probSim01

March 16, 2023

Simulate Probabilities

[4]:
```
:opt no-lint
```

[12]:
```
-- Imports
--{-# LANGUAGE FlexibleContexts#-}
import System.Random
import Data.List
import Text.Printf
import Data.Bits
```

Random Numbers

[13]:
```
-- Random Number Lists of Lists
rList' :: Int -> [[Float]]
rList' rSeed = [randomRs (0, 1.0) g | g <- gl (mkStdGen rSeed)]
   where gl gen = gen:(gl (fst $ split gen)) -- returns list of generators
```

Utilities

[14]:
```
toInts :: [Bool] -> [Int]
toInts bs = map (\b -> if b then 1 else 0) bs

tupleToList :: (Bool,Bool,Bool,Bool) -> [Bool]
tupleToList (a,b,c,d) = [a,b,c,d]

boolListToInt :: [Bool] -> Int
boolListToInt = foldl (\acc x -> acc * 2 + fromEnum x) 0

event :: Float -> [Float] -> [Bool]
event prob rList =
      map (\r -> r < prob) rList
```

Main Functions

The functions below deal with exactly 4 events.

I might expand it to use lists of events later.

[15]:
```
anyNEvents :: (Int -> Bool) -> [Bool] -> [Bool] -> [Bool] -> [Bool] -> Float
anyNEvents f as bs cs ds =
```

```
    let
      iList = map tupleToList (zip4 as bs cs ds)
      iInts = map toInts iList
      counts = map (\ks -> sum ks) iInts
      ones = filter f counts
    in (fromIntegral (length ones)) / (fromIntegral (length counts))

specificEvents :: (Int -> Bool) -> [Bool] -> [Bool] -> [Bool] -> [Bool] -> Float
specificEvents f as bs cs ds =
    let
      bList = map tupleToList (zip4 as bs cs ds)
      iInts = map boolListToInt bList
      trues = filter f iInts
    in (fromIntegral (length trues)) / (fromIntegral (length as))

atLeastMaskedEvents :: (Int -> Int) -> [Bool] -> [Bool] ->
                                       [Bool] -> [Bool] -> Float
atLeastMaskedEvents f as bs cs ds =
    let
      bList = map tupleToList (zip4 as bs cs ds)
      iInts = map boolListToInt bList
      val = f 15 -- figuure out and value from user
      anding = map f iInts
      ones = filter (\x -> x == val) anding
    in (fromIntegral (length ones)) / (fromIntegral (length iInts))
```

Make the Events

[16]:
```
randLists = rList' 12345 -- random seed

rLen = 200000
a = take rLen $ event 0.2 (randLists !! 0) -- e.g. Follow the money
b = take rLen $ event 0.2 (randLists !! 1) -- e.g. Whistleblower truthful
c = take rLen $ event 0.6 (randLists !! 2) -- e.g. Availability bias
d = take rLen $ event 0.1 (randLists !! 3) -- e.g. Russian plot
```

Function Examples

[19]:
```
-- Routine Test Cases

a1e = anyNEvents (\k -> k == 1) a b c d
printf "Any one event being true = %5.2f\n\n" a1e

a2e = anyNEvents (\k -> ((k == 1) || (k == 2))) a b c d
printf "Any one or two events being true = %5.2f\n\n" a2e

aes = anyNEvents (\k -> ((k==1) || (k==2) || (k==3) || (k==4))) a b c d
printf "Any event being true = %5.2f\n\n" aes
```

```
aevN = anyNEvents (\k -> (k == 0)) a b c d
printf "Any event being true (using neg) = %5.2f\n\n" (1-aevN)

se = specificEvents (\k -> k == 6) a b c d
printf "\nSpecifc Events b and c not a not d = %5.2f\n\n" se

oldWay = 0.8 * 0.2 * 0.6 * 0.9
printf "The old fashoned way = %5.2f\n\n" oldWay

se2 = specificEvents (\k -> (k == 6) || (k == 2)) a b c d
printf "\nSpecifc Events b and c but never together = %5.2f\n\n" se2

al = atLeastMaskedEvents (\k -> k .&. 6) a b c d
printf "\nAt Least Events b and c (a and d don't care) = %5.2f\n\n" al
```

```
Any one event being true =  0.49

Any one or two events being true =  0.73

Any event being true =  0.77

Any event being true (using neg) =  0.77


Specifc Events b and c not a not d =  0.09

The old fashoned way =  0.09


Specifc Events b and c but never together =  0.43


At Least Events b and c (a and d don't care) =  0.12
```

Early Simple And/Or Attempt

[113]:
```
pAnd :: [Bool] -> [Bool] -> [Bool]
pAnd b1 b2 = map (\(a,b) -> a && b) $ zip b1 b2

pOr :: [Bool] -> [Bool] -> [Bool]
pOr b1 b2 = map (\(a,b) -> a || b) $ zip b1 b2

calc :: ([Bool] -> [Bool] -> [Bool]) -> [Bool] -> [Bool] -> Float
calc f b1 b2 =
    let a = f b1 b2
        suc = foldl  (\cum b -> if b then (cum+1.0) else cum) 0.0 a
    in suc/(fromIntegral (length b1))

calc pOr (pOr a b) (pOr c d)
```

```
0.76973
```

[ ]: