ECSE 6810 PGM HW.

Shiuli Subhra Ghosh.
RIN - 662056315

1. We wish to calculate $P(H, E_1, E_2)$

(b) $P(E_1, E_2)$, $P(H)$ and $P(E_1 E_2|H)$ is sufficient.

Because

$$P(H, E_1, E_2) = P(H|E_1,E_2) \cdot P(E_1, E_2) = \underline{P(E_1, E_2|H) \cdot P(H)}$$

So, From the conditional Expansion, $P(E_1, E_2|H) \cdot P(H)$ is sufficient we don't even need to calculate $P(E_1, E_2)$.

however (a) $P(E_1, E_2)$, $P(H)$, $P(E_1|H)$ is and $P(E_2|H)$ this is insufficient. In particular we can't obtain $P(E_1, E_2|H)$ from $P(E_1|H)$ and $P(E_2|H)$ unless we know that E_1 and E_2 are independent. Knowing marginal distribution of $P(E_1 E_2)$ will not help in this case.

(c) $P(H)$, $P(E_1|H)$ and $P(E_2|H)$ is also not sufficient because it is same as a ~~b~~ only without $P(E_1, E_2)$. In fact we don't need to know $P(E_1, E_2)$. We can find it using the conditional sum $= P(E_1, E_2) = \sum_{H=a} P(H=a) \times P(E_1, E_2|H=a)$. If we know $P(E_1, E_2|H)$. But again it is impossible to find $P(E_1, E_2|H)$ from $P(E_1|H)$ and $P(E_2|H)$ unless we know that E_1 and E_2 are independent.

Now if we know E_1 and E_2 are independent.

$$P(H, E_1, E_2) = P(H) \cdot P(E_1, E_2 | H).$$
$$= P(H) \cdot P(E_1|H) \cdot P(E_2|H).$$

So,
- $P(E_1, E_2)$, $P(H)$, $P(E_1|H)$, $P(E_2|H)$
- $P(E_1, E_2)$, $P(H)$, $P(E_1, E_2|H)$
- $P(E_1|H)$, $P(E_2|H)$ and $P(H)$

all 3 are sufficient.

2. (a) $X \perp Y, W | Z \Rightarrow X \perp Y | Z$ (true)

$X \perp Y, W | Z \Rightarrow P(X, Y, W | Z) = P(X|Z) \cdot P(Y, W | Z).$

lets marginalize $P(X, Y, W | Z)$ on W.

$$\sum_{W} P(X, Y, W | Z) = \sum_{W} P(X|Z) \cdot P(Y, W|Z)$$
$$= P(X|Z) \sum_{W} P(Y, W|Z)$$
$$= P(X|Z) \sum_{W} P(W|Z) \cdot P(Y|W, Z)$$
$$= P(X|Z) \cdot P(Y|Z)$$

If there are 3 RV's X, Y, Z and if X, Y are independent given Z, then $P(X, Y|Z) = P(X|Z) \cdot P(Y|Z)$

and. we denote it as $X \perp Y | Z$

So, $P(X|Z) \cdot P(Y|Z) \Rightarrow X \perp Y | Z$. hence the identity is true.

(b) $X \perp Y | Z$ and $X, Y \perp W | Z \Rightarrow X \perp W | Z$.

$X \perp Y | Z \Rightarrow P(X, Y | Z) = P(X|Z) \cdot P(Y|Z)$.

$X, Y \perp W | Z \Rightarrow P(X, Y, W | Z) = P(X, Y | Z) \cdot P(W|Z)$

$= P(X|Z) \cdot P(Y|Z) \cdot P(W|Z)$

So, X, Y, Z, W are 4 RVs. out of which X, Y and W are independent given Z.

marginalize on Y,

$\sum_y P(X, Y, W | Z) = P(X, W | Z)$.

or, $\sum_y P(X|Z) \cdot P(Y|Z) \cdot P(W|Z) = P(X, W|Z)$

or, $P(X|Z) \cdot P(W|Z) \sum_y P(Y|Z) = P(X, W | Z)$

or, $P(X|Z) \cdot P(W|Z) \cdot 1 = P(X, W | Z)$.

hence $X \perp W | Z$. (proved).

(c). $X \perp Y, W \mid Z$ and $Y \perp W \mid Z$.

So, $X \perp Y, W \mid Z \Rightarrow P(X, Y, W \mid Z) = P(X \mid Z) . P(Y, W \mid Z)$

$Y \perp W \mid Z \Rightarrow P(Y, W \mid Z) = P(Y \mid Z) . P(W \mid Z)$.

hence, $P(X, Y, W \mid Z) = P(X \mid Z) . P(Y \mid Z) . P(W \mid Z)$

$= P$

lets marginalize w.r.t Y.

So, $\sum_y P(X, Y, W \mid Z) = P(X, W \mid Z) . = \sum_y P(X \mid Z) P(Y \mid Z) P(W \mid Z)$.

$= P(X \mid Z) . P(W \mid Z) . 1$.

So, $X \perp W \mid Z$

So, $P(X, Y, W \mid Z) = P(X \mid Z) . P(Y \mid Z) . P(W \mid Z)$

$= P(X, W \mid Z) . P(Y \mid Z)$

hence from this factorization we get.

$X, W \perp Y \mid Z$. (proved)

(d) $X \perp Y \mid Z$ and $X \perp Y \mid W \Rightarrow X \perp Y \mid Z, W$ (False)

Counter example –

Let X, Y, Z ~~and W~~ are iid, each with equal probabilities of being -1 and 1.

Let $W = XYZ$

So, in this example, $X \perp Y \mid Z$

$X \perp Y \mid W$

But when we consider 4 variables, and if we know Z and W then the independence doesn't hold true. because we know $W = XYZ$.

Hence $X \not\perp Y \mid Z, W$.

Also, graphical example (done a course on Causality (ETH Zurich))

$\cancel{X \rightarrow Z \rightarrow D \leftarrow W}$. $X \rightarrow Z \leftarrow D \rightarrow W \leftarrow Y$

there is a path from X to Y. Z and W are the colliders. So, If we don't condition on both the colliders the path is blocked. Because there is a confounder D. But conditioning on both Z and W makes the path Active.

So, $X \perp Y \mid Z$

$X \perp Y \mid W$.

$X \not\perp Y \mid Z, W$.

hence the result is False.

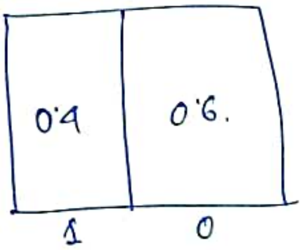
3. 1) Given a binary random variable $X \in \{0,1\}$ with $P[X=1] = 0.4$ we need to draw N random samples where $N = [50, 100, 150, 200]$.

So, we will follow Monte carlo simulation for random sample generation.

here the random variable can take 2 values.

Hence we will divide the region between 0 and 1 in 2 parts using parameter θ.

here $\theta = 0.4$.



then we will generate $N = [50, 100, 150, 200]$ random variables following $U(0,1)$.

$X = 1$ if u is in between ~~0.6 to~~ $[0, 0.4]$
0 to 0.4.

$X = 0$ if u is in between 0.4 to 1.
$(0.4, 1]$

So, this is the sampling method we can follow to generate N numbers of samples.

Given the samples, now we will obtain the MLE of $P[X=1]$, which is $= \dfrac{\text{number of times } x=1}{N}$

Problem 3

September 18, 2022

[2]:
```
import math
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
from scipy.optimize import brentq
from scipy.stats._discrete_distns import binom
```

Given a binary random variable X ∈ {0,1} first we need to generate random samples using Monte Carlo Simulation. Details description is in the sheet.

[3]:
```
N = [50,100,150,200] # Array of number of samples need to be generated
theta = 0.4          # theta = P[X=1] = 0.4 (given)
sample_dict = {}

def generate_samples(n, theta):
    sample = []
    region = [0,1]
    for i in range(n):
        x = np.random.uniform(low=0.0, high=1.0)
        if 0<= x <theta :
            sample.append(region[1])
        else:
            sample.append(region[0])
    return(sample)
```

1 Generating the samples N = [50,100,150,200]

[8]:
```
for i in N:
    sample_dict[i] = generate_samples(i, theta)

print(sample_dict)
```

```
{50: [0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0,
1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1], 100:
[0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1,
0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1,
0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1], 150: [1, 1, 0, 0,
1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0,
1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1,
```

```
0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1,
1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1], 200: [0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0,
0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0,
0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0,
0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1,
1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0]}
```

2 Finding theta_mle for each sample size

[9]:
```python
mle_dict = {}
for i in N:
    mle_dict[i] = sum(sample_dict[i])/i
print(mle_dict)
```

```
{50: 0.42, 100: 0.43, 150: 0.44, 200: 0.39}
```

3 Bernouli Confidence Interval using approximation using 95% confidence

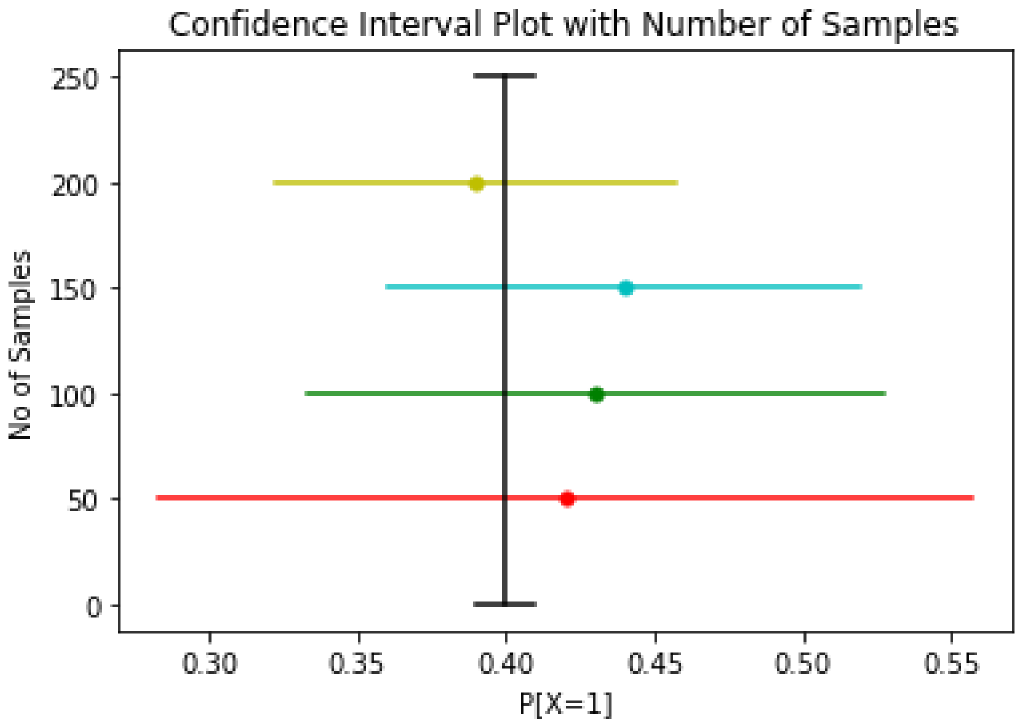
The confidience interval is $Z_{\frac{1+\alpha}{2}} \times \sqrt{(\theta \times (1-\theta)/N)}$

[10]:
```python
Z_95 = stats.norm.ppf((1+0.95)/2, 0,1)
conf_dict_norm = {}
lower_conf = []
upper_conf = []

for i in N:
    theta_hat = mle_dict[i]
    conf_dict_norm[i] = [theta_hat - Z_95 * math.sqrt(theta_hat * (1-theta_hat)/i) ,␣
 ↪theta_hat + Z_95 * math.sqrt(theta_hat * (1-theta_hat)/i)]
    lower_conf.append(theta_hat - Z_95 * math.sqrt(theta_hat * (1-theta_hat)/i))
    upper_conf.append(theta_hat + Z_95 * math.sqrt(theta_hat * (1-theta_hat)/i))

colors = ['r', 'g', 'c', 'y']

for i in range(len(N)):
    plt.plot([lower_conf[i], upper_conf[i]],[N[i], N[i]], color = colors[i])
    plt.plot(mle_dict[N[i]], N[i], marker="o", markersize=5, color = colors[i])
plt.plot([theta,theta], [0, 250], color = 'black')
plt.plot([theta - 0.01, theta + 0.01], [250,250], color = 'black')
plt.plot([theta - 0.01, theta + 0.01], [0,0], color = 'black')
plt.title('Confidence Interval Plot with Number of Samples')
plt.xlabel('P[X=1]')
plt.ylabel('No of Samples')
plt.show()
```



4 Exact confidence interval (source code from scipy github)

[11]:
```
def _findp(func):
    try:
        p = brentq(func, 0, 1)
    except RuntimeError:
        raise RuntimeError('numerical solver failed to converge when '
                           'computing the confidence limits') from None
    except ValueError as exc:
        raise ValueError('brentq raised a ValueError; report this to the '
                         'SciPy developers') from exc
    return p


def binom_exact_conf_int(k, n, confidence_level, alternative):
    """
    Compute the estimate and confidence interval for the binomial test.
    Returns proportion, prop_low, prop_high
    """
    if alternative == 'two-sided':
        alpha = (1 - confidence_level) / 2
        if k == 0:
            plow = 0.0
        else:
            plow = _findp(lambda p: binom.sf(k-1, n, p) - alpha)
```

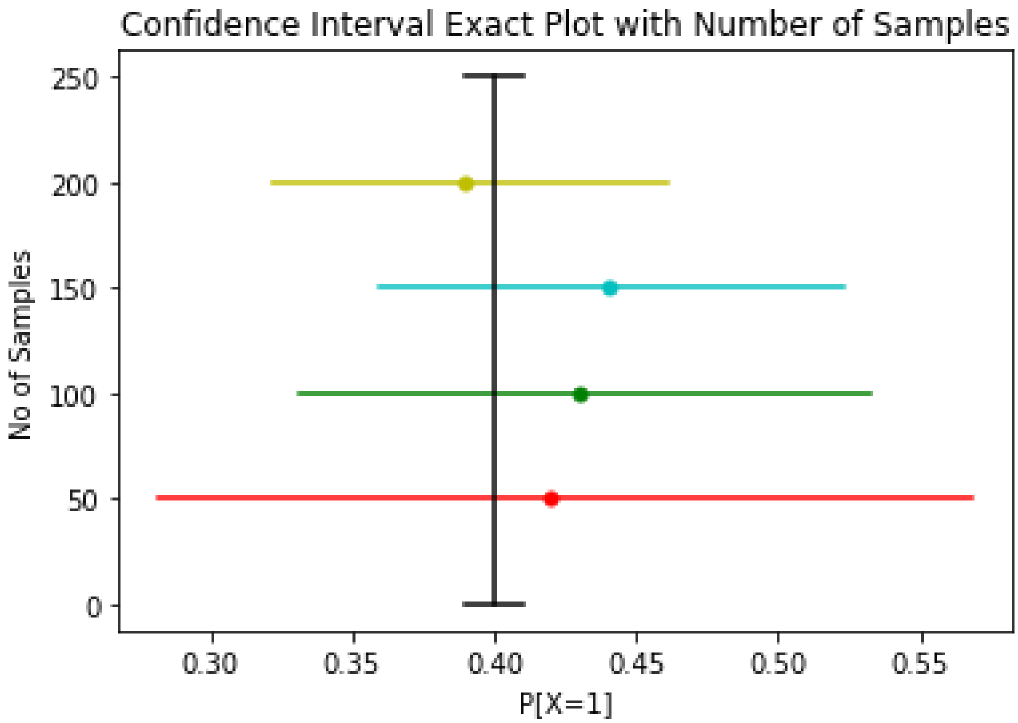
```
        if k == n:
            phigh = 1.0
        else:
            phigh = _findp(lambda p: binom.cdf(k, n, p) - alpha)
    elif alternative == 'less':
        alpha = 1 - confidence_level
        plow = 0.0
        if k == n:
            phigh = 1.0
        else:
            phigh = _findp(lambda p: binom.cdf(k, n, p) - alpha)
    elif alternative == 'greater':
        alpha = 1 - confidence_level
        if k == 0:
            plow = 0.0
        else:
            plow = _findp(lambda p: binom.sf(k-1, n, p) - alpha)
        phigh = 1.0
    return(plow, phigh)
```

[12]:
```
conf_dict_exact = {}
lower_conf_exact = []
upper_conf_exact = []

for i in N:
    low, high = binom_exact_conf_int(sum(sample_dict[i]),i, 0.95, alternative =␣
 ↪'two-sided')
    conf_dict_exact[i] = [low, high]
    lower_conf_exact.append(low)
    upper_conf_exact.append(high)

colors = ['r', 'g', 'c', 'y']

for i in range(len(N)):
    plt.plot([lower_conf_exact[i], upper_conf_exact[i]],[N[i], N[i]], color = colors[i])
    plt.plot(mle_dict[N[i]], N[i], marker="o", markersize=5, color = colors[i])
plt.plot([theta,theta], [0, 250], color = 'black')
plt.plot([theta - 0.01, theta + 0.01], [250,250], color = 'black')
plt.plot([theta - 0.01, theta + 0.01], [0,0], color = 'black')
plt.title('Confidence Interval Exact Plot with Number of Samples')
plt.xlabel('P[X=1]')
plt.ylabel('No of Samples')
plt.show()
```



4.1 Observations

After plotting the confidence intervals with respect to the sample size, for both the cases approximate and the exact one, the confidece interval is getting smaller if the sample size increases. Infact the sample mean is also close to the true mean when the sample size is large. this supports the statement for Law of Large numbers. More number of samples provides better estimates for mean.

5 Comparison of tightness

For Approximate bounds

[13]:
```
interval_normal = []
for i in range(len(upper_conf)):
    interval_normal.append(upper_conf[i] - lower_conf[i])
interval_normal
```

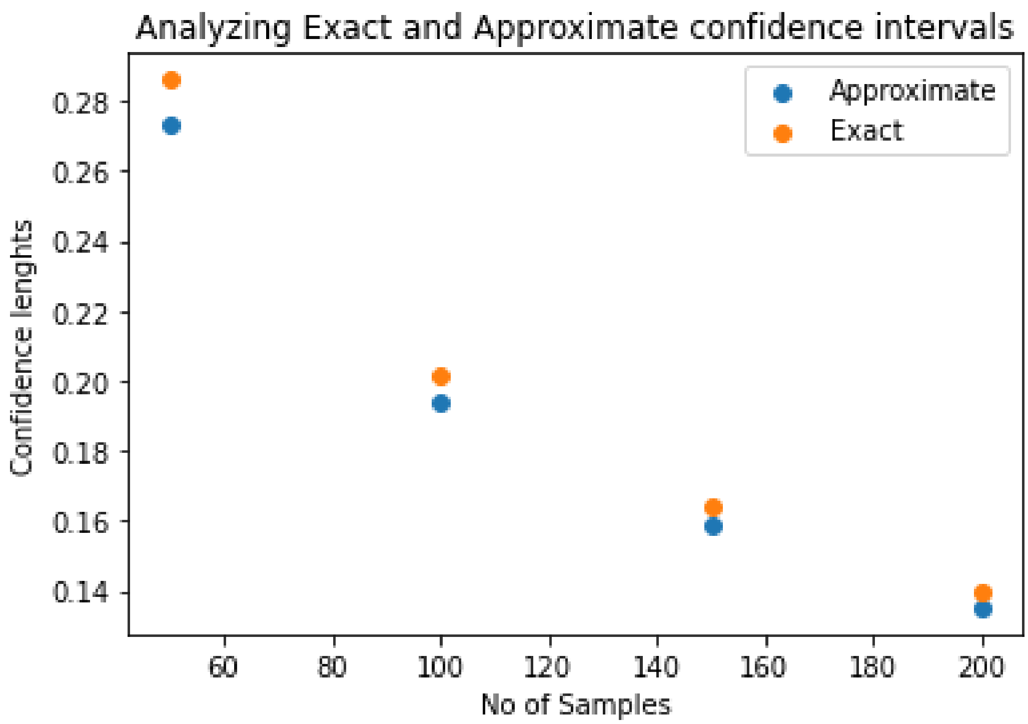
[13]:
```
[0.2736098490509563,
 0.19406612862136757,
 0.15887399228479004,
 0.13519490030641923]
```

For Exact bounds

[14]:
```
interval_exact = []
for i in range(len(upper_conf_exact)):
    interval_exact.append(upper_conf_exact[i] - lower_conf_exact[i])
interval_exact
```

[14]: [0.28605732381074533,
 0.20147524504376574,
 0.16414794980684966,
 0.1393301704299213]

[19]:
```
plt.scatter(N, interval_normal, label = 'Approximate')
plt.scatter(N, interval_exact, label = 'Exact')
plt.legend()
plt.title('Analyzing Exact and Approximate confidence intervals')
plt.xlabel('No of Samples')
plt.ylabel('Confidence lenghts')
plt.show()
```



From this graph we can see that Exact methods for estimating confidence intervals is less strict than the Normal approximate one. This is because, for approximate case, the distribution of the test statistic, then the interval is approximate. This often fail to let us know the exact distribution of the test statistic when the assumptions involved in the setup are not met.

4. For a Bernouli trial involving coin toss, assuming that we have tossed the coin N times resulting a Data

$$D = [x_1, x_2, \dots, x_N]$$

out of which we observed head up $(x=1)$ k times.

Let θ be the probability of head up.

$$\theta = P(X=1)$$

Given D, we need to find θ_{MLE}, θ_{MAP} and θ_{Bay} of θ.

Derivation of θ_{MLE}

$$\theta_{MLE} = \arg\max_{\theta} \log P(D|\theta) \qquad x_1, x_2 \dots, x_N \overset{iid}{\sim} ber(\theta)$$

$$\log P(D|\theta) = \log P(x_1, x_2 \dots, x_N | \theta)$$

$$= \log \prod_{i=1}^{N} P(x_i|\theta)$$

$$= \sum_{i=1}^{N} \log P(x_i|\theta)$$

$$= \sum_{i=1}^{N} \log \theta^{\mathbb{I}(x_i=1)} (1-\theta)^{\mathbb{I}(x_i=0)}$$

$$= \sum_{i=1}^{N} \mathbb{I}(x_i=1) \log\theta + \sum_{i=1}^{N} \mathbb{I}(x_i=0) \log(1-\theta)$$

$$= n\log\theta + (N-n)\log(1-\theta)$$

n = no. of times head occurs.

Given Log-likelihood take the first order derivative w.r.t θ

$$\frac{d}{d\theta} \log P(D|\theta) = \frac{n}{\theta} + \frac{N-n}{\theta-1} = 0$$

or, $\frac{n}{\theta} = \frac{N-n}{1-\theta}$

or, $\frac{1-\theta}{\theta} = \frac{N-n}{n}$ So, $\hat{\theta}_{MLE} = \frac{n}{N}$.

or, $\frac{1}{\theta} = \frac{N}{n}$

or, $\theta = \frac{n}{N}$

Derivation of θ_{MAP}

$$\theta_{MAP} = \arg\max_{\theta} \log P(\theta|D)$$

$$= \arg\max_{\theta} \left[\log P(\theta) + \log P(D|\theta)\right]$$

$$= \arg\max_{\theta} \left\{\log P(\theta) + \sum_{i=1}^{N} \log P(D_i|\theta)\right\}$$

Conjugate prior of Binomial distribution is Beta distribution.

So, $\theta \sim Beta(\alpha, \beta)$

$$\theta_{MAP} = \arg\max_{\theta} \left\{ \log \frac{\Gamma\alpha \, \Gamma\beta}{\Gamma\alpha+\beta} + (\alpha+1)\log\theta + (\beta-1)\log(1-\theta) + n\log\theta + (N-n)\log(1-\theta) \right\}.$$

derivative w.r.t θ.

$$\frac{\alpha-1}{\theta} - \frac{\beta-1}{1-\theta} + \frac{n}{\theta} - \frac{N-n}{1-\theta} = 0.$$

Bayesian Estimator Derivation :-

$\theta \sim \text{Beta}(\alpha, \beta)$

hence

$$P(\theta|D) \propto \theta^{n}(1-\theta)^{N-n} \;\; \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

n = no. of success.

So, $P(\theta|D) \propto \theta^{n+\alpha-1}(1-\theta)^{N-n+\beta-1}$

to make the density a new beta density,

$$P(\theta|D) = c \,.\, \theta^{n+\alpha-1}(1-\theta)^{N-n+\beta-1}$$

we need to choose c Such that $\int_{-\infty}^{\infty} P(\theta|D)\, d\theta = 1$

So, as $P(\theta|D)$ follows beta with $(n+\alpha)$ and $(N-n+\beta)$

then $c = \dfrac{\Gamma(n+\alpha+\beta)}{\Gamma(n+\alpha)\,\Gamma(N-n+\beta)}$

So, to find the Bayesian estimate, we need to find the mean of Beta distribution $(n+\alpha,\; N-n+\beta)$.

So, mean of Beta $= \dfrac{n+\alpha}{n+\alpha+N-n+\beta} = \boxed{\dfrac{n+\alpha}{N+\alpha+\beta} = \hat{\theta}_{Bay}}$

If $\alpha = \beta = 1$ then.

$$\hat{\theta}_{Bay} = \frac{n+1}{N+2}.$$

$$\frac{\alpha-1}{\theta} + \frac{n}{\theta} = \frac{\beta-1}{1-\theta} + \frac{N-n}{1-\theta}$$

or, $$\frac{\alpha+n-1}{\theta} = \frac{\beta+N-n-1}{1-\theta}$$

or, $$\frac{1-\theta}{\theta} = \frac{\beta+N-n-1}{\alpha+n-1}$$

or, $$\frac{1}{\theta} = \frac{\beta+N-\cancel{n}-1+\alpha+\cancel{n}-1}{\alpha+n-1}$$

or, $$\frac{1}{\theta} = \frac{\beta+N+\alpha-2.}{\alpha+n-1}$$

or, $$\theta = \frac{\alpha+n-1}{\beta+N+\alpha-2.}$$

$$\hat{\theta}_{MAP} = \frac{\alpha+n-1}{\beta+N+\alpha-2.}$$

If $\theta \sim \text{Beta}(1,1)$ then $\hat{\theta}_{MAP} = \frac{n}{N}.$

4.(b) As all the tosses are independent, the probability of $N+1$ th toss of being 1 is same as θ.

$$P[X_{N+1}=1]=\theta.$$

hence $\hat{\theta}_{MLE} = \frac{n}{N}$ $= P[X_{N+1}=1 \mid \hat{\theta}_{MLE}]$

$\hat{\theta}_{MAP} = \frac{n}{N}$ [Assuming prior. dist of θ follows Beta(1,1)]

$\hat{\theta}_{Bay} = \frac{n+1}{N+2}.$ "

4.(c) Now we need to perform a Bayesian prediction of the probability that head is up for the $N+1$ th toss

$$P[X_{N+1}=1 \mid D]$$

$$= \sum_{\theta} P[X_{N+1}=1, \theta \mid D]$$

$P[X_{N+1}=1 \mid D, \theta]$

$= P[X_{N+1}=1 \mid \theta]$

$$= \int_{\theta} P(\theta \mid D) \cdot P(X_{N+1} \mid D, \theta)\, d\theta$$

[Because once we know the prior estimate, we don't need to ~~calculate~~ consider the sample till N]

$$= \int_{\theta} P(\theta \mid D) \cdot P(X_{N+1}=1 \mid \theta)\, d\theta$$

$$= \int_{\theta} \theta \, P(\theta \mid D)\, d\theta = E_{P(\theta \mid D)}(\theta)$$

$$= \frac{n+\alpha}{N+\alpha+\beta}.$$

$$= \frac{n+1}{N+2} \quad [\text{if } \alpha=\beta=1]$$