MIT data science

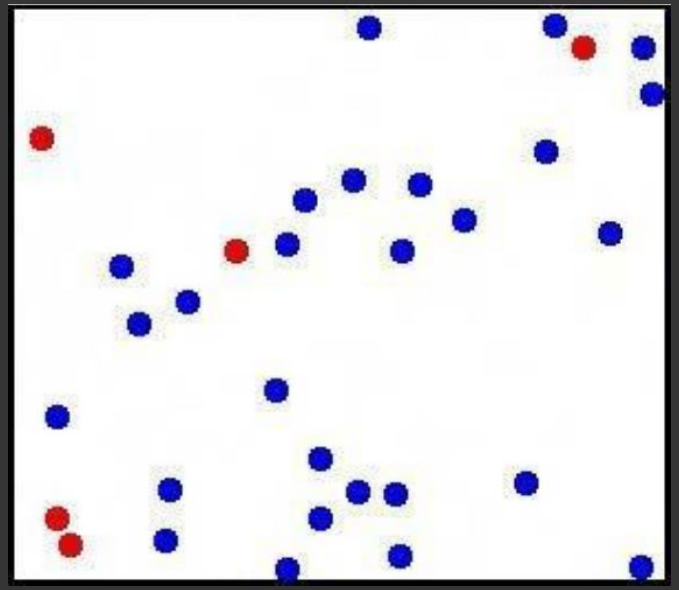
5. Random Walks
6. 몬테카를로 시뮬레이션

Sunggon Song

5. Random Walks

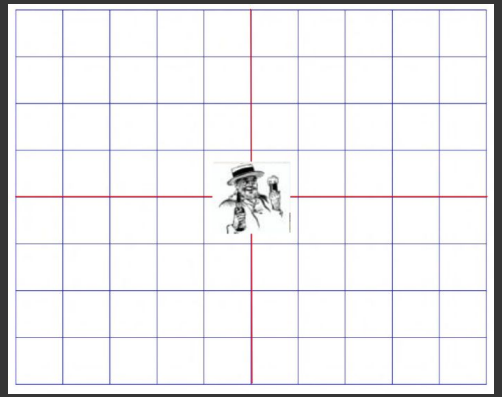
- ➔ 무작위 행보(Random Walks)
- ➔ 위생 검사(Sanity Check)

무작위 행보(Random Walks)

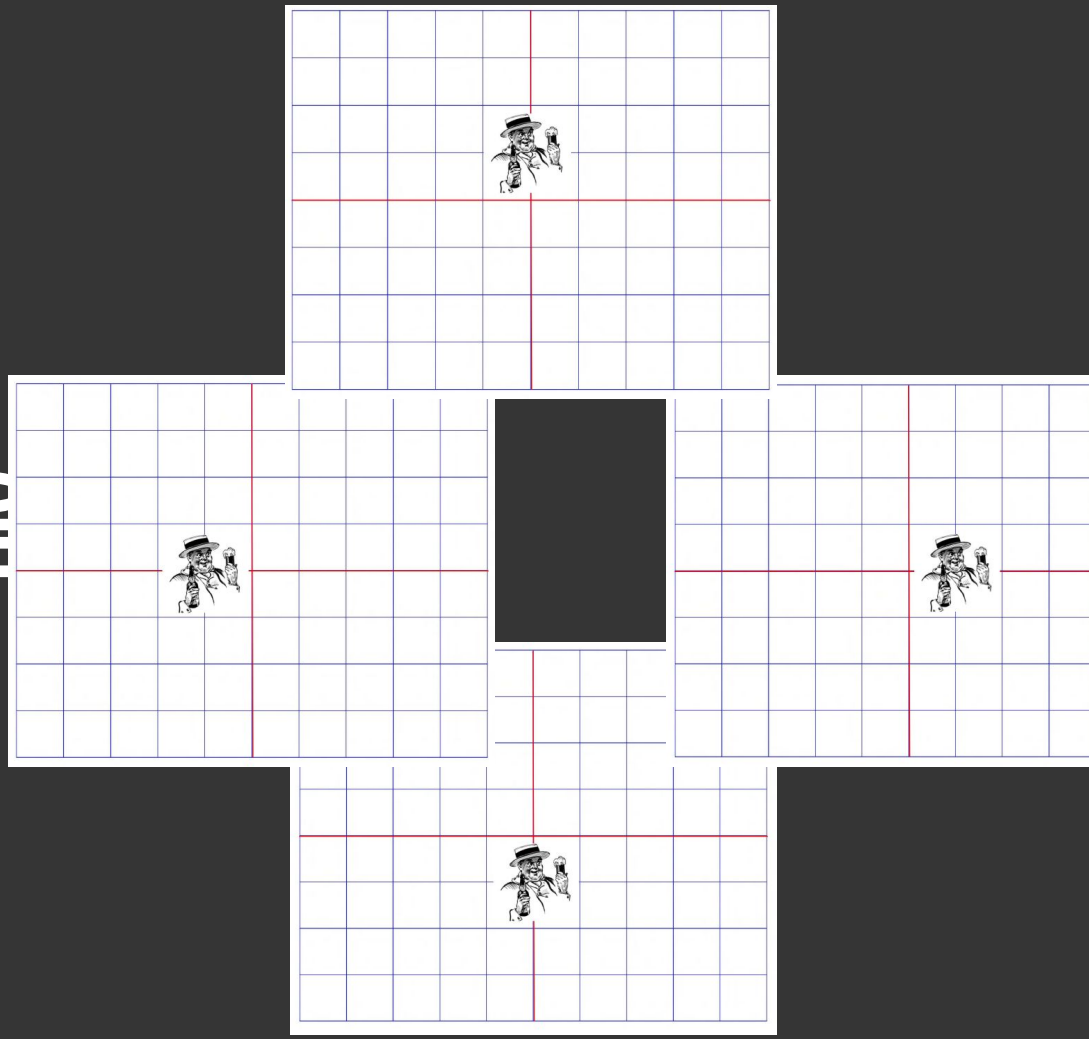


- 많은 영역에서 중요
  - 주식 시장에서 가격의 움직임을 입증하는데 가장 좋은 모델
  - 현대의 많은 포트폴리오 분석이 기반을 두고 있음
  - 전파를 모델링할 때도 랜덤 워크를 사용, 열 전파나 분자의 전파
- 우리 주변의 세계를 이해하기 위해 시뮬레이션을 어떻게 사용할지에 대한 좋은 사례를 제공
- 강의의 예시 : 술 취한 사람의 걸음과 편향성을 가진 술 취한 사람의 걸음 차이를 파악

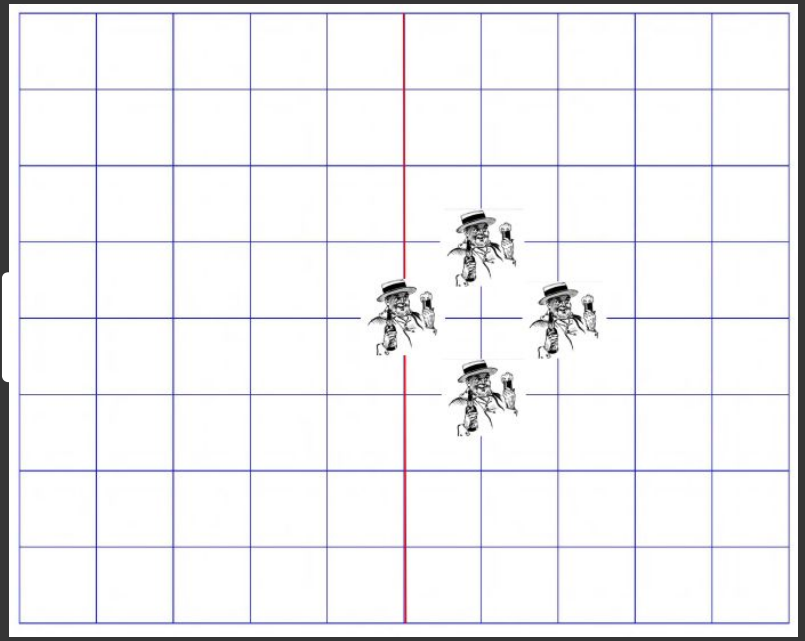
주정뱅이의 걸음



가능한 첫걸음



두걸음 후 거리



100,000걸음 뒤 예상 거리?

시뮬레이션 구조

- 걸음 걷는 한 행보를 시뮬레이션
- 이 행보를 번 시뮬레이션
- 원점에서의 거리의 평균 보고

—

우선 몇 가지 유용한 추상화

- Location — 위치
- Field — Location 클래스와 Drunk 클래스의 집합
- Drunk — 영역 안에서 위치를 오가는 한 사람

Location Class

- __init__ 위치 x, y 설정
- move 이동한 위치 반환
- getX, getY : 현재 위치 x, y 반환
- distFrom : other Location과의 거리 반환
- __str__ : 현재 x, y 위치를 문자열로 반환

Immutable ?

Drunk Class

- __init__ : 이름 설정
- __str__ : 이름 반환

상속하기 위한 base class

Drunk의 하위 클래스 2가지

- 일반적인 취객은 무작위로 돌아다님
- 마조히즘적인 취객은 북쪽으로 이동하려 함

Drunk의 하위 클래스 2가지

- class UsualDrunk(Drunk):
  def takeStep(self):
    stepChoices = [(0,1), (0,-1), (1, 0), (-1, 0)]
    return random.choice(stepChoices)

- class MasochistUsualDrunk(Drunk):
  def takeStep(self):
    stepChoices = [(0.0,1.1), (0.0,-0.9), (1.0, 0.0), (-1.0, 0.0)]
    return random.choice(stepChoices)

Immutable ?

Field 클래스

- __init__ : drunks 맵 초기화 (drunk가 key이고, 값이 Location)
- addDrunk: drunks 맵에 drunk가 있다면 중복에러 발생, 없다면 맵에 추가
- getLoc: drunk가 맵에 없다면 존재하지 않는다는 에러 발생, 있으면 위치 반환
- moveDrunk: drunk가 맵에 없다면 존재하지 않는다는 에러 발생, 있다면 drunk의 takeStep 호출얻은 거리 만큼 drunk의 move를 호출하여 이동한 거리를 drunks 맵에 설정

Immutable ?

한 행보 시뮬레이션

- def walk(f, d, numSteps):

여러 행보 시뮬레이션

- def simWalks(numSteps, numTrials, dClass):

모두 합치기

- def drunkTest(walkLengths, numTrials, dClass):

시도해봅시다

- drunkTest((10, 100, 1000, 10000), 100, UsualDrunk)

그럴듯한가요?

위생 검사(Sanity Check)

- 시뮬레이션을 생성할 땐 위생 검사가 필요
- 시뮬레이션이 실제로 말이 되는지 확인하는 과정
  - 의심을 가져야 함
  - 코드에 버그가 있을 수 있음!
  - 우리가 답을 안다고 하는 경우에 대해서 시도
- 사용할 수 있는 라이브러리
  - Numpy
  - SciPy
  - Matplotlib
  - PyLab

위생 검사(Sanity Check)

- 우리가 답을 알고 있다고 생각하는 경우를 시도해봅시다

- drunkTest((0, 1, 2) 100, UsualDrunk)

시도해봅시다

- drunkTest((10, 100, 1000, 10000), 100, UsualDrunk)

마조히즘적인 술 취한 사람은

- random.seed(0)
- simAll((UsualDrunk, MasochistDrunk), (1000,10000), 100)

트렌드 시각화

- 각 술 취한 사람의 종류에 대해 여러 길이의 행보를 시뮬레이션
- 각 술 취한 사람의 종류에 대해 각 길이의 행보의 종점의 거리를 그래프로 나타냄

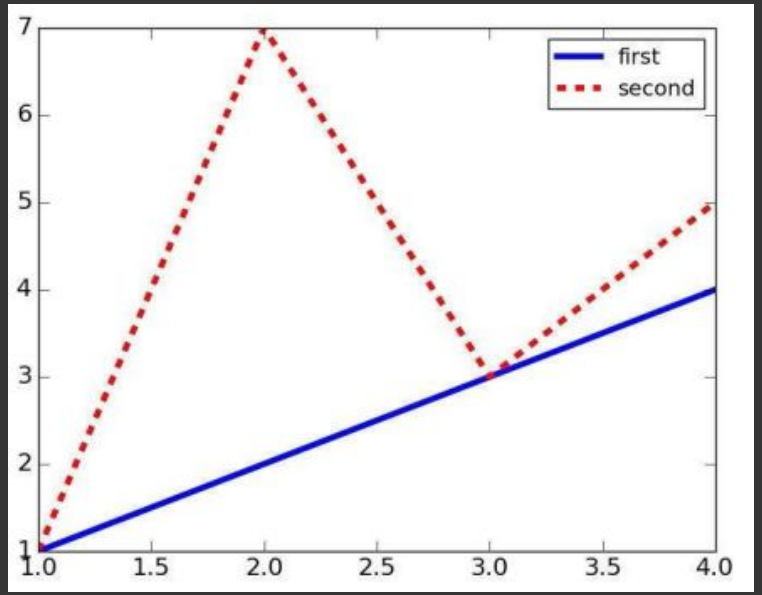
Pylab

- NumPy 는 벡터 행렬 그리고 많은 고수준의 수학적인 함수를 제공
- SciPy 는 과학자들에게 유용한 수학적 클래스와 함수들을 제공
- Matploblib 은 그래프 그리는 것을 위한 객체 지향 API를 제공
- PyLab 은 MATLAB과 비슷한 인터페이스를 제공하는 다른 라이브러리를 결합

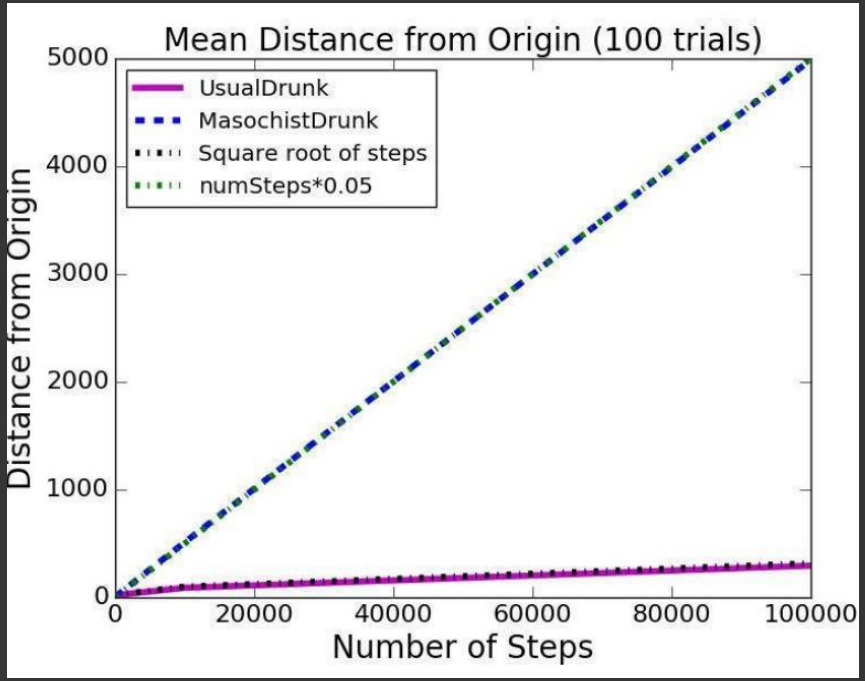
그래프

- pylab.plot 의 처음 두 인자는 같은 길이의 수열이어야 함
- 첫 번째 인자는 x축을 의미
- 두 번째 인자는 y축을 의미
- 추가적인 인자도 많음
- 점들이 순서대로 그려짐. 각 점이 그려지면 디폴트 스타일로 점들을 잇는 선이 그려짐

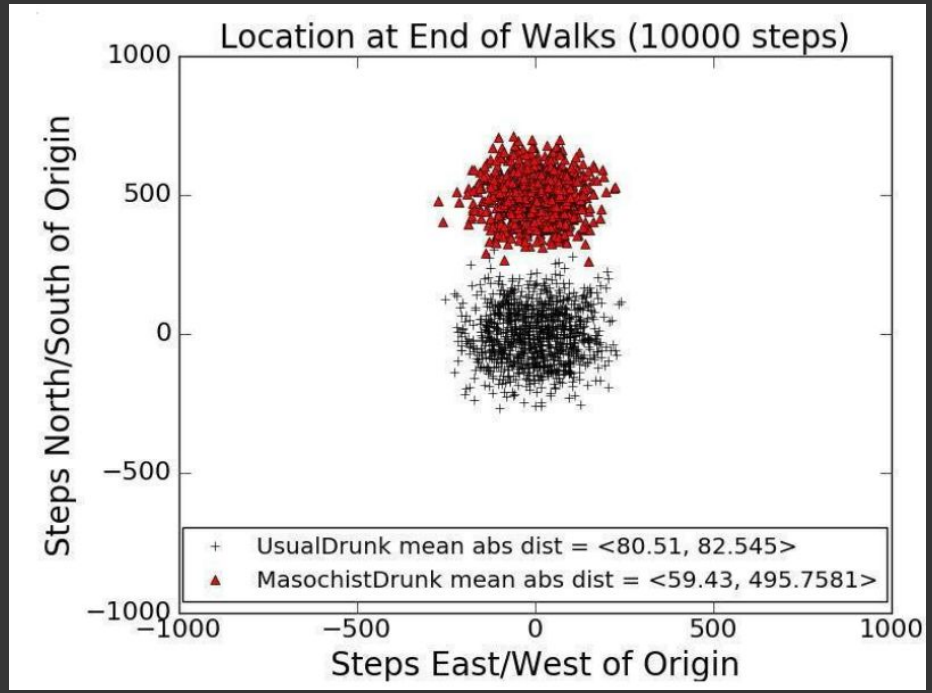
예제



거리 트렌드


Mean Distance from Origin (100 trials)
UsualDrunk
MasochistDrunk
Square root of steps
numSteps*0.05
Distance from Origin
Number of Steps
0
1000
2000
3000
4000
5000
0
20000
40000
60000
80000
100000

종점 위치

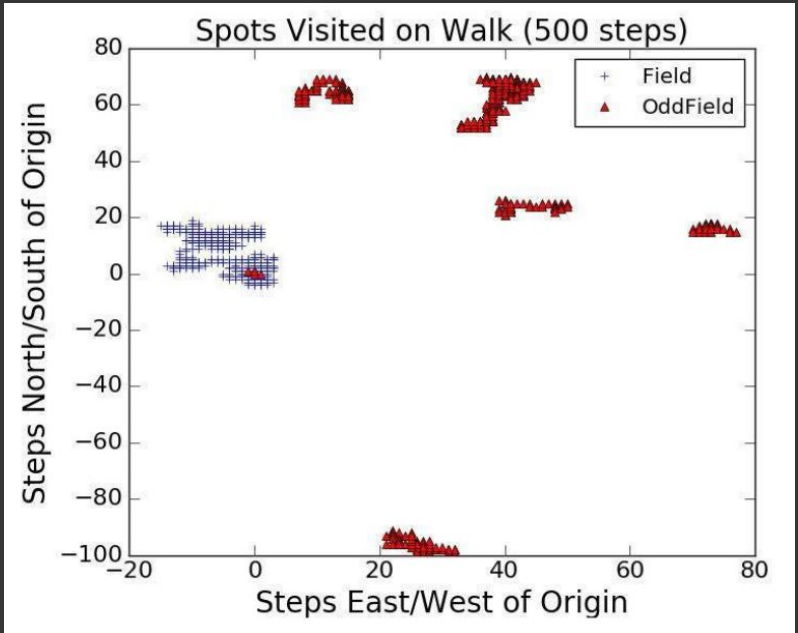

Location at End of Walks (10000 steps)
Steps North/South of Origin
Steps East/West of Origin
1000
500
0
−500
−1000
−1000
−500
0
500
1000
UsualDrunk mean abs dist = <80.51, 82.545>
MasochistDrunk mean abs dist = <59.43, 495.7581>

Wormholes이 있는 Field

Field의 하위 클래스 (OddField)

- __init__ : 전달받은 Hole개수만큼 womholes맵을 랜덤한 (x, y)를 키로하고 랜덤한 위치를 값으로 저장
- moveDrunk : Field의 moveDrunk를 호출하고 drunks맵의 현재 x,y위치를 받아 (x,y)가 wormhomes에 있으면 drunks의 위치를 wormhomes의 위치로 변경

한 행보 동안 들른 장소


Spots Visited on Walk (500 steps)
Field
OddField
Steps North/South of Origin
Steps East/West of Origin
80
60
40
20
0
−20
−40
−60
−80
−100
−20
0
20
40
60
80

정리

- 랜덤 워크를 하는 요점은 시뮬레이션 모델이 아닌 어떻게 만드는지에 관한 것
- 클래스를 정의하는 것부터 한번의 시도와 여러 번의 시도에 대응하는 함수를 만들고 결과를 보는 과정
- 시뮬레이션에 점진적 변화를 줘서 다른 문제를 조사할 수 있음
- 처음엔 간단한 시뮬레이션으로 시작해 잘 되지 않는 이유를 알고, Sanity check로 잘못된 것을 체크
- 통찰력을 얻기 위해서 plot 스타일로 그래프를 그림

6. 몬테카를로 simulation

- ➔ 몬테 카를로 시뮬레이션(Monte Carlo Simulation)
- ➔ 큰 수의 법칙 또는 베르누이의 법칙
- ➔ 도박사의 오류
- ➔ 평균으로의 회귀
- ➔ 분산과 표준편차
- ➔ 확률 분포

몬테 카를로 시뮬레이션 (Monte Carlo Simulation)

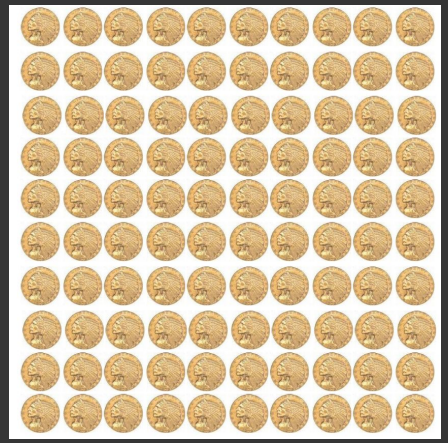
몬테 카를로 시뮬레이션 (Monte Carlo Simulation)

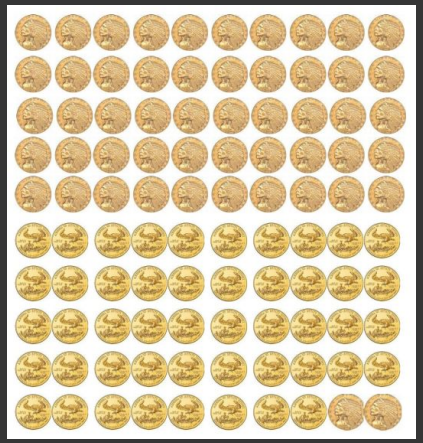
- 추리통계학을 이용해 알 수 없는 값을 추정하는 방법
- 핵심 개념은 모집단

- 모집단 : 가능한 예시들의 전체 집합
- 모집단부터 적당한 부분 집합을 뽑음
  - 표본의 통계를 통해 모집단을 추론
  - 일반적으로 모집단은 매우 큰 집단이고 표본은 그것보다 작은 집단
  - 무작위로 표본을 추출하면 그 표본이 모집단과 동일한 특성을 갖는 경향이 있음
  - 우리가 취할 수 있는 수많은 랜덤 워크, 만개를 보지 않고 100개 추출해서 평균 계산한 후 기대값 계산
  - 표본 추출이 무작위어야 함! 무작위로 뽑은 표본이 아니면 모집단과 같은 특성을 가질 것이라고 기대할 수 근거가 없음

예시

- 동전 한 개를 무한 번 던졌을 때, 앞면이 나오는 비율을 추정해라
- 한 번 던지기, 동전 두 번 던지기, 동전 100번 던지기





신뢰도에 차이가 있는 이유

- 예측의 신뢰도는 두 가지에 의존
- 표본의 크기 (e.g., 100 vs. 2)
- 표본의 분산 (e.g., 모두 앞면 vs. 52개 앞면)
- 분산이 커질수록, 같은 수준의 신뢰도를 갖기 위해서는 더 큰 표본이 필요

룰렛

클래스 정의(FairRoulette)

- __init__ : pockets배열에 1부터37까지 숫자를 추가한다.
- spin : pockets배열에서 랜덤하게 선택한 숫자로 ball를 설정한다.
- betPocket: 배팅한 pocket과 spin에서 설정한 ball이 일치하면 amt * pocketOdds만큼 반환한다. 그렇지 않으면 -amt를 반환한다.
- __str__ : 'Fair Roulette' 문자열 반환

몬테 카를로 시뮬레이션

- def playRoulette

100번과 1M번의 휠 회전

베르누이의 법칙 - 큰 수의 법칙

- 실제 확률이 동일한 독립 사건이 반복되면 실행 횟수가 무한대로 갈수록 p와 다른 결과가 나오는 횟수와 비율이 0으로 수렴
- 공정한 룰렛 휠을 무한번 돌리면 기대값이 0이 됨

도박사의 오류

- 도박사의 오류에 의하면 사람들은 기대와 다른 이변이 일어나면 미래에 다시 정상으로 돌아올 것이라고 믿음
- 서로 영향을 끼치지 않는 일련의 확률적 사건들에서 상관관계를 찾는 오류를 발생
- "1913년 8월 18일, 몬테 카를로의 카지노에서검정색이 26번 연속으로 나왔다[룰렛에서]. ...검정색이 경이적으로 15번 나올 때쯤부터,사람들은 미친듯이 달려가 빨간색에 베팅했다."
- 26번 연속으로 빨간색이 나올 확률 : 1/67,108,865
- 앞의 25번이 모두 빨간색이었을 때, 26번 연속으로 빨간색이 나올 확률 : 1/2

평균으로의 회귀

- 부모가 둘 다 평균보다 키크면 자식이 부모보다 작을 가능성이 높다
- 역으로 부모가 평균보다 작으면 자식은 부모보다 클 가능성이 높다
  - 도박사의 오류와 미묘하게 다름
- 극단적인 사건 다음에 오는 사건은 덜 극단적인 경향이 있다
  - 공정 룰렛 휠을 10번 돌려 빨간색이 나온다면(극단적 사례) 10번에서는 빨간색이 10보다 적게 나온다는 개념. 덜 극단적인 사건이다
- 극단적인 결과보다는 20번 돌린 것의 평균 결과가 50% 빨간색이라는 평균값에 더 가까울 것
- 더 많은 표본을 취할수록 평균에 더 가까워짐

공정하지 않은 카지노

룰렛의 하위 클래스 2개

- class EuRoulette(FairRoulette):
- class AmRoulette(EuRoulette):

게임 결과 비교

가능한 결과들의 표본 공간

- 표본 추출을 통해서는 완벽한 정확도 보장 불가
- 예측값이 무조건 틀리다는 건 아님
- 중요한 질문:
  - 우리의 대답에 정당한 확신을 갖기 위해서는 얼마나 많은 표본이 필요한가?
- 기저 분포의 변이성에 의존

분산과 표준편차

- 분산
- 표준편차
- 신뢰 구간
- 경험적인 규칙을 적용하기 위한 가정

—

- 기저 확률의 변이성에 따라 필요한 표본의 수가 달라짐
- 데이터의 변이성을 알기 위해 알아야 되는 개념 : 분산
  - 단지 항목의 개수가 많다는 이유로 분산이 높아지는 것을 막기 위해서, 항목의 개수로 정규화함
  - 제곱을 하는 이유 :
    - 차이가 양수든 음수든 상관이 없다는 것, 평균으로부터 어느 쪽에 있는지보다 근처에 있지 않다는 것이 중요함
    - 거리를 제곱하여 이상치를 특별히 강조함 (장점이자 단점)
- 표준편차
  - 분산의 제곱근
  - 표준편차 그 자체로는 의미없고, 항상 평균을 고려해 생각해야 함

$$variance(X) = \frac{\sum_{x \epsilon X}(x-\mu)^2}{|X|}$$

$$\sigma(X) = \sqrt{\frac{1}{|X|}\sum_{x \in X}(x-\mu)^2}$$

코드를 선호하는 사람들을 위해

- def getMeanAndStd(X):

```
    mean = sum(X)/float(len(X))
    tot = 0.0
    for x in X:
      tot += (x - mean)**2
    std = (tot/len(X))**0.5
    return mean, std
```

신뢰 구간

- 우리는 종종 평균만 가지고 예측하려고 함
- 값을 모르는 매개변수를 설명할 때 기댓값과 같은 특정 값을 주는 것보단 신뢰 구간으로 표현하는 것이 좋음
- 신뢰구간은 모르는 값이 포함될 가능성이 높은 구간과 그 구간에 존재할 확률을 알려줌

“ 유럽 룰렛을 10k번 돌린 결과가 -3.3%이다. 오차 범위는 +/-3.5%이고 신뢰도는 95%이다. ”

이게 무슨 의미일까?

10k번의 베팅을 무한 번 실행하면,

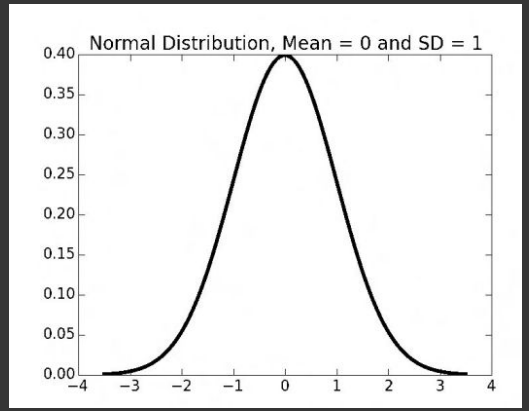
결과의 기댓값은 -3.3%
전체 실행 중 95%는 6.8%와 0.2% 사이의 결과

신뢰 구간은 어떻게 계산할까?

- 경험적인 규칙을 사용
  - 데이터를 얻어 평균을 찾고 표준편차를 계산하면 데이터의 68%는 평균값 앞뒤로 1 표준 편차 범위 이내에 있음
  - 데이터의 ~95%는 평균으로부터 1.96 표준편차 이내에 존재
  - 데이터의 ~99.7%는 평균으로부터 3 표준편차 이내에 존재

경험적인 규칙을 적용하기 위한 가정

- 평균 추정 오차가 0
  - 높게 예측할 가능성과 낮게 예측할 가능성이 같아야 함
  - 오차에 편향이 없다는 가정
- 오차의 분포가 정규분포
  - 가우스 분포, 정규분포
- 이 두 가정하에 경험적 규칙은 항상 유효



확률 분포

- 확률 분포 : 확률 변수가 서로 다른 값을 가지는 상대적 빈도를 나타내는 개념
- 확률 변수
  - 이산 확률 변수
    - 유한 집합의 값들을 가짐
    - 동전을 던지면 앞면과 뒷면이란 2개의 값만 나옴
  - 연속 확률 변수
    - 확률밀도함수(PDF)를 사용
    - 두 값 사이 어딘가에 존재할 확률을 알려줌

PDF’s

- 확률밀도함수(PDFs)로 정의되는 분포
- 확률변수가 두 값 사이의 구간에 속할 확률을 구할 수 있음
- 변수의 최솟값과 최댓값 사이의 값을 x축으로 갖는 곡선 정의
- 두 점 사이의 곡선하 면적이 해당 구간에 위치할 확률과 같음

Reference

MIT data science

https://www.edwith.org/datascience/lecture/33892/