 5 here, or between Urn Task 4 and the previous problem?

END