YaleNUSCollege

YSC2239 Lecture 6

Today’s class

- Iteration
- Chance/Probability

- Reading: Chapter 8, 9

Important Table Methods

```
t.select(column, …) or t.drop(column, …)
t.take([row_num, …]) or t.exclude([row_num, …])

t.sort(column, descending=False, distinct=False)
t.where(column, are.condition(...))

t.apply(function_name, column, …)
t.group(column) or t.group(column, function_name)
t.group([column, …]) or t.group([column, …], function_name)
t.pivot(cols, rows) or t.pivot(cols, rows, vals, function_name)

t.join(column, other_table, other_table_column)
```

Comparison and Booleans

Comparison Operators

The result of a comparison expression is a **`bool`** value

```
  x = 2          y = 3
```

Assignment statements

```
  x > 1          x > y          y >= 3

x == y          x != 2         2 < x < 5
```

Comparison expressions

Aggregating Comparisons

Summing an array or list of bool values will count the True values only.

```
1    + 0     + 1           == 2
True + False + True        == 2
sum([1   , 0    , 1   ])   == 2
sum([True, False, True])   == 2
```

(Demo)

Control Statements

Control Statements

These statements *control* the sequence of computations that are performed in a program

- The keywords `if` and `for` begin control statements

- The purpose of `if` is to define functions that choose different behavior based on their arguments

(Demo)

Random Selection

Random Selection

`np.random.choice`

- Selects uniformly at random
- with replacement
- from an array,
- a specified number of times

```
np.random.choice(some_array, sample_size)
```

(Demo)

Appending Arrays

A Longer Array

- `np.append(array_1, value)`
  - new array with `value` appended to `array_1`
  - `value` has to be of the same type as elements of `array_1`
- `np.append(array_1, array_2)`
  - new array with `array_2` appended to `array_1`
  - `array_2` elements must have the same type as `array_1` elements

(Demo)

Iteration

`for` Statements

- `for` is a keyword that begins a control statement

- The purpose of `for` is to perform a computation for every element in a list or array

(Demo)

Optional: Advanced `where`

A Closer Look at `where`

`t.where(array_of_bool_values)`

returns a table
with only the rows of `t` for which
the corresponding `bool` is `True`.

(Demo)

Probability

Basics

- Lowest value: 0
    - Chance of event that is impossible
- Highest value: 1 (or 100%)
    - Chance of event that is certain

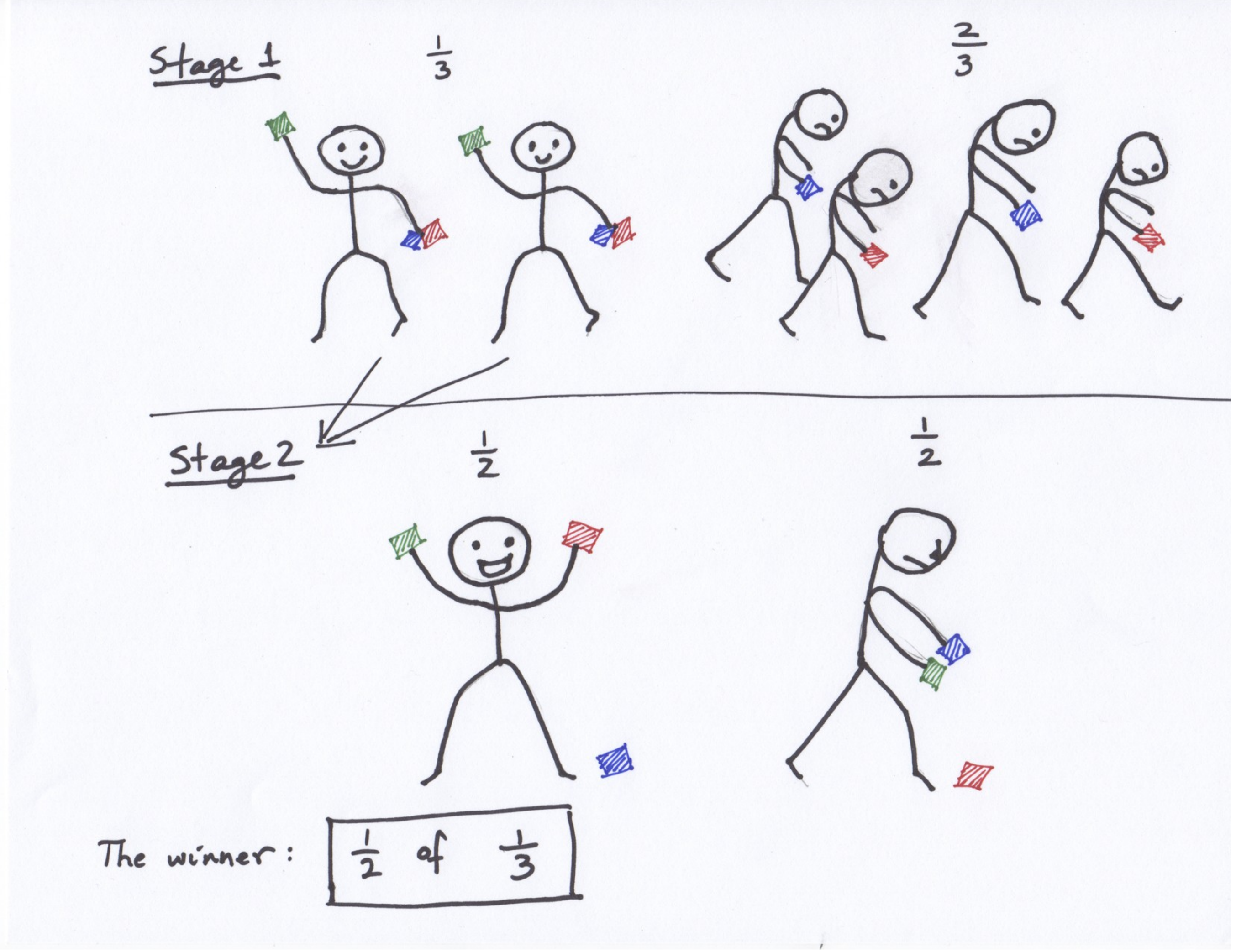
- If an event has chance 70%, then the chance that it doesn’t happen is
    - 100% - 70% = 30%
    - 1 - 0.7 = 0.3

Equally Likely Outcomes

Assuming all outcomes are equally likely, the chance of an event A is:

$$P(A) = \frac{\text{number of outcomes that make A happen}}{\text{total number of outcomes}}$$

Fraction of a Fraction



Multiplication Rule

Chance that two events *A* and *B* both happen

= P(*A* happens) x P(*B* happens given that *A* has happened)

- The answer is *less than or equal to* each of the two chances being multiplied
- The more conditions you have to satisfy, the less likely you are to satisfy them all

Addition Rule

If event *A* can happen in *exactly one* of two ways, then

$$P(A) = P(\text{first way}) + P(\text{second way})$$

- The answer is *greater than or equal to* the chance of each individual way

Example: At Least One Head

- In 3 tosses:
  - Any outcome *except* TTT
  - P(TTT) = (½) x (½) x (½) = ⅛
  - P(at least one head) = 1 - P(TTT) = ⅞ = 87.5%

- In 10 tosses:
  - 1 - (½)**10
  - 99.9%