DS6040: Homework 1: Probability Review and Priors

Diana McSpadden (hdm5s)

NOTE ON HOMEWORK PROCESS:

I learned a LOT working through these problems and found office hours, an SDS tutor (Andrew Graves), and fellow students very helpful as I tried to understand these problems and how to solve them. In particular, I often brainstormed and cross-checked results with David Fuentes, Jordan Hiatt, and Jing Huang. It is very interesting to me that David Fuentes and I selected different distributions for Question 5, but our posterior results and extra credit answers are almost identical, which helps support that the assumptions made with Bayesian analysis are valid.

Q1 (15 points):

You are a data science and are choosing between three approaches, A, B, and C, to a problem.

- With approach A you will spend a total of four days coding and running an algorithm and it will not produce useful results.
- With approach B you will spend a total of three days coding and running an algorithm and it will not produce useful results.
- With approach C you will spend one day coding and running an algorithm and it will give the results you are looking for.

You are equally likely to choose among unselected options.

What is the expected time in days for you to obtain the results you are looking for? What is the variance on this time?

Answer:

- P(A->C,5 days) = (1/3) * 0.5 = (1/6)
- P(A->B->C,8 days) = (1/3) * 0.5 = (1/6)
- P(B->A->C,8 days) = (1/3) * 0.5 = (1/6)
- P(B->C,4 days) = (1/3) * 0.5 = (1/6)
- P(C,1 day) = (1/3)

In [1]:
```
# confirm these sum to one
((1/6) * 4) + (1/3)
```

Out[1]:
```
1.0
```

In [2]:
```
print("Expected Days: {:1.2f} days".format((5 * (1/3) * 0.5) + (8 * (1/3) * 0.5) + (8 *
```

```
Expected Days: 4.50 days
```

In [3]:
```
import numpy as np
import pandas as pd
```

In [4]:
```
expected_days = 4.5 # from code above
lst_days = [5,8,8,4,1] # list of actual days for selections
lst_prob = [(1/6),(1/6),(1/6),(1/6),(1/3)] # list of probabilities for selections
var_sum = 0
for d,p in zip(lst_days,lst_prob):
    var_sum += p * (np.square(d - expected_days))

print("Variance: {:1.2f}".format(var_sum))
```

```
Variance: 8.25
```

Q2 (15 points)

Suppose if it is sunny or not in Charlottesville depends on the weather of the last three days. Show how this can be modeled as a Markov chain

Answer:

from what I read the Marchov Property says that "only the most recent point in the trajectory affects what happens next" (https://www.stat.auckland.ac.nz/~fewster/325/notes/ch8.pdf)

We can create a transition matrix for Sunny Today To Sunny Tomorrow - where I am stating that there is an 80% probability it is sunny tomorrow if it is sunny today, and a 60% chance it if sunny tomorrow if it is not sunny today.

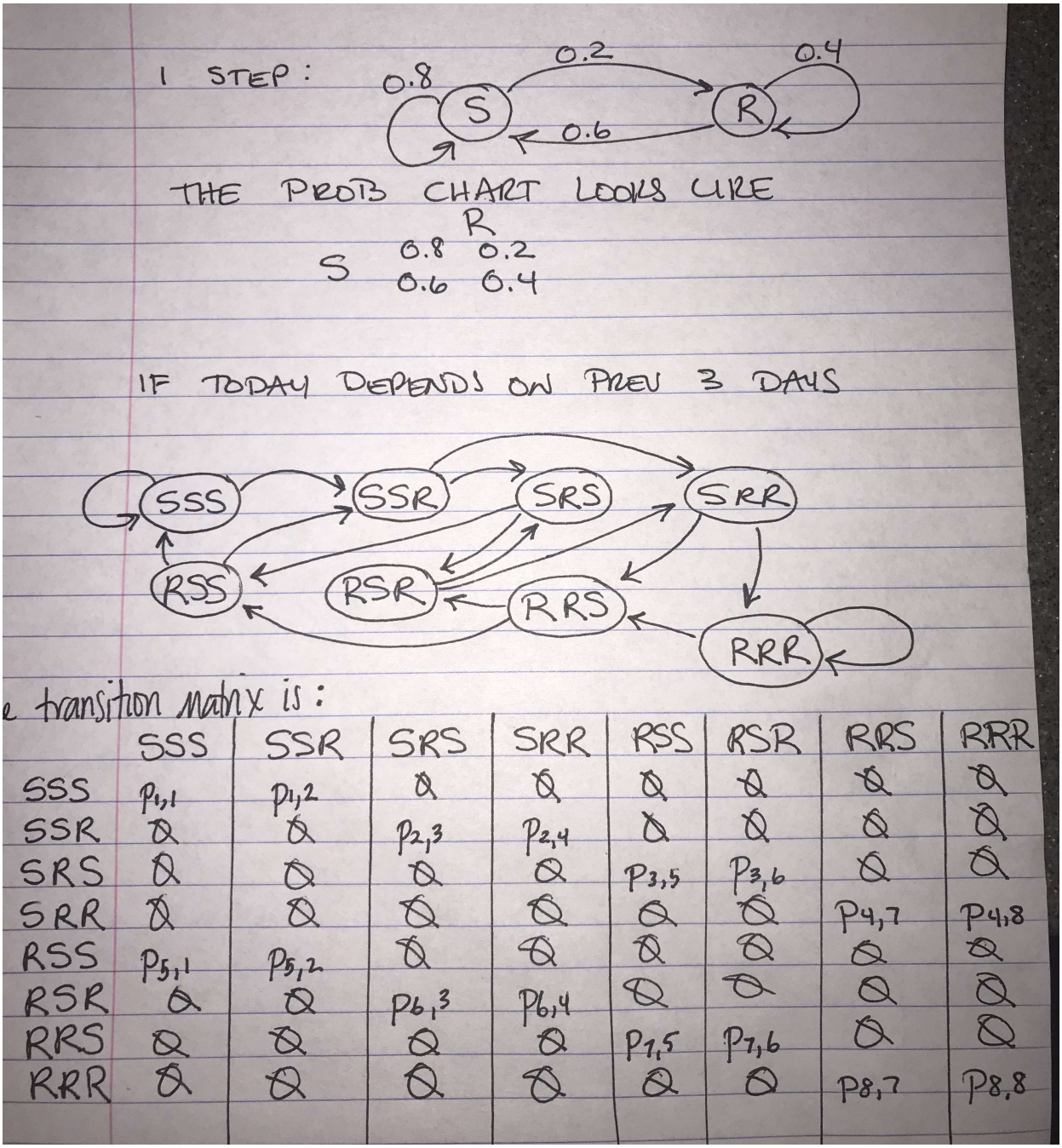
	Sunny Tomorrow	Not Sunny Tomorrow
Sunny Today	0.8	0.2
Not Sunny Today	0.6	0.4

The question does not ask, but the probability of 3 sunny days would be 0.8^3.

This question **does ask** for a multi-step transition diagram and table:

What are the 8 possible sequences of S and R for 3 days?

Here are my drawings of single step and the continual cycles from each state to itself and the other states:



Question 3 (15 points)

Assume a Gaussian distribution for observations, Xi, i = 1, . . . , N with unknown mean, M, and known variance 5.

Suppose the prior for M is Gaussian with variance 10.

How large a random sample must be taken (i.e., what is the minimum value for N) to specify an interval having unit length of 1 such that the probability that M lies in this interval is 0.95?

Answer:

- Use 1.96 for 95% CI sigma
- we have a Gaussian distribution (likelihood)

- unknown M, known variance
  - var = 5
  - prior variance = 10
- M is the same for prior and posterior

What is minimum value of N?

Work:

Using Bayes Theorem, the posterior variance is the Likelihood * Prior / Evidence =

sigma-prior^2 *sigma^2 / (sigma^2 + (N* sigma-prior^2)) =

(10 *5) / (5 + (10* N)) =

50 / 5 + 10N

Because we are using a normal/Gaussian distribution our 95% condidence range is [-1.96 *sigma, 1.96* sigma]

But we want this range to be unit length 1, so we need 1.96 * sigma to be 0.5

sigma = square root(50 / (5 + 10N))

sigma needs to equal 0.5 / 1.96

solving for N:

In [5]:
```
N = 49.675 / 0.65
print('Total number of sampled needed: {:2.0f}'.format(np.ceil(N)))
```

```
Total number of sampled needed: 77
```

N = 385 random samples

Question 4 (15 points)

You have started an online business selling books that are of interest to your customers. A publisher has just given you a large book with photos from famous 20th century photographers. You think this book will appeal to people who have bought art books, history books and coffee table books. In an initial offering of the new book you collect data on purchases of the new book and combine these data with data from the past purchases (see ArtHistBooks.csv).

Use Bayesian analysis to give the posterior probabilities for purchases of art books, history books and coffee table books, as well as, the separate probabilities for purchases of the new book given each possible combination of prior purchases of art books, history books and coffee table books.

Do this by first using beta priors with values of the hyperparameters that represent lack of prior information.

Then compute these probabilities again with beta priors that show strong weighting for low likelihood of a book purchase. Compare your results.

Answer:

In [6]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import binom
from scipy.stats import beta
```

In [7]:

```
def posterior_from_conjugate_prior(**kwargs):
    if kwargs['Likelihood_Dist_Type'] == 'Binomial':
        # Get the parameters for the likelihood and prior distribution from the key wor
        x = kwargs['x'] # This is state space of possible values for p = 'probability o
        n = kwargs['n'] # This is the number of Bernoili trials.
        k = kwargs['k'] # This is the number of 'successes'.
        a = kwargs['a'] # This is the parameter alpha for the prior Beta distribution
        b = kwargs['b'] # This is the parameter beta for the prior Beta distribution

        print(f'a_prime = {k + a}.')
        print(f'b_prime = {n - k + b}.')
        Likelihood = binom.pmf(p=x, n=n, k=k)
        Prior = beta.pdf(x=x, a=a, b=b)
        Posterior = beta.pdf(x=x, a=k+a, b=n-k+b)

        return [Prior, Likelihood, Posterior]

    else:
        print('Distribution type not supported.')

def printConfidenceInterval(**kwargs):
    post = kwargs['post']
    dx = kwargs['dx']
    CI_lb = kwargs['lb']
    CI_ub = kwargs['ub']

    cdf = np.zeros(len(post))
    for idx in range(len(post)):
        cdf[idx] = np.sum(post[:idx])*dx

    # CI
    a = x[np.argmin(np.abs(cdf-CI_lb))] # find the x-value where the cdf is closest to
    b = x[np.argmin(np.abs(cdf-CI_ub))]# find the x-value where the cdf is closest to C
    print('The probability is {:1.2f} that the value for p in the interval [{:1.2f},{:1

def printProbPlots(i_x, i_prior, i_likelihood, i_posterior, label):
    fig, axes = plt.subplots(1, 4, sharex=True, figsize=(21,4))
    fig.suptitle('{} Probability Plots'.format(label), fontsize=12, fontweight='bold')

    sns.lineplot(x=i_x, y=i_prior, color='red', ax=axes[0])
    axes[0].set(xlabel='x', ylabel='P')
    axes[0].set_title('{}: Prior PDF'.format(label),fontdict= {'fontsize': 9})

    sns.lineplot(x=i_x, y=i_likelihood, color='blue', ax=axes[1])
    axes[1].set(xlabel='x', ylabel='f(x)')
    axes[1].set_title('{}: Likelihood f(x)'.format(label),fontdict= {'fontsize': 9})

    sns.lineplot(x=i_x, y=i_posterior, color='orange', ax=axes[2])
    axes[2].set(xlabel='x', ylabel='P')
    axes[2].set_title('{}: Posterior PDF'.format(label),fontdict= {'fontsize': 9})
```

```
    sns.lineplot(x=i_x, y=i_prior/np.max(i_prior), color='red', label='Prior PDF', ax=a
    sns.lineplot(x=i_x, y=i_likelihood/np.max(i_likelihood), color='blue', label='Likel
    sns.lineplot(x=i_x, y=i_posterior/np.max(i_posterior), color='orange', label='Poste
    axes[3].set(xlabel='x')

def printPosteriorPlots(i_x, i_posterior, label):
    fig, axes = plt.subplots(1, 1, sharex=True, figsize=(10,3))
    fig.suptitle('{} Posterior Plot'.format(label), fontsize=12, fontweight='bold')

    sns.lineplot(x=i_x, y=i_posterior, color='orange')
```

In [8]:

```
df_Test_All = pd.read_csv(r'C:\Users\dianam\Documents\Personal\BayesianML\Mod2\ArtHistB

# total observations
cnt_observations = df_Test_All.shape[0]
cnt_purchasedA_And_A_And_New = df_Test_All.query("ArtBooks > 0 & HistoryBooks > 0 & Pur
cnt_purchasedA_And_T_And_New = df_Test_All.query("ArtBooks > 0 & TableBooks > 0 & Purch
cnt_purchasedH_And_T_And_New = df_Test_All.query("HistoryBooks > 0 & TableBooks > 0 & P

# count purchased Art, History, and Table Books
cnt_purchasedAHT = df_Test_All.query("ArtBooks > 0 & HistoryBooks > 0 & TableBooks > 0"
# percent who purchased Art, History, and Table Books
perc_purchasedAHT = cnt_purchasedAHT / cnt_observations

# count who purchased the New Book
cnt_purchasedNew = df_Test_All.query("Purchase > 0").shape[0]
# percent who purchased the New Book
perc_purchasedNew = cnt_purchasedNew / cnt_observations

# 3 book conjugate
# count who purchased Art, History, Table AND the New Book
cnt_purchasedAll = df_Test_All.query("ArtBooks > 0 & HistoryBooks > 0 & TableBooks > 0
# percent who purchased Art, History, Table AND the New Book
perc_purchasedAll = cnt_purchasedAll / cnt_observations

# percent who purchased New | purchased A, H, T
perc_purchasedNew_OfAHT = cnt_purchasedAll / cnt_purchasedAHT

# count who purchased A
cnt_purchasedA = df_Test_All.query("ArtBooks > 0").shape[0]
# percent who purchased A
perc_purchasedA = cnt_purchasedA / cnt_observations

# count who purchased A and New
cnt_purchasedA_New = df_Test_All.query("ArtBooks > 0 & Purchase > 0").shape[0]
# percent who purchased A and New
perc_purchasedA_New = cnt_purchasedA_New / cnt_observations
# one book conjugate
# percent who purchased A and New of those who purchased A
perc_purchasedA_New_OfA = cnt_purchasedA_New / cnt_purchasedA

# count who purchased H
cnt_purchasedH = df_Test_All.query("HistoryBooks > 0").shape[0]
# percent who purchased H
perc_purchasedH = cnt_purchasedH / cnt_observations

# count who purchased H and New
cnt_purchasedH_New = df_Test_All.query("HistoryBooks > 0 & Purchase > 0").shape[0]
```

```
# percent who purchased A and New
perc_purchasedH_New = cnt_purchasedH_New / cnt_observations
# one book conjugate
# percent who purchased H and New of those who purchased H
perc_purchasedH_New_OfH = cnt_purchasedH_New / cnt_purchasedH

# count who purchased T
cnt_purchasedT = df_Test_All.query("TableBooks > 0").shape[0]
# percent who purchased T
perc_purchasedT = cnt_purchasedT / cnt_observations

# count who purchased T and New
cnt_purchasedT_New = df_Test_All.query("TableBooks > 0 & Purchase > 0").shape[0]
# percent who purchased T and New
perc_purchasedT_New = cnt_purchasedT_New / cnt_purchasedT
# one book conjugate
# percent who purchased T and New of those who purchased T
perc_purchasedT_New_OfT = cnt_purchasedT_New / cnt_purchasedT

# now for the 2 book conjugates
# A and H
cnt_purchasedAH = df_Test_All.query("ArtBooks > 0 & HistoryBooks > 0").shape[0]
# percent who purchased A and H
perc_purchasedAH = cnt_purchasedAH / cnt_observations
# two book conjugate
cnt_purchasedAH_New = df_Test_All.query("ArtBooks > 0 & HistoryBooks > 0 & Purchase > 0
# percent who purchased T and New of those who purchased T
perc_purchased_New_OfAH = cnt_purchasedAH_New / cnt_purchasedAH

# A and T
cnt_purchasedAT = df_Test_All.query("ArtBooks > 0 & TableBooks > 0").shape[0]
# percent who purchased A and H
perc_purchasedAT = cnt_purchasedAT / cnt_observations
# two book conjugate
cnt_purchasedAT_New = df_Test_All.query("ArtBooks > 0 & TableBooks > 0 & Purchase > 0")
# percent who purchased T and New of those who purchased T
perc_purchased_New_OfAT = cnt_purchasedAT_New / cnt_purchasedAT

# T and H
cnt_purchasedTH = df_Test_All.query("HistoryBooks > 0 & TableBooks > 0").shape[0]
# percent who purchased A and H
perc_purchasedTH = cnt_purchasedTH / cnt_observations
# two book conjugate
cnt_purchasedTH_New = df_Test_All.query("HistoryBooks > 0 & TableBooks > 0 & Purchase >
# percent who purchased T and New of those who purchased T
perc_purchased_New_OfTH = cnt_purchasedTH_New / cnt_purchasedTH


print("# Observations: {}".format(cnt_observations))
print("_"*30)
print("# (N): {}".format(cnt_purchasedNew))
print("% P(N): {}".format(perc_purchasedNew))
print("# (N | A, H, T): {}".format(cnt_purchasedAll))
print("% P(N, A, H, T): {}".format(perc_purchasedAll))
print("% P(N | A, H, T): {}".format(perc_purchasedNew_OfAHT))

print("_"*30)
print("# (A,H): {}".format(cnt_purchasedAH))
print("% P(A,H): {}".format(perc_purchasedAH))
print("# (N | A,H): {}".format(cnt_purchasedAH_New))
print("% P(N | A, H): {}".format(perc_purchased_New_OfAH))
```

```
print("_"*30)
print("# (A,T): {}".format(cnt_purchasedAT))
print("% P(A,T): {}".format(perc_purchasedAT))
print("# (N | A,T): {}".format(cnt_purchasedAT_New))
print("% P(N | A, T): {}".format(perc_purchased_New_OfAT))

print("_"*30)
print("# (T,H): {}".format(cnt_purchasedTH))
print("% P(T,H): {}".format(perc_purchasedTH))
print("# (N | T,H): {}".format(cnt_purchasedTH_New))
print("% P(N | T, H): {}".format(perc_purchased_New_OfTH))
```

```
# Observations: 1000
______________________________
# (N): 89
% P(N): 0.089
# (N | A, H, T): 26
% P(N, A, H, T): 0.026
% P(N | A, H, T): 0.36619718309859156
______________________________
# (A,H): 171
% P(A,H): 0.171
# (N | A,H): 42
% P(N | A, H): 0.24561403508771928
______________________________
# (A,T): 125
% P(A,T): 0.125
# (N | A,T): 30
% P(N | A, T): 0.24
______________________________
# (T,H): 192
% P(T,H): 0.192
# (N | T,H): 40
% P(N | T, H): 0.20833333333333334
```

In [9]:

```
print("_"*30)
print("# (A,H,T): {}".format(cnt_purchasedAHT))
print("% P(A,H,T): {}".format(perc_purchasedAHT))

print("_"*30)
print("# (N, A, H, T): {}".format(cnt_purchasedAll))
print("% P(A, N, H, T): {}".format(perc_purchasedAll))
```

```
______________________________
# (A,H,T): 71
% P(A,H,T): 0.071
______________________________
# (N, A, H, T): 26
% P(A, N, H, T): 0.026
```

In [10]:

```
print("_"*30)
print("# (A): {}".format(cnt_purchasedA))
print("% P(A): {}".format(perc_purchasedA))

print("_"*30)
print("# (A,N): {}".format(cnt_purchasedA_New))
print("% P(A,N): {}".format(perc_purchasedA_New))
print("% P(N | A): {}".format(perc_purchasedA_New_OfA))
```

```
______________________________
# (A): 301
% P(A): 0.301

______________________________
# (A,N): 58
% P(A,N): 0.058
% P(N | A): 0.19269102990033224
```

In [11]:

```
print("_"*30)
print("# (H): {}".format(cnt_purchasedH))
print("% P(H): {}".format(perc_purchasedH))

print("_"*30)
print("# (H,N): {}".format(cnt_purchasedH_New))
print("% (H,N): {}".format(perc_purchasedH_New))
print("% P(N | H): {}".format(perc_purchasedH_New_OfH))
```

```
______________________________
# (H): 543
% P(H): 0.543

______________________________
# (H,N): 66
% (H,N): 0.066
% P(N | H): 0.12154696132596685
```

In [12]:

```
print("_"*30)
print("# (T): {}".format(cnt_purchasedT))
print("% P(T): {}".format(perc_purchasedT))
print("_"*30)
print("# (T,N): {}".format(cnt_purchasedT_New))
print("% P(T,N): {}".format(perc_purchasedT_New))
print("% P(N | T): {}".format(perc_purchasedT_New_OfT))
```

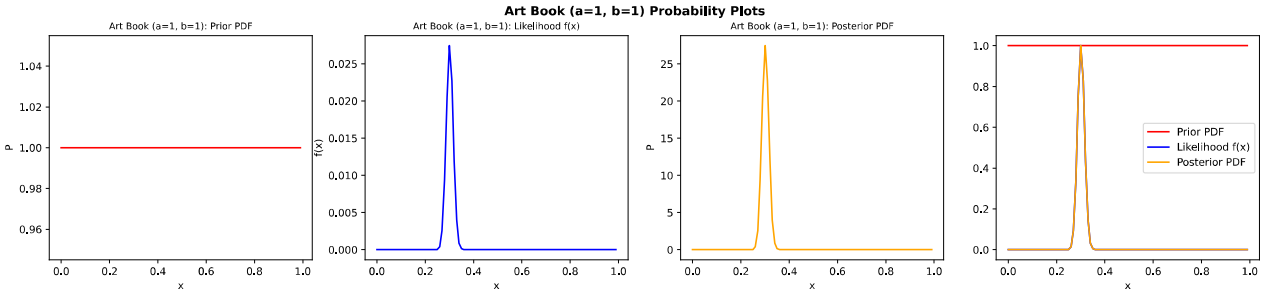
```
______________________________
# (T): 380
% P(T): 0.38

______________________________
# (T,N): 47
% P(T,N): 0.12368421052631579
% P(N | T): 0.12368421052631579
```

In [13]:

```
print("Probablity Distribution For Purchase of An Art Book: Uninformed Prior")

x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=1000,
    k=cnt_purchasedA,
    a=1,
    b=1)

denominator = np.sum(Prior*Likelihood) * (x[1] - x[0])
post = Likelihood*Prior / denominator

printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "Art Book (a=1, b=1)")
```

```
Probablity Distribution For Purchase of An Art Book: Uninformed Prior
```

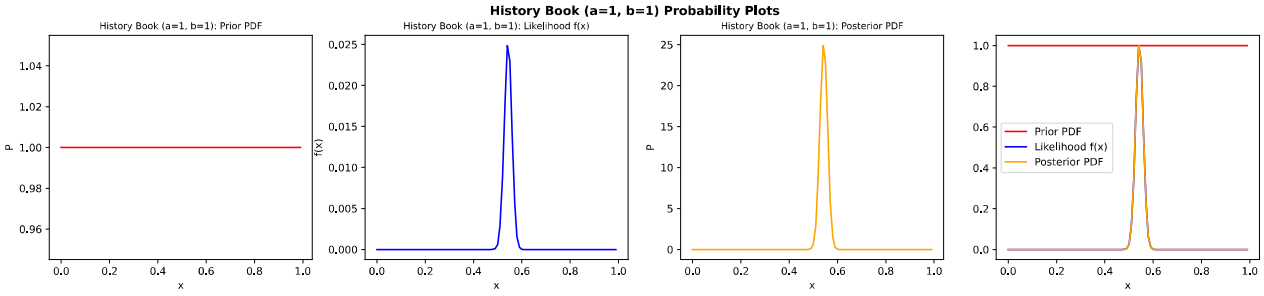
```
a_prime = 302.
b_prime = 700.
The probability is 0.95 that the value for p in the interval [0.28,0.34].
```



In [14]:

```
print("Probablity Distribution For Purchase of A History Book: Uninformed Prior")

x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=1000,
    k=cnt_purchasedH,
    a=1,
    b=1)

dx = x[1] - x[0]
denominator = np.sum(Prior*Likelihood*dx)
post = Likelihood*Prior / denominator
printConfidenceInterval(post = post, dx = dx, lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "History Book (a=1, b=1)")
```

```
Probablity Distribution For Purchase of A History Book: Uninformed Prior
a_prime = 544.
b_prime = 458.
The probability is 0.95 that the value for p in the interval [0.52,0.58].
```

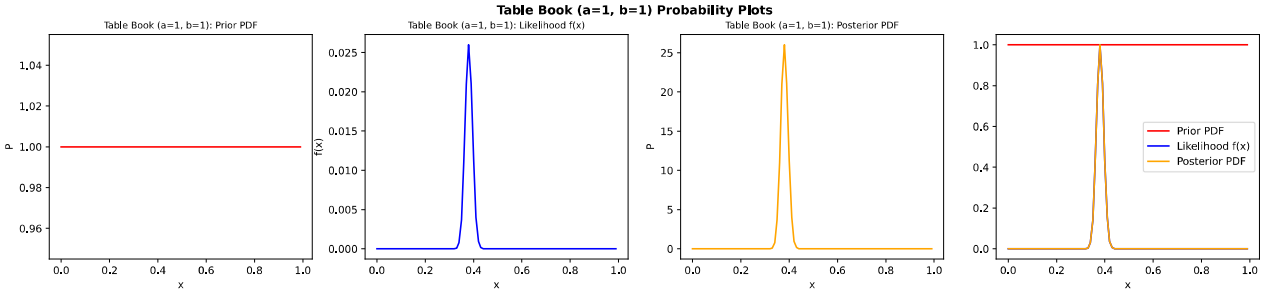


In [15]:

```
print("Probablity Distribution For Purchase of A Table Book: Uninformed Prior")

x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=1000,
    k=cnt_purchasedT,
    a=1,
    b=1)

denominator = np.sum(Prior*Likelihood*dx)
post = Likelihood*Prior / denominator
```

```
printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "Table Book (a=1, b=1)")
```

```
Probablity Distribution For Purchase of A Table Book: Uninformed Prior
a_prime = 381.
b_prime = 621.
The probability is 0.95 that the value for p in the interval [0.35,0.42].
```



Posterior Probabilities For:

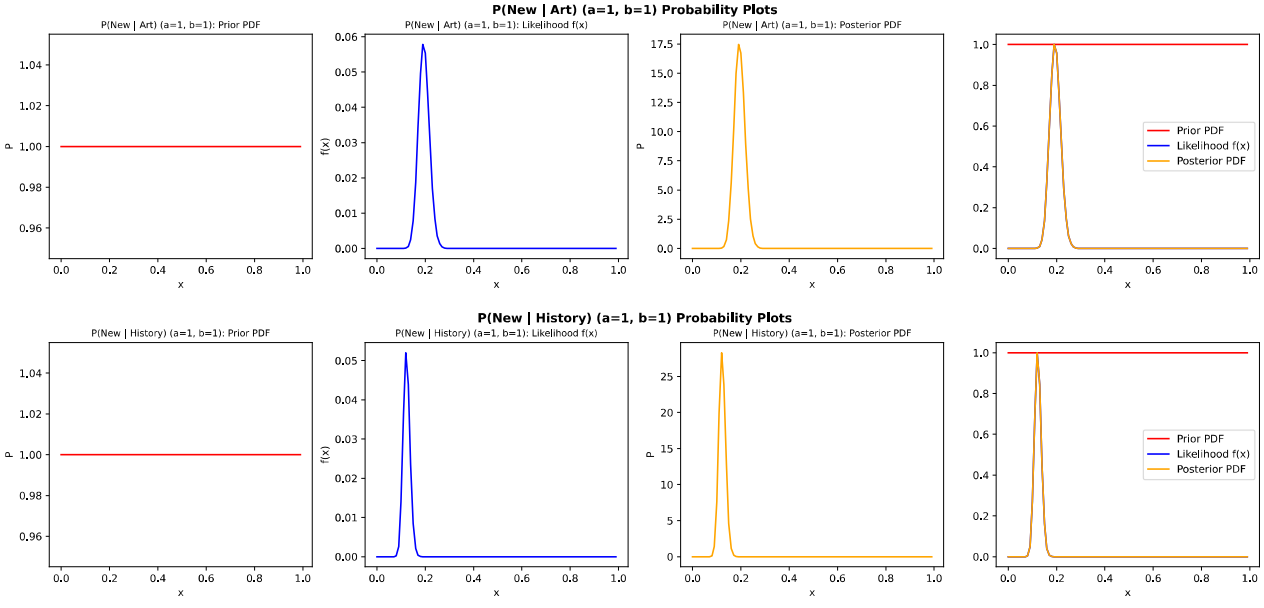
- P(New | Art)
- P(New | History)
- P(New | Table)

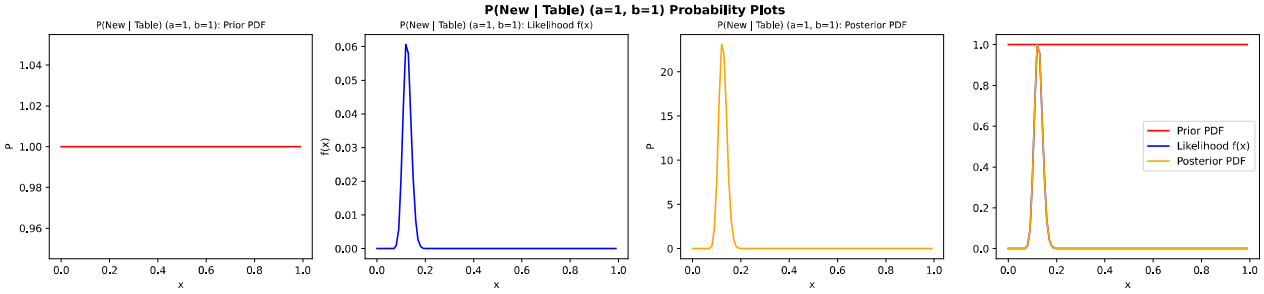
In [16]:

```
print("Posterior Probablity Distribution For Purchase of the New Book Given Purchase of

x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=cnt_purchasedA,
    k=cnt_purchasedA_New,
    a=1,
    b=1)

denominator = np.sum(Prior*Likelihood*dx)
post = Likelihood*Prior / denominator
printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "P(New | Art) (a=1, b=1)")

print("-"*50)

print("Posterior Probablity Distribution For Purchase of the New Book Given Purchase of

x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=cnt_purchasedH,
    k=cnt_purchasedH_New,
    a=1,
    b=1)

denominator = np.sum(Prior*Likelihood*dx)
post = Likelihood*Prior / denominator
printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "P(New | History) (a=1, b=1)")

print("-"*50)
```

```
print("Posterior Probablity Distribution For Purchase of the New Book Given Purchase of

x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=cnt_purchasedT,
    k=cnt_purchasedT_New,
    a=1,
    b=1)

denominator = np.sum(Prior*Likelihood*dx)
post = Likelihood*Prior / denominator
printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "P(New | Table) (a=1, b=1)")
```

```
Posterior Probablity Distribution For Purchase of the New Book Given Purchase of Art Boo
k: Uninformed Priors
a_prime = 59.
b_prime = 244.
The probability is 0.95 that the value for p in the interval [0.16,0.25].
--------------------------------------------------
Posterior Probablity Distribution For Purchase of the New Book Given Purchase of History
Book: Uninformed Priors
a_prime = 67.
b_prime = 478.
The probability is 0.95 that the value for p in the interval [0.10,0.16].
--------------------------------------------------
Posterior Probablity Distribution For Purchase of the New Book Given Purchase of Table B
ook: Uninformed Priors
a_prime = 48.
b_prime = 334.
The probability is 0.95 that the value for p in the interval [0.10,0.17].
```





Posterior Probabilities For:

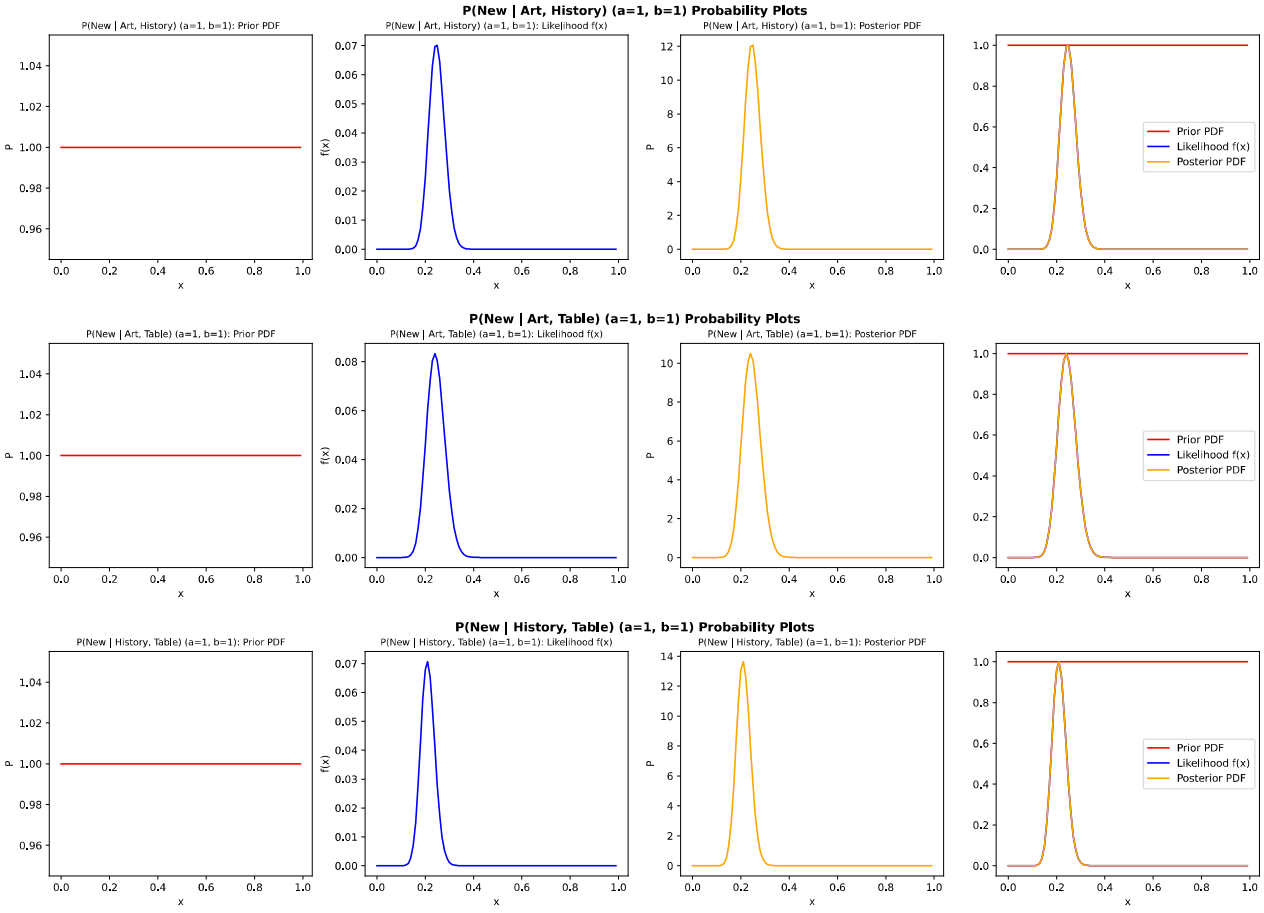
- P(New | Art, History)
- P(New | Art, Table)
- P(New | History, Table)

In [17]:

```
print("Posterior Probablity Distribution For Purchase of the New Book Given Purchase of

x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=cnt_purchasedAH,
    k=cnt_purchasedAH_New,
    a=1,
    b=1)

denominator = np.sum(Prior*Likelihood*dx)
post = Likelihood*Prior / denominator
printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "P(New | Art, History) (a=1, b=1)")

print("-"*50)

print("Posterior Probablity Distribution For Purchase of the New Book Given Purchase of

x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=cnt_purchasedAT,
    k=cnt_purchasedAT_New,
    a=1,
    b=1)

denominator = np.sum(Prior*Likelihood*dx)
post = Likelihood*Prior / denominator
printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "P(New | Art, Table) (a=1, b=1)")

print("-"*50)

print("Posterior Probablity Distribution For Purchase of the New Book Given Purchase of

x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
```

```
    n=cnt_purchasedTH,
    k=cnt_purchasedTH_New,
    a=1,
    b=1)

denominator = np.sum(Prior*Likelihood*dx)
post = Likelihood*Prior / denominator
printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "P(New | History, Table) (a=1, b=1)")
```

```
Posterior Probablity Distribution For Purchase of the New Book Given Purchase of Art and
History Book: Uninformed Priors
a_prime = 43.
b_prime = 130.
The probability is 0.95 that the value for p in the interval [0.19,0.32].
--------------------------------------------------
Posterior Probablity Distribution For Purchase of the New Book Given Purchase of Art and
Table Book: Uninformed Priors
a_prime = 31.
b_prime = 96.
The probability is 0.95 that the value for p in the interval [0.18,0.33].
--------------------------------------------------
Posterior Probablity Distribution For Purchase of the New Book Given Purchase of History
and Table Book: Uninformed Priors
a_prime = 41.
b_prime = 153.
The probability is 0.95 that the value for p in the interval [0.16,0.28].
```

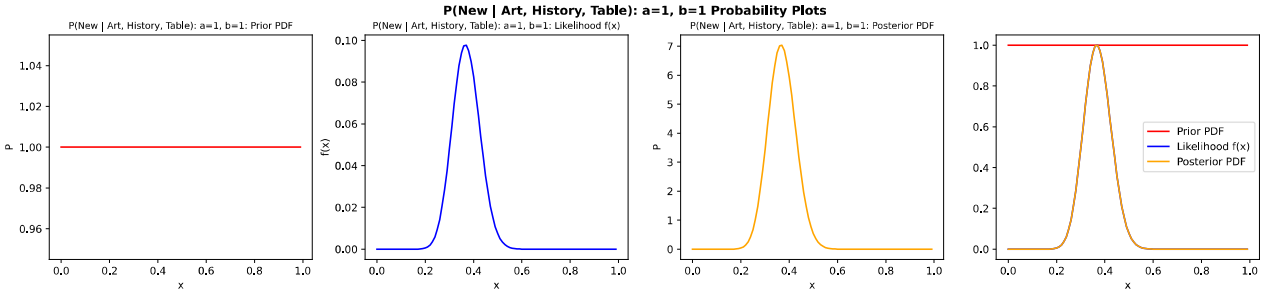


Posterior Probability For:

- P(New | Art, History, Table)

In [18]:
```
print("Posterior Probablity Distribution For Purchase of the New Book Given Purchase of

x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=cnt_purchasedAHT,
    k=cnt_purchasedAll,
    a=1,
    b=1)

denominator = np.sum(Prior*Likelihood*dx)
post = Likelihood*Prior / denominator
printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "P(New | Art, History, Table): a=1, b=1")
```

```
Posterior Probablity Distribution For Purchase of the New Book Given Purchase of Art, Hi
story, and Table Books: Uninformed Priors
a_prime = 27.
b_prime = 46.
The probability is 0.95 that the value for p in the interval [0.27,0.49].
```



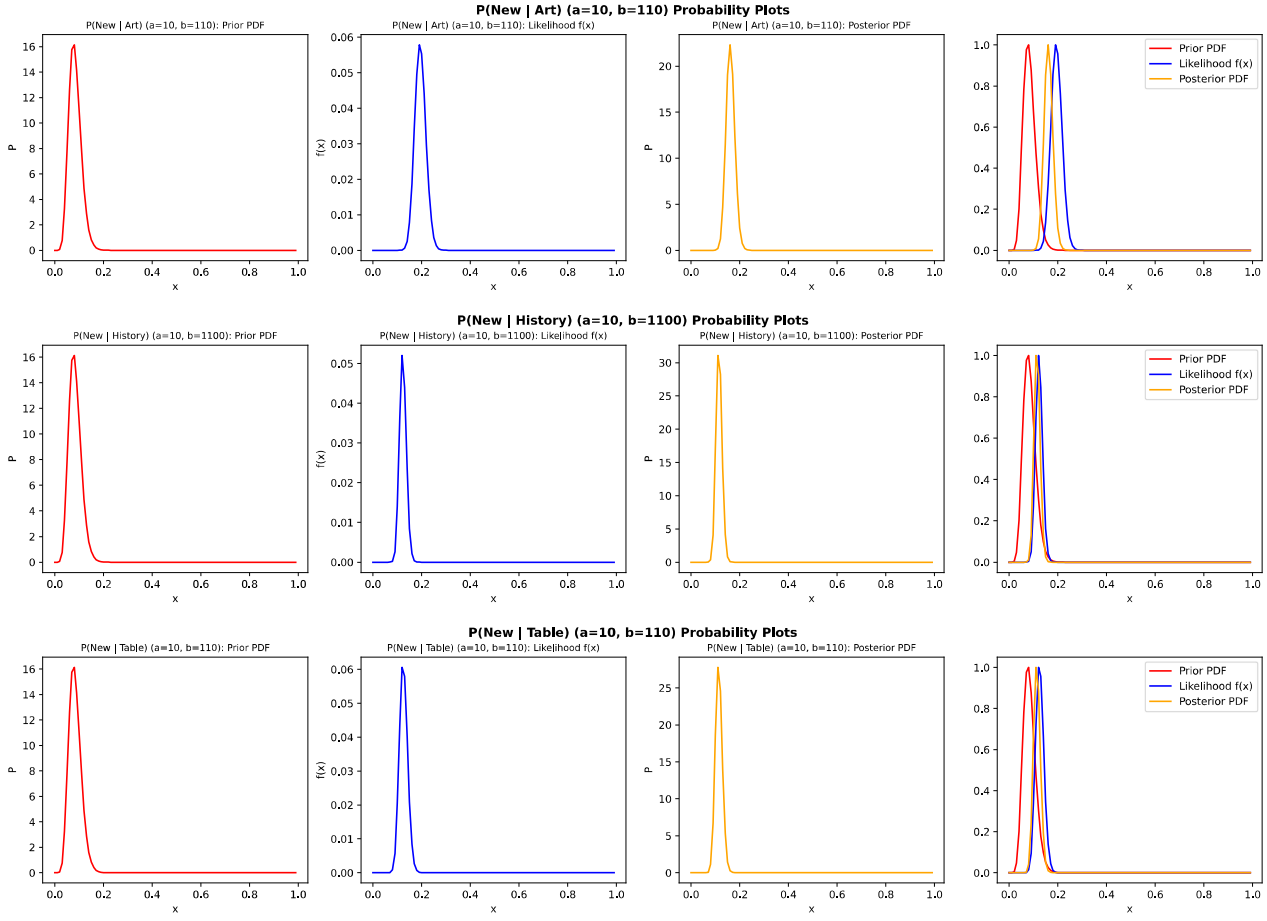
Posterior Probabilities for Purchasing New Book for all combinations with an informed prior indicating a low likelihood of purchasing the new book:

I used alpha = 10, and beta = 110 to represent a mean of 8.3%.

In [19]:
```
print("Posterior Probablity Distribution For Purchase of the New Book Given Purchase of

x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=cnt_purchasedA,
    k=cnt_purchasedA_New,
    a=10,
    b=110)

denominator = np.sum(Prior*Likelihood*dx)
post = Likelihood*Prior / denominator
printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "P(New | Art) (a=10, b=110)")

print("-"*50)

print("Posterior Probablity Distribution For Purchase of the New Book Given Purchase of
```

```
x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=cnt_purchasedH,
    k=cnt_purchasedH_New,
    a=10,
    b=110)

denominator = np.sum(Prior*Likelihood*dx)
post = Likelihood*Prior / denominator
printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "P(New | History) (a=10, b=1100)")

print("-"*50)

print("Posterior Probablity Distribution For Purchase of the New Book Given Purchase of

x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=cnt_purchasedT,
    k=cnt_purchasedT_New,
    a=10,
    b=110)

denominator = np.sum(Prior*Likelihood*dx)
post = Likelihood*Prior / denominator
printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "P(New | Table) (a=10, b=110)")

print("-"*50)
```

```
Posterior Probablity Distribution For Purchase of the New Book Given Purchase of Art Boo
k: Informed Priors
a_prime = 68.
b_prime = 353.
The probability is 0.95 that the value for p in the interval [0.13,0.20].
--------------------------------------------------
Posterior Probablity Distribution For Purchase of the New Book Given Purchase of History
Book: Informed Priors
a_prime = 76.
b_prime = 587.
The probability is 0.95 that the value for p in the interval [0.10,0.15].
--------------------------------------------------
Posterior Probablity Distribution For Purchase of the New Book Given Purchase of Table B
ook: Informed Priors
a_prime = 57.
b_prime = 443.
The probability is 0.95 that the value for p in the interval [0.09,0.15].
--------------------------------------------------
```

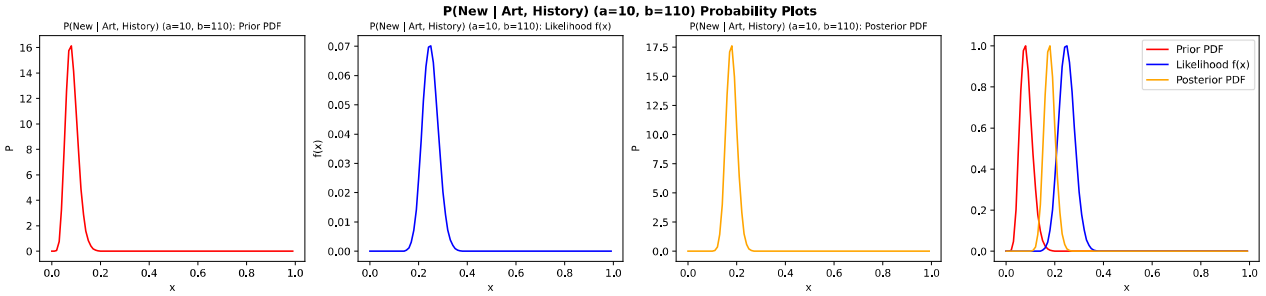


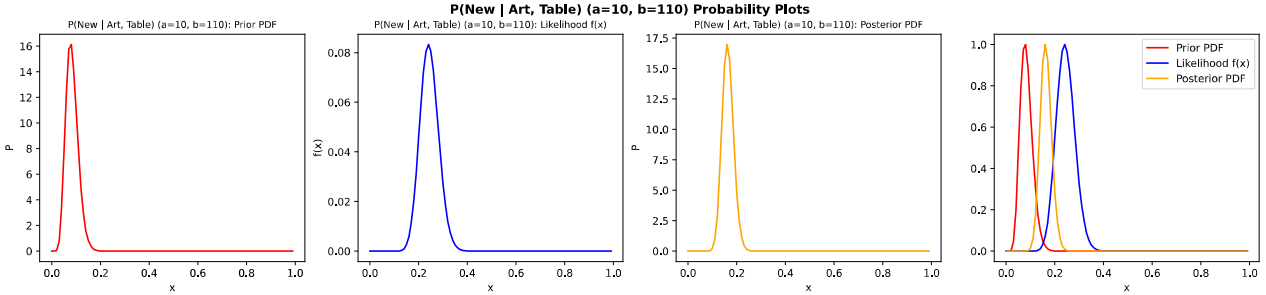
In [20]:

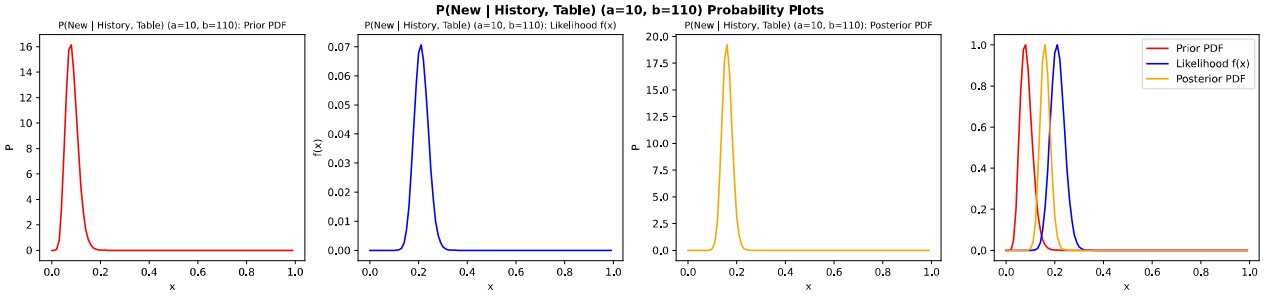
```
print("Posterior Probablity Distribution For Purchase of the New Book Given Purchase of

x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=cnt_purchasedAH,
    k=cnt_purchasedAH_New,
    a=10,
    b=110)

denominator = np.sum(Prior*Likelihood*dx)
post = Likelihood*Prior / denominator
printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "P(New | Art, History) (a=10, b=110)")

print("-"*50)

print("Posterior Probablity Distribution For Purchase of the New Book Given Purchase of

x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=cnt_purchasedAT,
    k=cnt_purchasedAT_New,
    a=10,
    b=110)

denominator = np.sum(Prior*Likelihood*dx)
```

```
post = Likelihood*Prior / denominator
printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "P(New | Art, Table) (a=10, b=110)")

print("-"*50)

print("Posterior Probablity Distribution For Purchase of the New Book Given Purchase of

x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=cnt_purchasedTH,
    k=cnt_purchasedTH_New,
    a=10,
    b=110)

denominator = np.sum(Prior*Likelihood*dx)
post = Likelihood*Prior / denominator
printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "P(New | History, Table) (a=10, b=110)")
```

```
Posterior Probablity Distribution For Purchase of the New Book Given Purchase of Art and
History Book: Informed Priors
a_prime = 52.
b_prime = 239.
The probability is 0.95 that the value for p in the interval [0.14,0.23].
--------------------------------------------------
Posterior Probablity Distribution For Purchase of the New Book Given Purchase of Art and
Table Book: Informed Priors
a_prime = 40.
b_prime = 205.
The probability is 0.95 that the value for p in the interval [0.12,0.22].
--------------------------------------------------
Posterior Probablity Distribution For Purchase of the New Book Given Purchase of History
and Table Book: Informed Priors
a_prime = 50.
b_prime = 262.
The probability is 0.95 that the value for p in the interval [0.13,0.21].
```



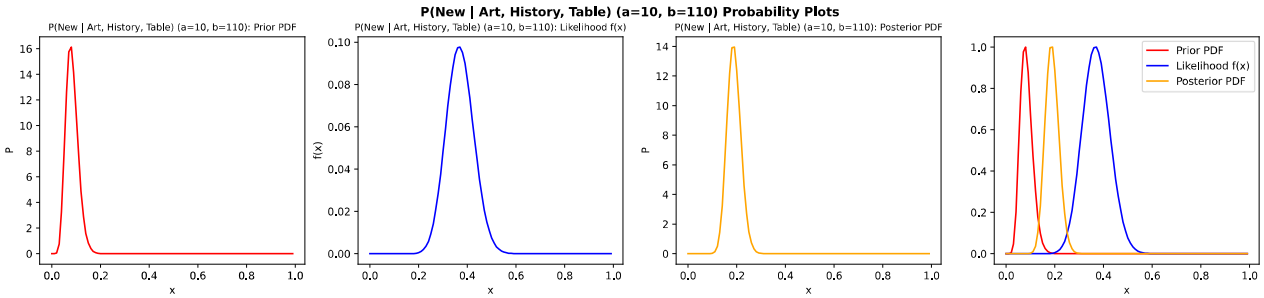




In [21]:

```
print("Posterior Probablity Distribution For Purchase of the New Book Given Purchase of
x = np.arange(0, 1, 0.01)
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(
    Likelihood_Dist_Type='Binomial',
    x=x,
    n=cnt_purchasedAHT,
    k=cnt_purchasedAll,
    a=10,
    b=110)

denominator = np.sum(Prior*Likelihood*dx)
post = Likelihood*Prior / denominator
printConfidenceInterval(post = post, dx = x[1] - x[0], lb=0.025, ub=0.975)
printProbPlots(x, Prior, Likelihood, post, "P(New | Art, History, Table) (a=10, b=110)"
```

```
Posterior Probablity Distribution For Purchase of the New Book Given Purchase of Art, Hi
story, and Table Books: Informed Priors
a_prime = 36.
b_prime = 155.
The probability is 0.95 that the value for p in the interval [0.14,0.25].
```



Question 4 Comments: From both the confidence intervals and posterior probability distributions one can see that the probability of purchasing the new books increases as the subpopulation of book purchasers becomes more targeted. The CI of probability has the highest upper bound for customers who purchased Art, History and Table Books.

The change in probability from the prior to the posterior using Bayes Theorem is also of interest as the likelihood function takes into account the differing subpopulation behaviors. From the plots and CI's using informed priors, purchasing both art and history books appear to have the greatest predictive impact on the purchasing of the new book with the smallest CI range and almost as large a max probability in the CI.

Question 5 (15 points)

The data set CHDdata.csv contains cases of coronary heart disease (CHD) and variables associated with the patient's condition: systolic blood pressure, yearly tobacco use (in kg), low density

lipoprotein (ldl), adiposity, family history (0 or 1), type A personality score (typea), obesity (body mass index), alcohol use, age, and the diagnosis of CHD (0 or 1).

Perform a Bayesian analysis of these data that finds the posterior marginal probability distributions for the means for the data of patients with and without CHD.

You should first standard scale (subtract the mean and divide by the standard deviation) all the numeric variables (remove family history and do not scale CHD). Then separate the data into two sets, one for patients with CHD and one for patients without CHD.

Your priors for both groups should assume means of 0 for all variables and a correlation of 0 between all pairs of variables. You should assume all variances for the variables are 1.

Use a prior alpha equal to one plus the number of predictor variables.

Compute and compare the Bayesian estimates for the posterior means for each group.

example of the question is: how confident are we that tobacco use is positive we will have chd

- do single variable analysis for each variable.

need to compute the posterior CI

we are given information about priors - we are given prior mean and stdv.

- priors are mean 0, stdv 1
- Likelihood: mean = 0, variance = 1

- need to use the formulas for posterior means and stdv - will need to select the right approach for the conjugate prior: https://en.wikipedia.org/wiki/Conjugate_prior

Answer:

In [22]:

```
X_cols = ['sbp','tobacco','ldl','adiposity','typea','obesity','alcohol','age']

df_Q5_X = pd.read_csv(r'C:\Users\dianam\Documents\Personal\BayesianML\Mod2\CHDdata.csv'
print(df_Q5_X.head())

df_Q5_y = pd.read_csv(r'C:\Users\dianam\Documents\Personal\BayesianML\Mod2\CHDdata.csv'
print(df_Q5_y.head())
```

```
   sbp  tobacco   ldl  adiposity  typea  obesity  alcohol  age
0  160    12.00  5.73      23.11     49    25.30    97.20   52
1  144     0.01  4.41      28.61     55    28.87     2.06   63
2  118     0.08  3.48      32.28     52    29.14     3.81   46
3  170     7.50  6.41      38.03     51    31.99    24.26   58
4  134    13.60  3.50      27.78     60    25.99    57.34   49
   chd
0    1
1    1
2    0
3    1
4    1
```

In [23]:
```
for key in df_Q5_X.keys()[0:8]:
    df_Q5_X[key] = df_Q5_X[key] - np.mean(df_Q5_X[key])
    df_Q5_X[key] = df_Q5_X[key] /df_Q5_X[key].std()

df_Q5_X.head()
```

Out[23]:

	sbp	tobacco	ldl	adiposity	typea	obesity	alcohol	age
0	1.057417	1.821099	0.477894	-0.295183	-0.418017	-0.176594	3.274189	0.628654
1	0.276789	-0.789382	-0.159507	0.411694	0.193134	0.670646	-0.612081	1.381617
2	-0.991731	-0.774141	-0.608585	0.883374	-0.112441	0.734723	-0.540597	0.217947
3	1.545310	0.841352	0.806252	1.622382	-0.214300	1.411091	0.294742	1.039361
4	-0.211103	2.169453	-0.598928	0.305020	0.702427	-0.012842	1.645991	0.423301

In [24]:
```
df_Scaled_X_y = df_Q5_X
df_Scaled_X_y['chd'] = df_Q5_y['chd']
cnt_records = df_Scaled_X_y.shape[0]
print('Count of records: {}'.format(cnt_records))
df_Scaled_X_y.head()
```

```
Count of records: 462
```

Out[24]:

	sbp	tobacco	ldl	adiposity	typea	obesity	alcohol	age	chd
0	1.057417	1.821099	0.477894	-0.295183	-0.418017	-0.176594	3.274189	0.628654	1
1	0.276789	-0.789382	-0.159507	0.411694	0.193134	0.670646	-0.612081	1.381617	1
2	-0.991731	-0.774141	-0.608585	0.883374	-0.112441	0.734723	-0.540597	0.217947	0
3	1.545310	0.841352	0.806252	1.622382	-0.214300	1.411091	0.294742	1.039361	1
4	-0.211103	2.169453	-0.598928	0.305020	0.702427	-0.012842	1.645991	0.423301	1

Split into chd and no chd

In [25]:
```
df_Scaled_chd1 = df_Scaled_X_y.query('chd == 1')
cnt_chd1 = df_Scaled_chd1.shape[0]
print('Count of records with chd: {}'.format(cnt_chd1))

df_Scaled_chd0 = df_Scaled_X_y.query('chd == 0')
cnt_chd0 = df_Scaled_chd0.shape[0]
print('Count of records without chd: {}'.format(cnt_chd0))
```

```
Count of records with chd: 160
Count of records without chd: 302
```

For each columns I want to calculate the "posterior marginal probability distributions for the means for the data of patients with and without CHD.

Without CHD:

In [26]:
```
lst_cols = df_Scaled_chd0.columns[0:8]
print('List of variable columns: {}'.format(lst_cols))
lst_means_chd0 = df_Scaled_chd0.describe().loc['mean'].values[0:8]
```

```
print('List of chd = 0 variable means: {}'.format(lst_means_chd0))

lst_mins_chd0 = df_Scaled_chd0.describe().loc['min'].values[0:8]
print('List of chd = 0 variable mins: {}'.format(lst_mins_chd0))

lst_maxs_chd0 = df_Scaled_chd0.describe().loc['max'].values[0:8]
print('List of chd = 0 variable maxs: {}'.format(lst_maxs_chd0))

lst_stdv_chd0 = df_Scaled_chd0.describe().loc['std'].values[0:8]
print('List of chd = 0 variable stdv: {}'.format(lst_stdv_chd0))

df_Scaled_chd0.describe()
```

```
List of variable columns: Index(['sbp', 'tobacco', 'ldl', 'adiposity', 'typea', 'obesit
y', 'alcohol',
       'age'],
      dtype='object')
List of chd = 0 variable means: [-0.13985805 -0.21792053 -0.19126202 -0.1847682  -0.0750
032  -0.07277776
 -0.0454652  -0.27118382]
List of chd = 0 variable mins: [-1.82114862 -0.79155896 -1.81578446 -2.3991074  -4.08492
557 -1.96837731
 -0.69622781 -1.90403862]
List of chd = 0 variable maxs: [3.69203694 3.56287064 5.11353927 2.14033074 2.43402301
4.87362266
 5.23856454 1.45006817]
List of chd = 0 variable stdv: [0.87747244 0.78643073 0.90319927 0.99899461 0.96945277
0.97082385
 0.95990682 1.01873552]
```

Out[26]:

	sbp	tobacco	ldl	adiposity	typea	obesity	alcohol	age
count	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000
mean	-0.139858	-0.217921	-0.191262	-0.184768	-0.075003	-0.072778	-0.045465	-0.271184
std	0.877472	0.786431	0.903199	0.998995	0.969453	0.970824	0.959907	1.018736
min	-1.821149	-0.791559	-1.815784	-2.399107	-4.084926	-1.968377	-0.696228	-1.904039
25%	-0.698996	-0.791559	-0.812602	-1.014913	-0.621734	-0.816771	-0.675395	-1.082625
50%	-0.308682	-0.566217	-0.367145	-0.100471	-0.061512	-0.112517	-0.449302	-0.192760
75%	0.276789	0.122871	0.264220	0.584879	0.600569	0.479602	0.219684	0.543090
max	3.692037	3.562871	5.113539	2.140331	2.434023	4.873623	5.238565	1.450068

With CHD

In [27]:

```
lst_means_chd1 = df_Scaled_chd1.describe().loc['mean'].values[0:8]
print('List of chd = 1 variable means: {}'.format(lst_means_chd1))

lst_mins_chd1 = df_Scaled_chd1.describe().loc['min'].values[0:8]
print('List of chd = 1 variable mins: {}'.format(lst_mins_chd1))

lst_maxs_chd1 = df_Scaled_chd1.describe().loc['max'].values[0:8]
print('List of chd = 1 variable maxs: {}'.format(lst_maxs_chd1))

lst_stdv_chd1 = df_Scaled_chd1.describe().loc['std'].values[0:8]
print('List of chd = 1 variable stdv: {}'.format(lst_stdv_chd1))
```

```
df_Scaled_chd1.describe()
```

```
List of chd = 1 variable means: [0.26398207 0.411325   0.36100706 0.34874997 0.14156853
0.13736803
 0.08581557 0.51185947]
List of chd = 1 variable mins: [-1.77235937 -0.79155896 -1.54054303 -2.05852102 -3.37191
557 -2.69221012
 -0.69622781 -1.76713631]
List of chd = 1 variable maxs: [3.88719395 6.00135122 4.54857002 2.1955957  2.53588158
4.66952554
 5.31617557 1.45006817]
List of chd = 1 variable stdv: [1.15520596 1.21165107 1.07444258 0.90711594 1.04370831
1.04215865
 1.06944321 0.72894453]
```

Out[27]:

	sbp	tobacco	ldl	adiposity	typea	obesity	alcohol	age
count	160.000000	160.000000	160.000000	160.000000	160.000000	160.000000	160.000000	160.000000
mean	0.263982	0.411325	0.361007	0.348750	0.141569	0.137368	0.085816	0.511859
std	1.155206	1.211651	1.074443	0.907116	1.043708	1.042159	1.069443	0.728945
min	-1.772359	-0.791559	-1.540543	-2.058521	-3.371916	-2.692210	-0.696228	-1.767136
25%	-0.528233	-0.464977	-0.386461	-0.249879	-0.545340	-0.571736	-0.676825	-0.004519
50%	-0.015946	0.107631	0.156779	0.385347	0.193134	0.102259	-0.355965	0.697105
75%	0.984233	0.993757	0.889549	1.051418	0.804286	0.649287	0.307916	1.107812
max	3.887194	6.001351	4.548570	2.195596	2.535882	4.669526	5.316176	1.450068

In [28]:
```
df_Scaled_chd1.columns
```

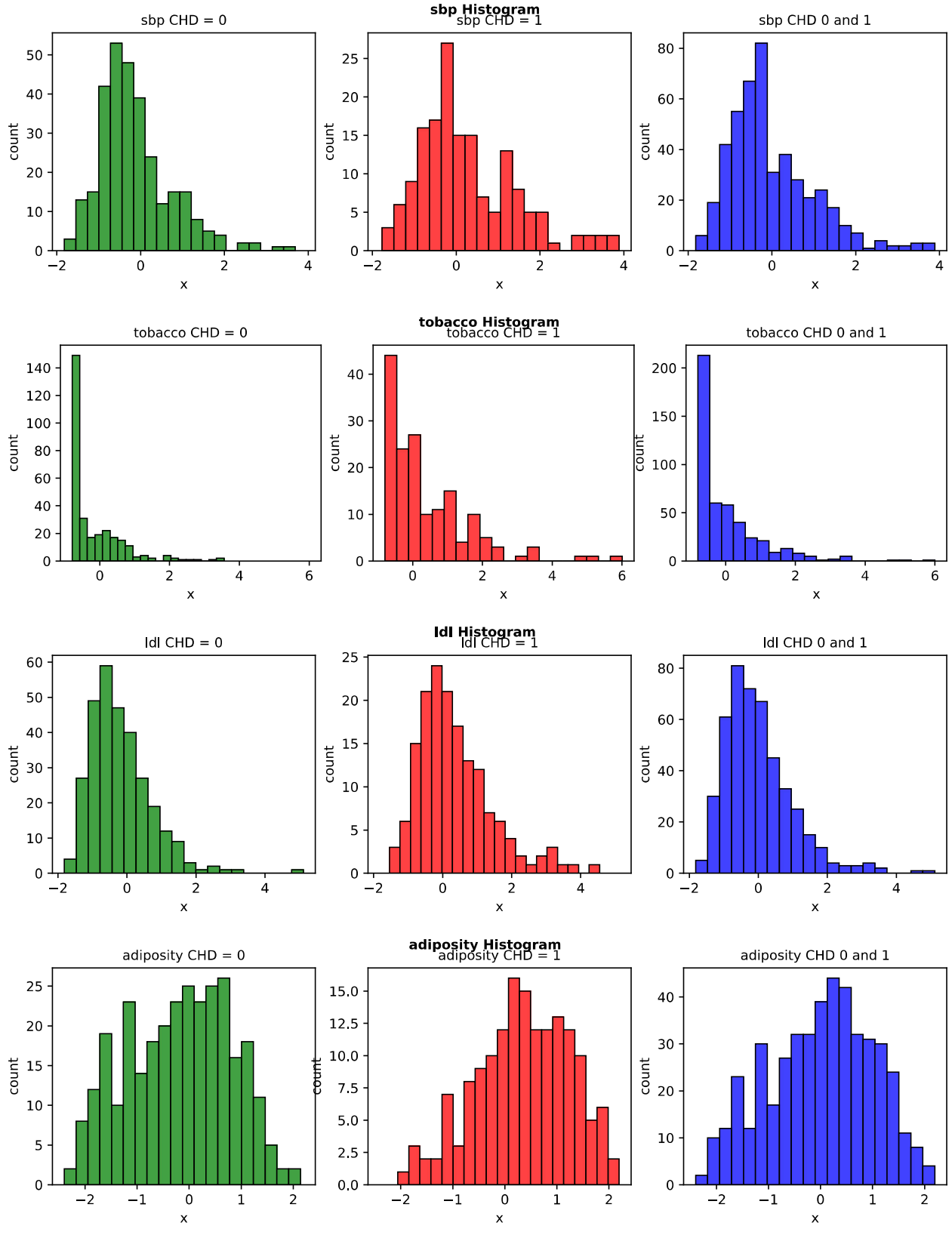
Out[28]:
```
Index(['sbp', 'tobacco', 'ldl', 'adiposity', 'typea', 'obesity', 'alcohol',
       'age', 'chd'],
      dtype='object')
```

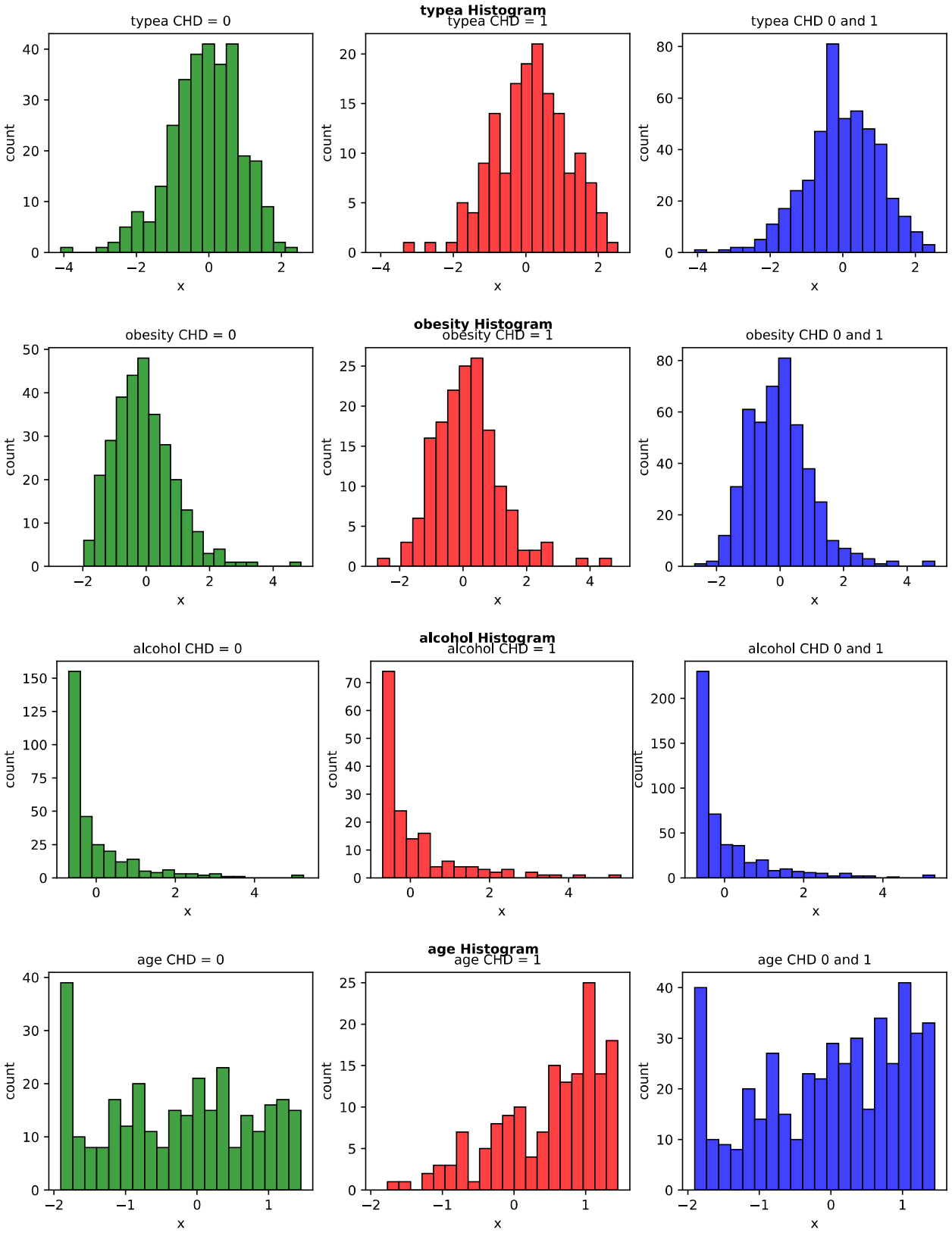
In [29]:
```
df_Scaled_All = df_Scaled_chd0.append(df_Scaled_chd1)
```

I wanted an idea of the distribution spread for each of the variables before I selected distributions to use:

In [30]:
```
for col in df_Scaled_chd1.columns[0:8]:
    fig, axes = plt.subplots(1, 3, sharex=True, figsize=(12,3))
    fig.suptitle('{} Histogram'.format(col), fontsize=10, fontweight='bold')

    sns.histplot(data=df_Scaled_chd0, x=col, bins=20, color='green', ax=axes[0])
    axes[0].set(xlabel='x', ylabel='count')
    axes[0].set_title('{} CHD = 0'.format(col),fontdict= {'fontsize': 10})

    sns.histplot(data=df_Scaled_chd1, x=col, bins=20, color='red', ax=axes[1])
    axes[1].set(xlabel='x', ylabel='count')
    axes[1].set_title('{} CHD = 1'.format(col),fontdict= {'fontsize': 10})

    sns.histplot(data=df_Scaled_All, x=col, bins=20, color='blue', ax=axes[2])
```

```
    axes[2].set(xlabel='x', ylabel='count')
    axes[2].set_title('{} CHD 0 and 1'.format(col),fontdict= {'fontsize': 10})
```





To model the continuous mean for each of the 8 predictor variables I will use a **NEED TO CHOOSE** distribution.

Based on the distribution plots above I feel that a Gaussian with Known Variance distribution is appropriate for several of the variables: sbp, ldl, adiposity, typea, and obesity.

First, I will use the Gaussian_Known_Variance distribution for all the predictors and analyze the means in the CHD = 0 and CHD = 1 subpopulations to perform what I am thinking of as a "Bayesian 2-sample t-test" to decide if the means are significantly different in the subpopulations, and this determine how confident we are that the predictor contributes to CHD.

After using the Gaussian Known Variance distribution for all predictors I will also calculate posterior mean and scale using the **Gaussian Likelihood with Unknown Mean and Unknown Precision** using calculations provided in Module 2.

First, Using Gaussian Known Variance

In [42]:

```
from scipy.stats import binom
from scipy.stats import beta
from scipy.stats import norm
from scipy.stats import gamma
from scipy.stats import t
import math

def posterior_from_conjugate_prior(**kwargs):
    if kwargs['Likelihood_Dist_Type'] == 'Binomial':
        # Get the parameters for the likelihood and prior distribution from the key wor
        x = kwargs['x'] # This is state space of possible values for p = 'probability o
        n = kwargs['n'] # This is the number of Bernoili trials.
        k = kwargs['k'] # This is the number of 'successes'.
        a = kwargs['a'] # This is the parameter alpha for the prior Beta distribution
        b = kwargs['b'] # This is the parameter beta for the prior Beta distribution

        print(f'a_prime = {k + a}.')
        print(f'b_prime = {n - k + b}.')
        Likelihood = binom.pmf(p=x, n=n, k=k)
        Prior = beta.pdf(x=x, a=a, b=b)
        Posterior = beta.pdf(x=x, a=k+a, b=n-k+b)

        return [Prior, Likelihood, Posterior]
    elif kwargs['Likelihood_Dist_Type'] == 'Gaussian_Known_Variance':
        # Get the parameters for the likelihood and prior distribution from the key wor
        x = kwargs['x'] # This is state space of possible values for x in (-infinity,in
        mu = kwargs['mu'] # This is the mean from the data
        var = kwargs['var'] # This is the variance from the data
        prior_mu = kwargs['prior_mu'] # This is the mean for the prior on mu
        prior_var = kwargs['prior_var'] # This is the variance for the prior on mu
        n = kwargs['n']
        #print(kwargs)

        # To answer the challenge question, modify this section with the correct formul
        xsum = n * mu
        #print(f'x sum = {xsum}.') # n * mu

        var_prime = 1 / ((1/prior_var) + (n/var))

        mu_part1 = var_prime
        mu_part2a = prior_mu / prior_var
        mu_part2b = xsum / var
        mu_part2 = (mu_part2a + mu_part2b)
        mu_prime = mu_part1 * mu_part2

        #print(f'mu_prime = {mu_prime}; var_prime = {var_prime}.')
```

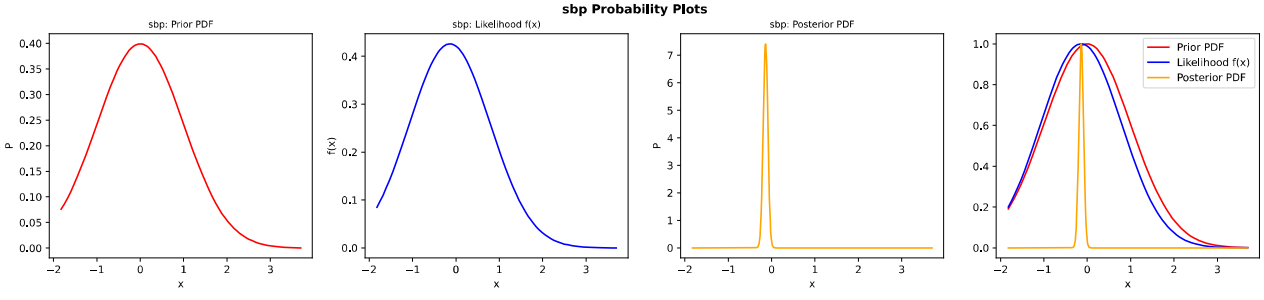
```
        #print(f'var_prime = {var_prime}.')
        Likelihood = norm.pdf(x=x, loc=mu, scale=math.sqrt(var))
        Prior = norm.pdf(x=x, loc=prior_mu, scale=math.sqrt(prior_var))

        Posterior = norm.pdf(x=x, loc=mu_prime, scale=math.sqrt(var_prime))
        #print(Posterior)

        return [Prior, Likelihood, Posterior]
    elif kwargs['Likelihood_Dist_Type'] == 'Gaussian_UnknownMean_UnknownPrec':
        x = kwargs['x'] # This is state space of possible values for x in (-infinity,in
        tau_prior = kwargs['tau_prior']
        mu_prior = kwargs['mu_prior']
        alpha_prior = kwargs['alpha_prior']
        beta_prior = kwargs['beta_prior']
        n = kwargs['n']

        mu_prime = ((tau_prior * mu_prior) + (n * np.mean(x))) / (tau_prior + n)
        alpha_prime = alpha_prior + n

        beta_prime_part2 = np.sum((x - np.mean(x))**2)
        beta_prime_part3 = (tau_prior * n * (np.mean(x) - mu_prior)**2) / (tau_prior +
        beta_prime = beta_prior + beta_prime_part2 + beta_prime_part3

        tau_prime = tau_prior + n

        scale_prime = np.sqrt(beta_prime / (alpha_prime * tau_prime))


        Likelihood = 1
        Prior = 1
        Posterior = t.pdf(x=x, df=alpha_prime, loc=mu_prime, scale=scale_prime)

    elif kwargs['Likelihood_Dist_Type'] == 'Normal_KnownMean_ConjPrior_Gamma':
        # Get the parameters for the likelihood and prior distribution from the key wor
        x = kwargs['x'] # This is state space of possible values for x in (-infinity,in
        alpha = kwargs['a'] # alpha for data
        mu = kwargs['mu'] # mu from data
        var = kwargs['var'] # var from data
        n = kwargs['n']
        #print(kwargs)

        alpha_prime = alpha + (n/2)

        print(f'alpha_prime = {alpha_prime}.')
        # gamma.pdf(x, a, loc=0, scale=1)
        Likelihood = norm.pdf(x=x, loc=mu, scale=math.sqrt(var))
        Prior = gamma.pdf(x=x, a=alpha) # assuming N(0,1)????
        Posterior = (Likelihood * Prior) / np.sum((Likelihood * Prior) * np.abs(x[1] -

        return [Prior, Likelihood, Posterior]
    elif kwargs['Likelihood_Dist_Type'] == 'Gamma_Known_Shape':
        # Get the parameters for the likelihood and prior distribution from the key wor
        x = kwargs['x'] # This is state space of possible values for x in (-infinity,in
        alpha = kwargs['a'] # alpha for data
        prior_alpha = kwargs['prior_a'] # alpha for prior
        n = kwargs['n']
        #print(kwargs)

        alpha_prime = prior_alpha + (n * alpha)

        print(f'alpha_prime = {alpha_prime}.')
```

```
        # gamma.pdf(x, a, loc=0, scale=1)
        Likelihood = gamma.pdf(x=x, a=alpha) # assuming N(0,1)
        Prior = gamma.pdf(x=x, a=prior_alpha) # assuming N(0,1)
        Posterior = gamma.pdf(x=x, a=alpha_prime) # assuming N(0,1)
        #print(Posterior)

        return [Prior, Likelihood, Posterior]

    else:
        print('Distribution type not supported.')
        return -1, -1, -1
```

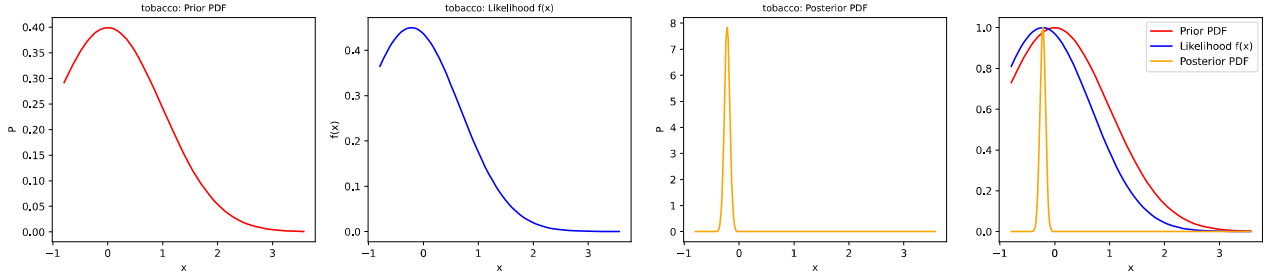
In [43]:

```
# first work with the CHD == 0 set
n = df_Scaled_chd0.shape[0]
print("CHD == 0 Predictor Variable Mean Distribution Plots: Gaussian Known Variance")
for col, mean, min, max, stdv in zip(lst_cols, lst_means_chd0, lst_mins_chd0, lst_maxs_
    x = np.arange(min - 0.0001, max + 0.0001, .001)
    print("-"*30)
    Prior, Likelihood, Posterior = posterior_from_conjugate_prior(Likelihood_Dist_Type=

    print("Confidence Interval for CHD = 0 {}: Gaussian Known Variance".format(col))
    printConfidenceInterval(post = Posterior, dx = abs(x[1] - x[0]), lb=0.025, ub=0.975
    printProbPlots(x, Prior, Likelihood, Posterior, col)
```

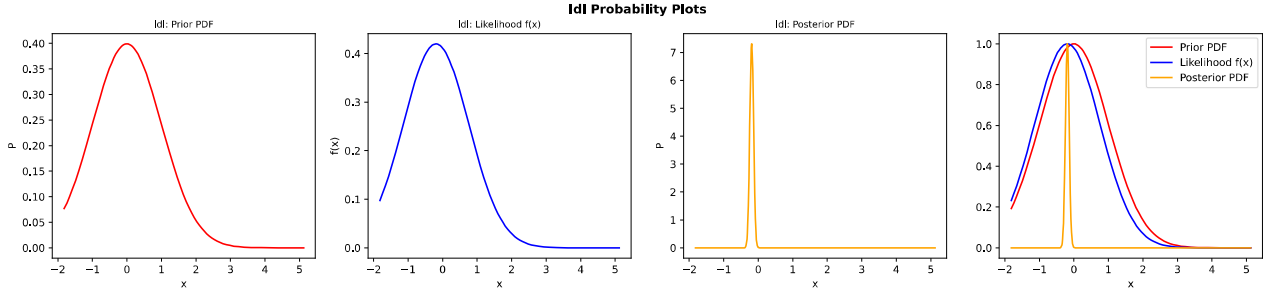
```
CHD == 0 Predictor Variable Mean Distribution Plots: Gaussian Known Variance
------------------------------
Confidence Interval for CHD = 0 sbp: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [-0.24,-0.03].
------------------------------
Confidence Interval for CHD = 0 tobacco: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [-0.32,-0.12].
------------------------------
Confidence Interval for CHD = 0 ldl: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [-0.30,-0.08].
------------------------------
Confidence Interval for CHD = 0 adiposity: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [-0.30,-0.07].
------------------------------
Confidence Interval for CHD = 0 typea: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [-0.19,0.04].
------------------------------
Confidence Interval for CHD = 0 obesity: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [-0.18,0.04].
------------------------------
Confidence Interval for CHD = 0 alcohol: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [-0.16,0.07].
------------------------------
Confidence Interval for CHD = 0 age: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [-0.38,-0.16].
```



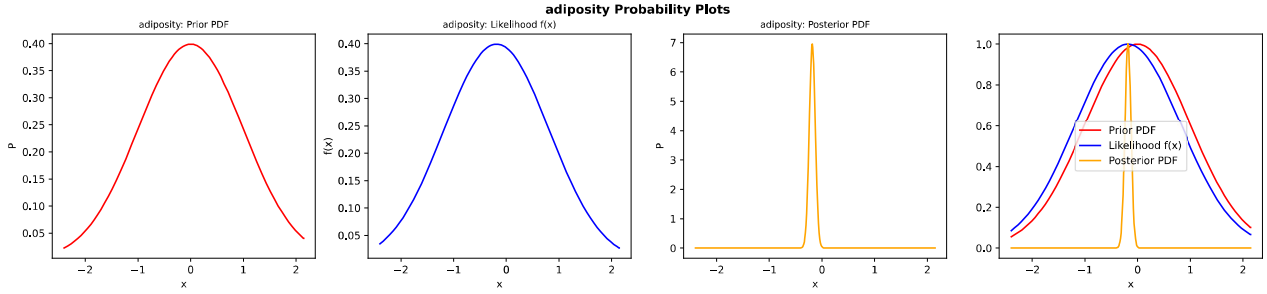
tobacco Probability Plots



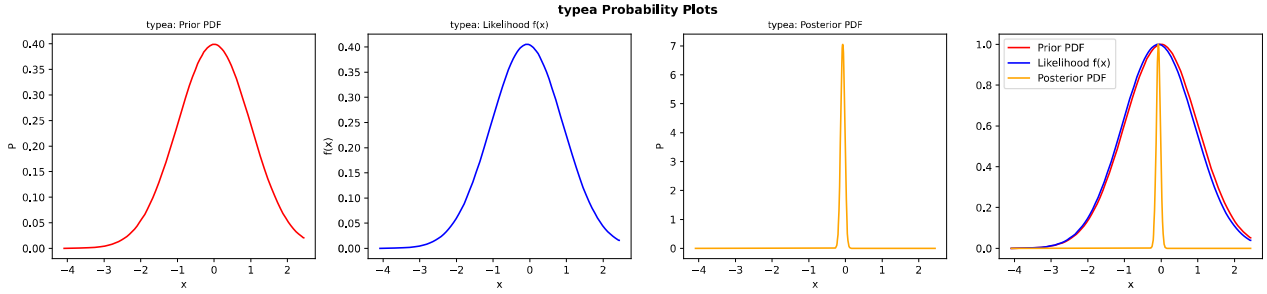
ldl Probability Plots



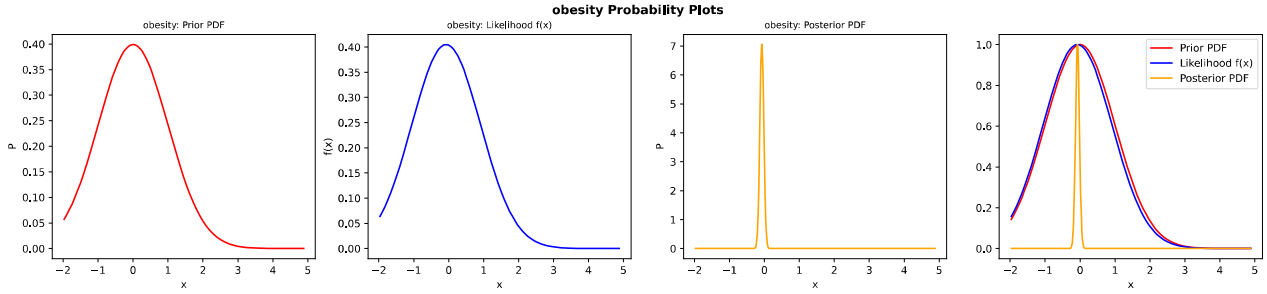
adiposity Probability Plots



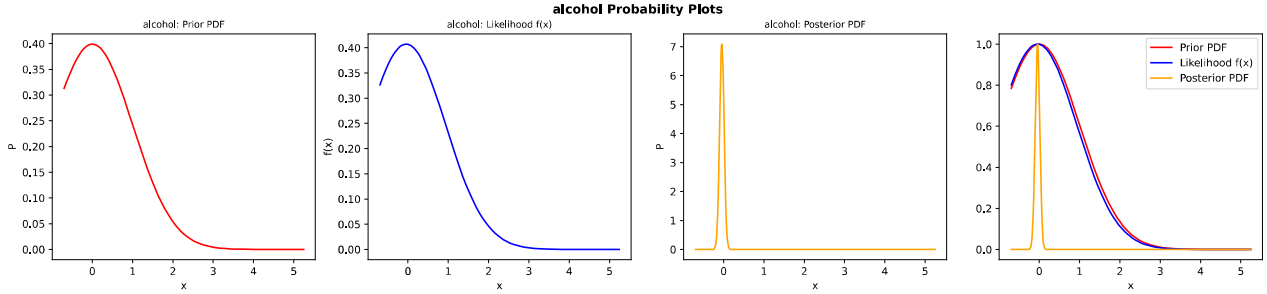
typea Probability Plots

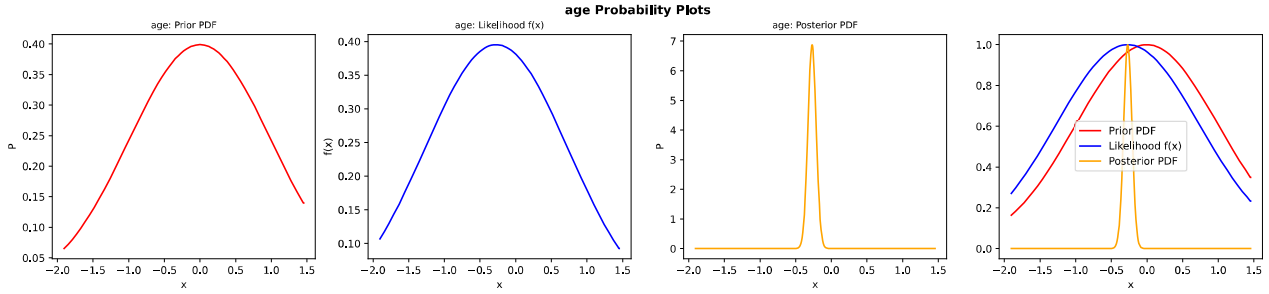


obesity Probability Plots



alcohol Probability Plots

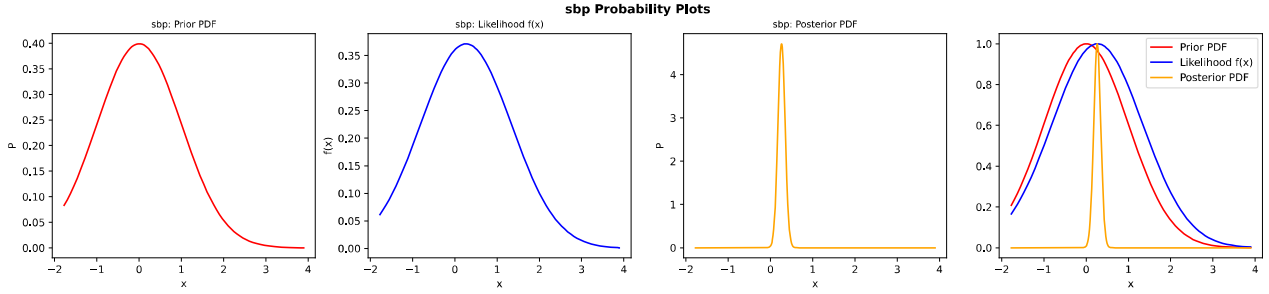




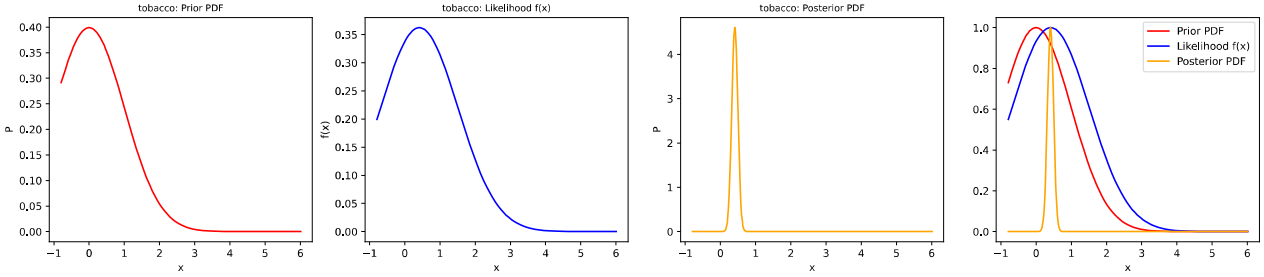
In [44]:

```
# CHD == 1 set
n = df_Scaled_chd1.shape[0]
print("CHD == 1 Predictor Variable Mean Distribution Plots: Gaussian Known Variance")
for col, mean, min, max, stdv in zip(lst_cols, lst_means_chd1, lst_mins_chd1, lst_maxs_
    x = np.arange(min - 0.0001, max + 0.0001, .001)
    print("-"*30)
    Prior, Likelihood, Posterior = posterior_from_conjugate_prior(Likelihood_Dist_Type=
    print("Confidence Interval for CHD = 1 {}: Gaussian Known Variance".format(col))
    printConfidenceInterval(post = Posterior, dx = abs(x[1] - x[0]), lb=0.025, ub=0.975
    printProbPlots(x, Prior, Likelihood, Posterior, col)
```

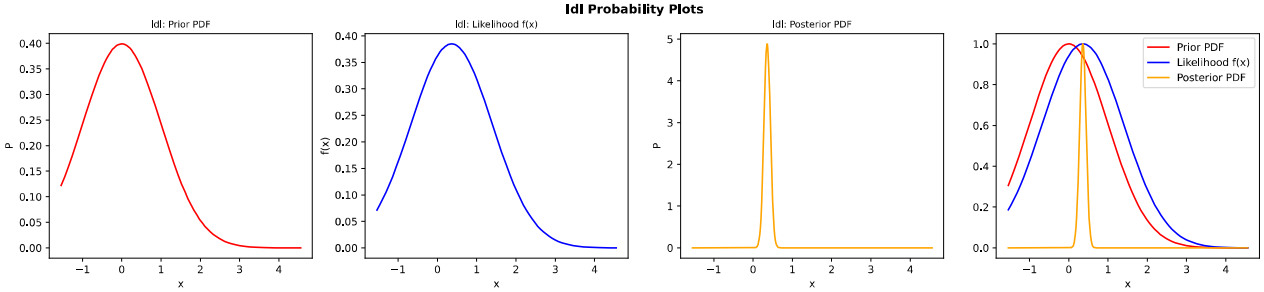
```
CHD == 1 Predictor Variable Mean Distribution Plots: Gaussian Known Variance
------------------------------
Confidence Interval for CHD = 1 sbp: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [0.10,0.43].
------------------------------
Confidence Interval for CHD = 1 tobacco: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [0.24,0.58].
------------------------------
Confidence Interval for CHD = 1 ldl: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [0.20,0.52].
------------------------------
Confidence Interval for CHD = 1 adiposity: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [0.20,0.49].
------------------------------
Confidence Interval for CHD = 1 typea: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [-0.02,0.30].
------------------------------
Confidence Interval for CHD = 1 obesity: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [-0.02,0.29].
------------------------------
Confidence Interval for CHD = 1 alcohol: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [-0.07,0.25].
------------------------------
Confidence Interval for CHD = 1 age: Gaussian Known Variance
The probability is 0.95 that the value for p in the interval [0.38,0.64].
```



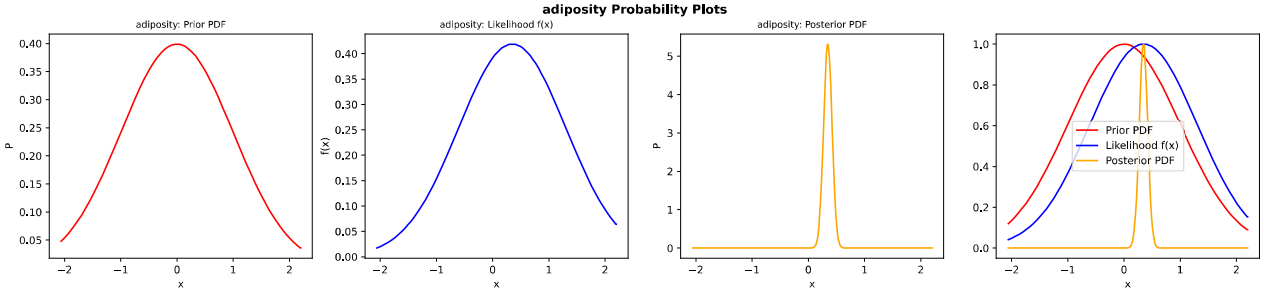
tobacco Probability Plots



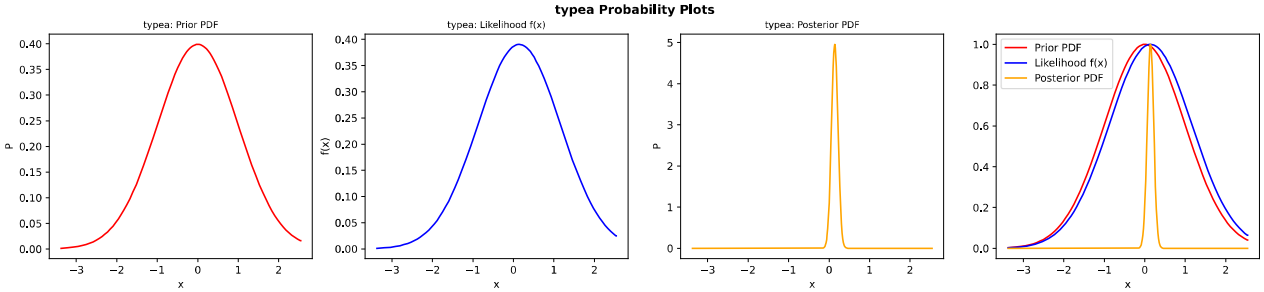
ldl Probability Plots



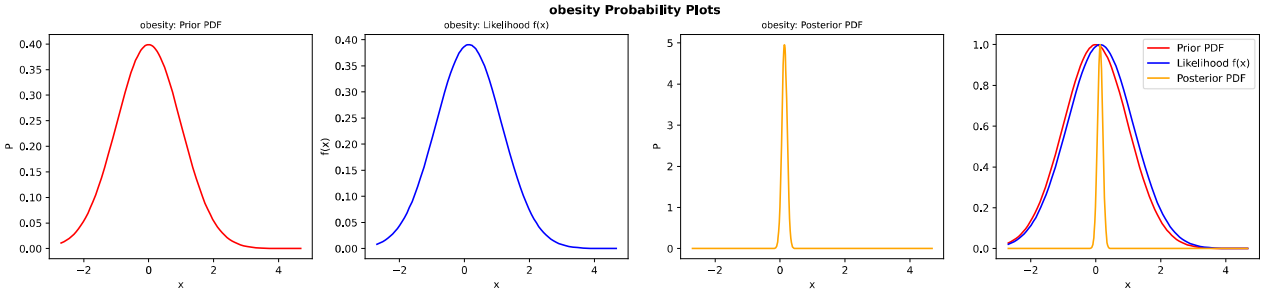
adiposity Probability Plots



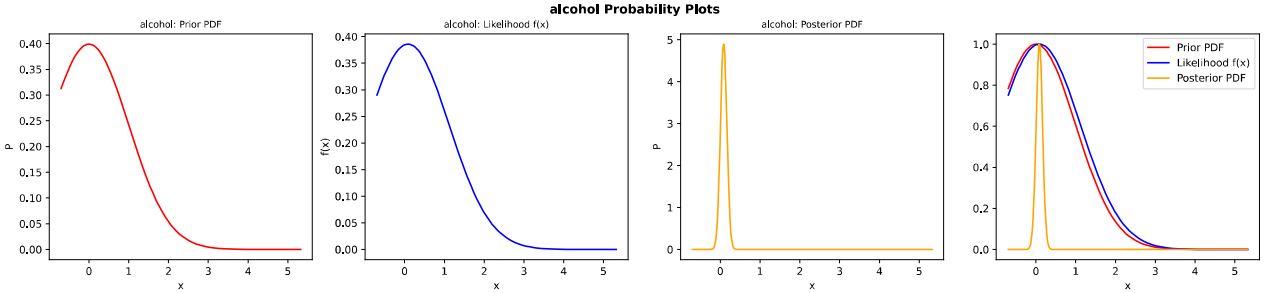
typea Probability Plots

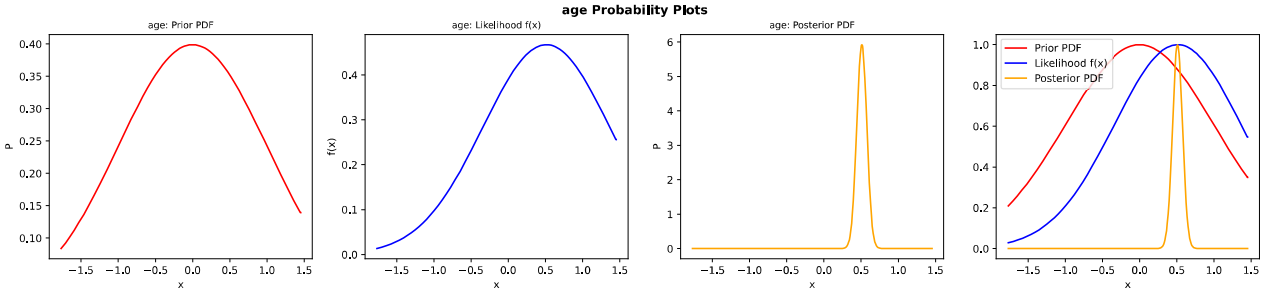


obesity Probability Plots



alcohol Probability Plots





Gaussian Known Varianace Analysis of Means for Predictors in CHD = 0 & CHD = 1

CHD predictor	CHD = 0 posterior mu CI (and width)	CHD = 1 posterior mu CI (and width)	Contains 0 Y/N
sbp	-0.24 - -0.03 (0.21)	0.10 - 0.42 (0.32)	CHD=0: N, CHD=1: N
tobacco	-0.32 - -0.10 (.22)	0.24 - 0.58 (.34)	CHD=0: N, CHD=1: N
ldl	-0.30 - -0.08 (.22)	0.20 - 0.52 (.32)	CHD=0: N, CHD=1: N
adiposity	-0.30 - -0.07 (.23)	0.20 - 0.49 (.29)	CHD=0: N, CHD=1: N
typea	-0.19 - 0.04 (.23)	-0.02 - 0.30 (.32)	**CHD=0: Y, CHD=1: Y**
obesity	-0.18 - 0.04 (.22)	-0.02 - 0.29 (.31)	**CHD=0: Y, CHD=1: Y**
alcohol	-0.16 - 0.07 (.23)	-0.07 - 0.25 (.32)	**CHD=0: Y, CHD=1: Y**
age	-0.38 - -0.16 (.22)	0.38 - 0.64 (.26)	CHD=0: N, CHD=1: N

The table above demonstrates that we are more certain about the CHD=0 means (smaller variance), which makes more sense because we have more observations in the CHD=0 data set (302 vs 160).

The typea, obesity and alcohol 95% CI for the means span 0, indicating that the differences in the means may be due to randomness in the data and are not significant; however, with so few observations in either data set, and because most of the range is isolated on the negative (for CHD = 0)amd positive (for CHD = 1) I would want to see more data to definitively say that typea, obesity and alcohol do not have different mean values in the CHD = 0 and CHD = 1 data sets.

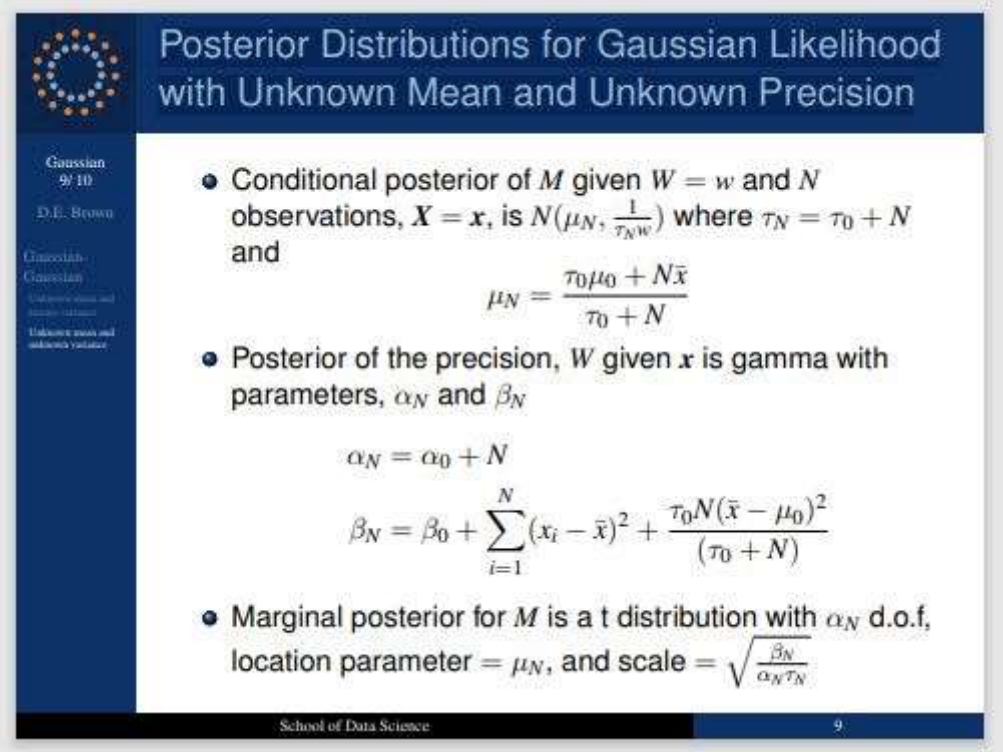
For sbp, tabacco, ldl, adiposity, and age the confidence intervals are either entirely negative (CHD = 0) or positive (CHD = 1) indicating that these mean differences are not do to randomness in sampling.

Now, with Gaussian Unknown Mean and Unknown Precision

I would now like to attempt to produce posterior means and scale using Likelihood Gaussian with Unknown Mean and Unknown Precision.

I am trying this with just the calculations instead of the plotting function used above just to try doing this in a different way.

I am using this slide for setting my parameters:



To calculate the **posterior mean** I need to identify/calculate:

1. tau_prior = weight to decrease the weight of the prior mean compared to the actual mean
    - number between 0 and N.
    - I will use **2** because I have very few samples compared to a population
2. mu_prior
    - **0** per the question definition
3. N = number of observations
    - CHD=0: **302**
    - CHD=1: **160**
4. mu = from the data for each predictor

To calculate the **posterior precision** I will need to identify/calculate:

1. alpha_prior & beta prior
    - I am going to set these to values for a gamma distribution to try to create as close to an uninformative gamma distribution as I have been able to do:
        - alpha_prior: 0.001
        - beta_prior: 0.001
2. N = number of observations
3. observation values from the CHD0 and CHD1 dataframes
4. means for each predictor from the dataframe
5. tau_prior

- **2** see above

6. mu_prior:
   - **0** per the question definition

After I have both the posterior mean, and posterior scale I can compare these with the Gaussian Known Variance analysis. I am very curious about the results and comparison.

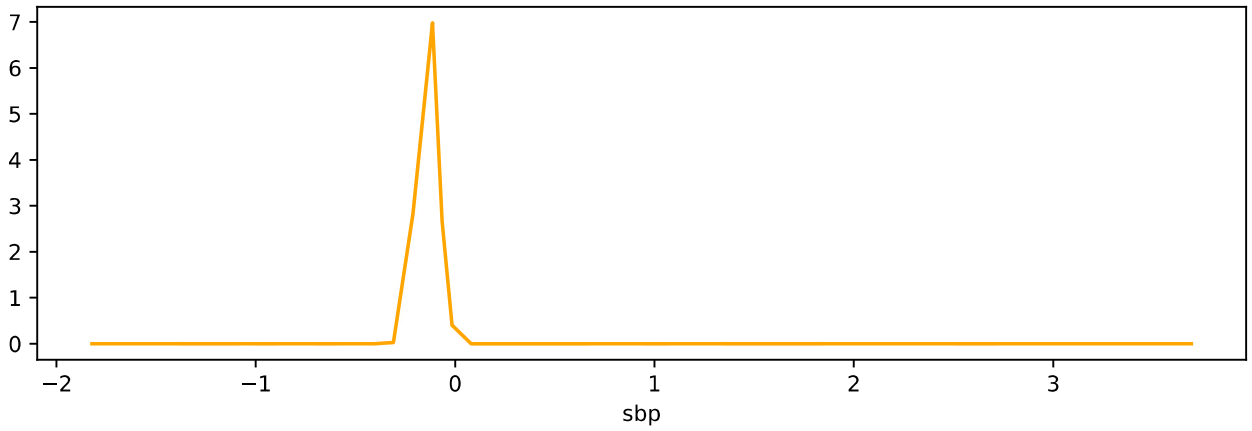
In [34]:

```
# for the extra credit save the posterior means and stdvs
lst_chd0_post_mean = []
lst_chd0_post_stdv = []

n = df_Scaled_chd0.shape[0]
print("CHD == 0: Gaussian Likelihood with Unknown Mean and Unknown Precision")
for col, mu in zip(lst_cols, lst_means_chd0):
    print("-"*30)
    # get the column values
    x = df_Scaled_chd0[col]
    tau_prior = 2
    mu_prior = 0
    alpha_prior = 0.001
    beta_prior = 0.001

    mu_prime = ((tau_prior * mu_prior) + (n * np.mean(x))) / (tau_prior + n)
    alpha_prime = alpha_prior + n

    beta_prime_part2 = np.sum((x - np.mean(x))**2)
    beta_prime_part3 = (tau_prior * n * (np.mean(x) - mu_prior)**2) / (tau_prior + n)
    beta_prime = beta_prior + beta_prime_part2 + beta_prime_part3

    tau_prime = tau_prior + n

    scale_prime = np.sqrt(beta_prime / (alpha_prime * tau_prime))

    # store post mean and stdv for extra credit
    lst_chd0_post_mean.append(mu_prime)
    lst_chd0_post_stdv.append(scale_prime)

    Posterior = t.pdf(x=x, df=alpha_prime, loc=mu_prime, scale=scale_prime)
    print("Confidence Interval for CHD = 1 {}: posterior loc: {:0.4f}, posterior scale:
    plus_minus = 1.96 * (scale_prime / np.sqrt(n))
    print('The probability is 0.95 that the value for p in the interval [{:0.3f},{:0.3f

    printPosteriorPlots(x, Posterior, 'Unknown Mean Unknown Precision: CHD=0: ' + col)
```

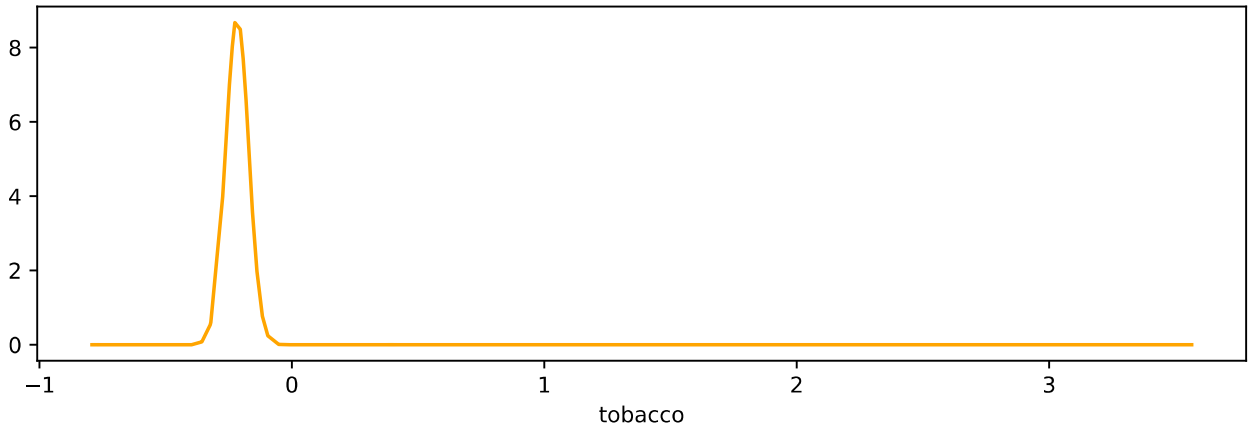
```
CHD == 0: Gaussian Likelihood with Unknown Mean and Unknown Precision
------------------------------
Confidence Interval for CHD = 1 sbp: posterior loc: -0.1389, posterior scale: 0.0502
The probability is 0.95 that the value for p in the interval [-0.145,-0.133].
------------------------------
Confidence Interval for CHD = 1 tobacco: posterior loc: -0.2165, posterior scale: 0.0450
The probability is 0.95 that the value for p in the interval [-0.222,-0.211].
------------------------------
Confidence Interval for CHD = 1 ldl: posterior loc: -0.1900, posterior scale: 0.0517
The probability is 0.95 that the value for p in the interval [-0.196,-0.184].
------------------------------
Confidence Interval for CHD = 1 adiposity: posterior loc: -0.1836, posterior scale: 0.05
72
The probability is 0.95 that the value for p in the interval [-0.190,-0.177].
```

```
------------------------------
Confidence Interval for CHD = 1 typea: posterior loc: -0.0745, posterior scale: 0.0555
The probability is 0.95 that the value for p in the interval [-0.081,-0.068].
------------------------------
Confidence Interval for CHD = 1 obesity: posterior loc: -0.0723, posterior scale: 0.0556
The probability is 0.95 that the value for p in the interval [-0.079,-0.066].
------------------------------
Confidence Interval for CHD = 1 alcohol: posterior loc: -0.0452, posterior scale: 0.0550
The probability is 0.95 that the value for p in the interval [-0.051,-0.039].
------------------------------
Confidence Interval for CHD = 1 age: posterior loc: -0.2694, posterior scale: 0.0583
The probability is 0.95 that the value for p in the interval [-0.276,-0.263].
```

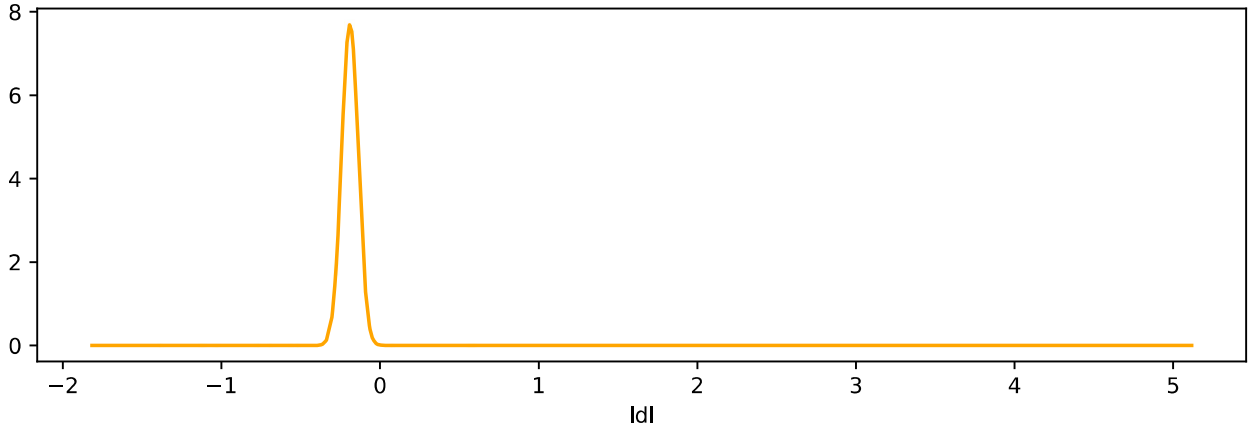
Unknown Mean Unknown Precision: CHD=0: sbp Posterior Plot



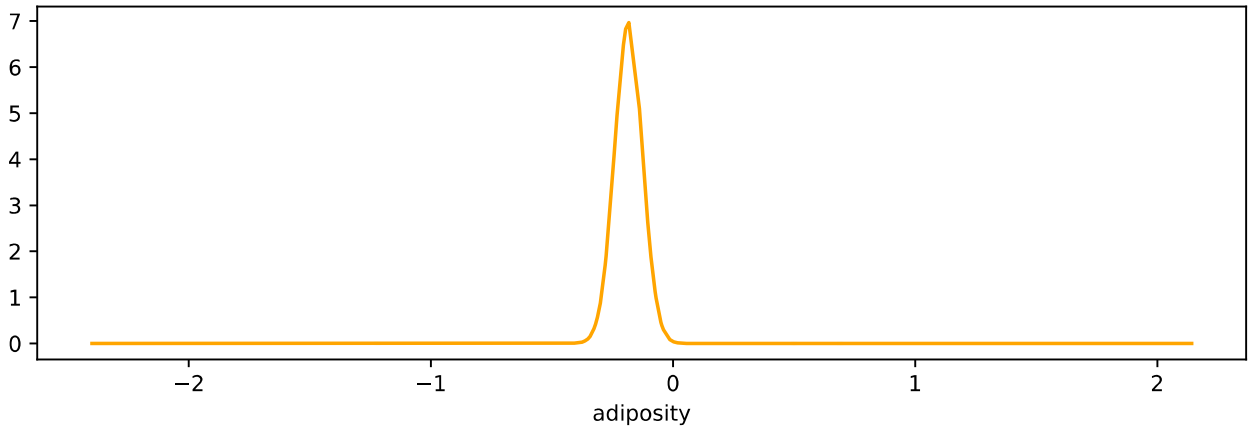
Unknown Mean Unknown Precision: CHD=0: tobacco Posterior Plot



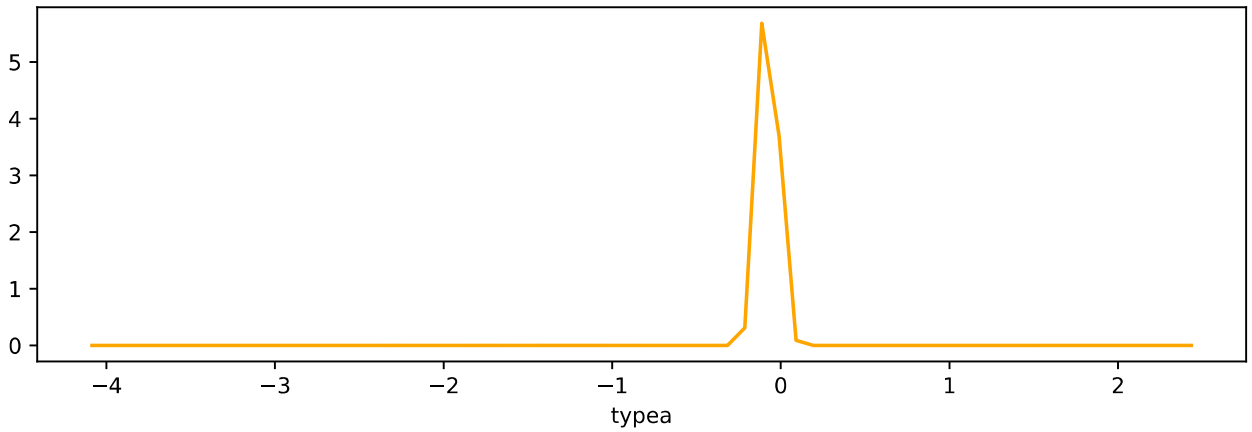
Unknown Mean Unknown Precision: CHD=0: ldl Posterior Plot



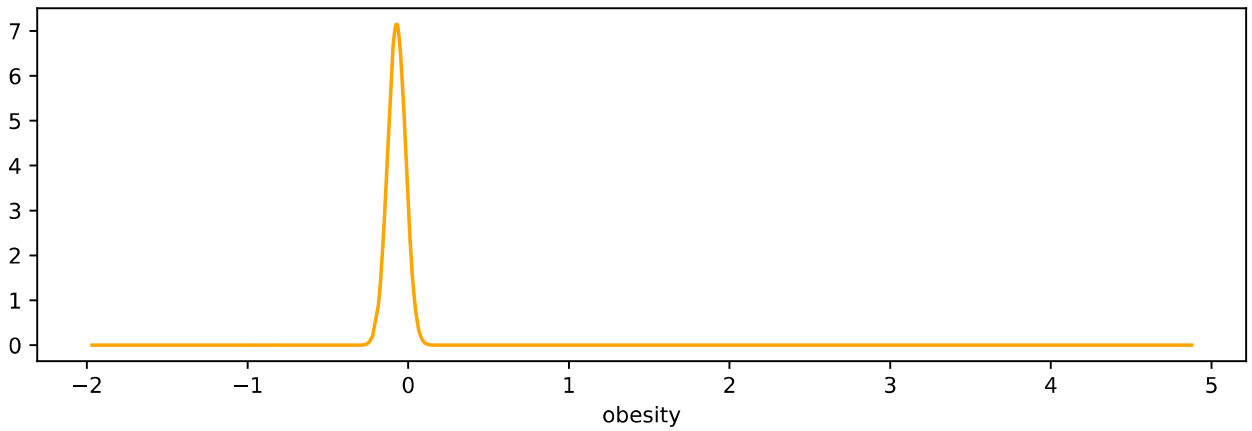
Unknown Mean Unknown Precision: CHD=0: adiposity Posterior Plot

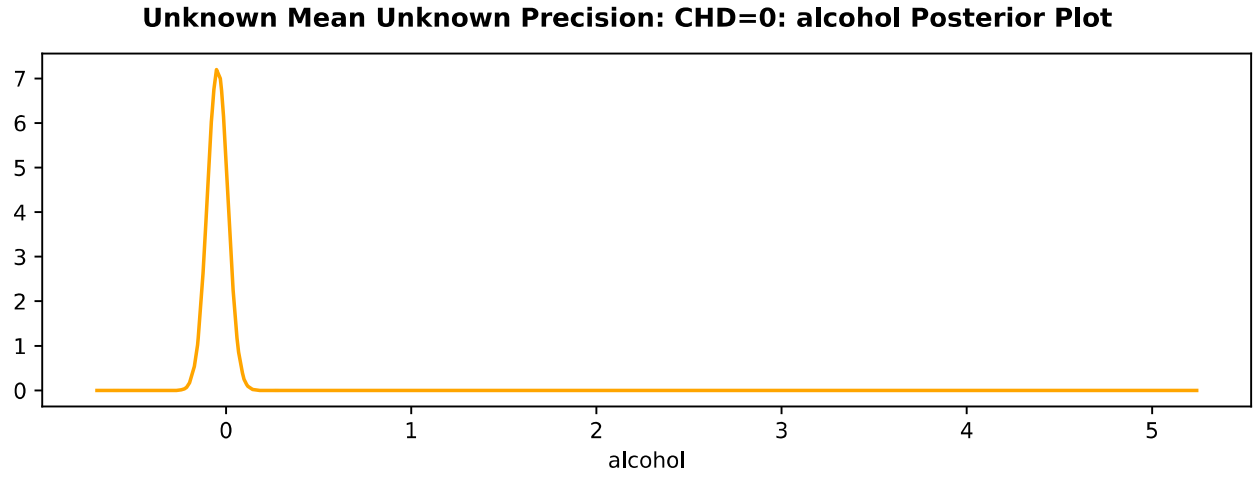


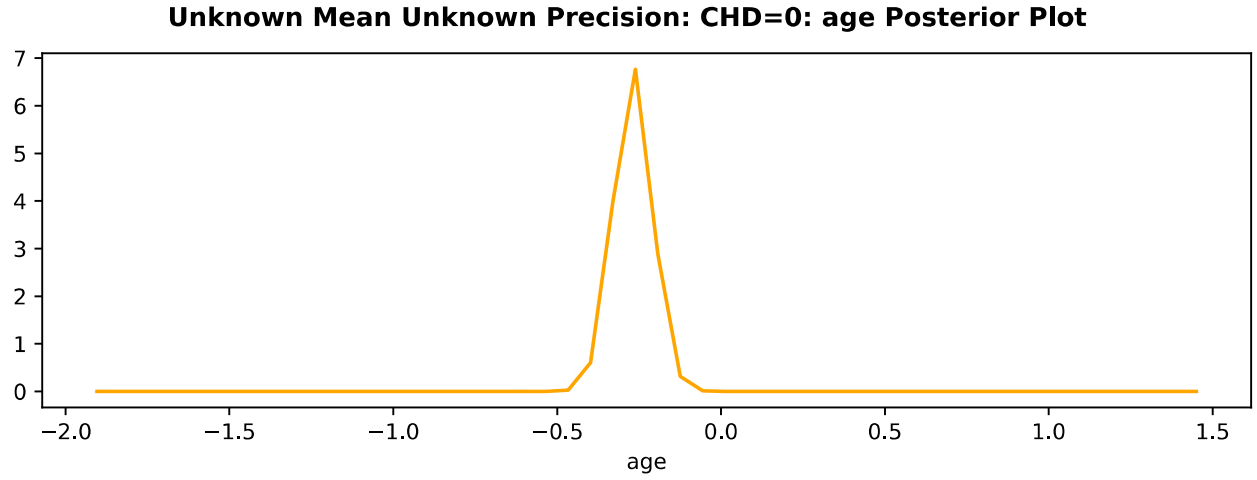
Unknown Mean Unknown Precision: CHD=0: typea Posterior Plot



Unknown Mean Unknown Precision: CHD=0: obesity Posterior Plot







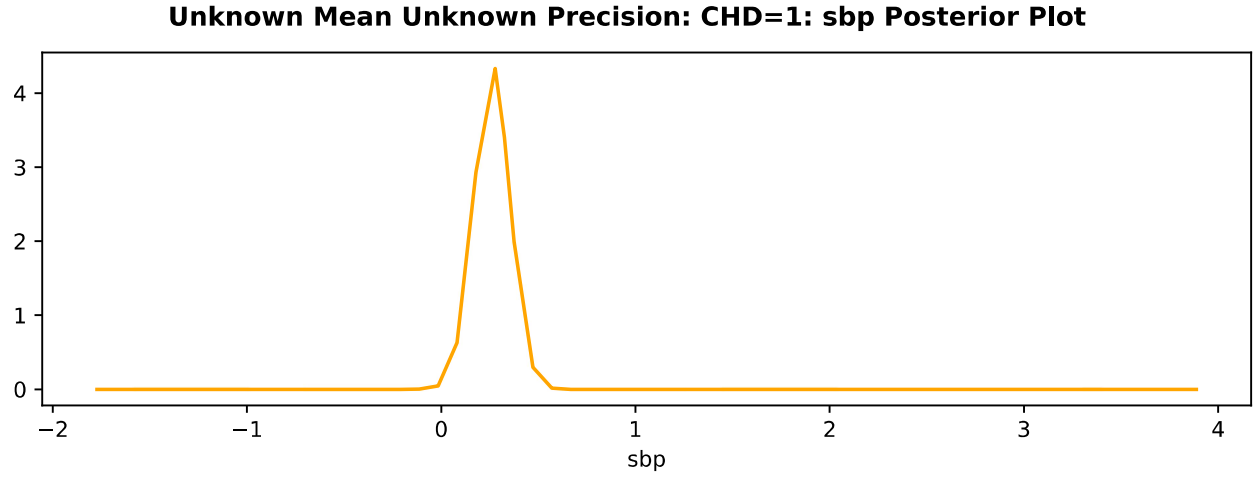
Now for CHD = 1 with Gaussian Unknown Mean Unknown Precision

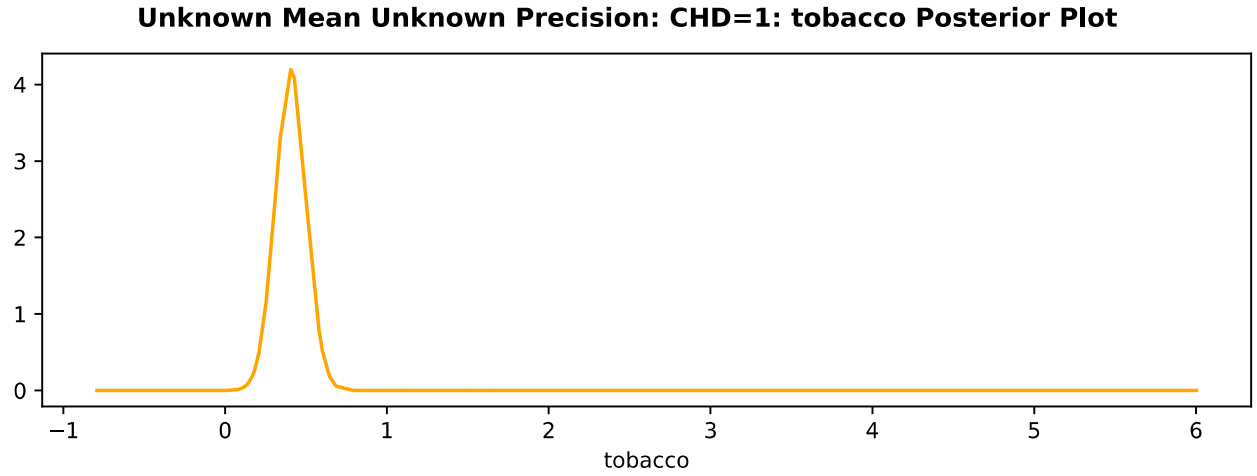
In [35]:

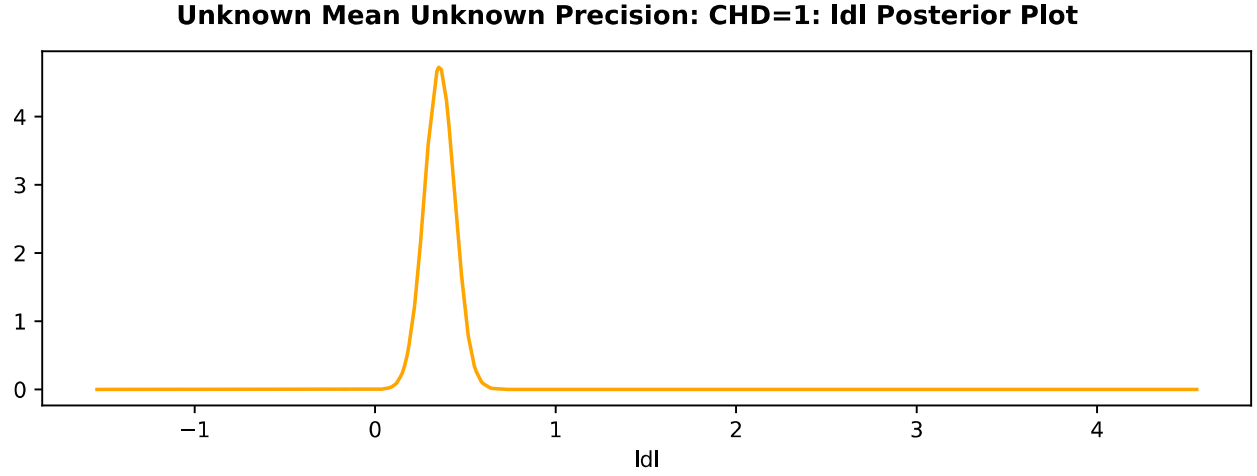
```
lst_chd1_post_mean = []
lst_chd1_post_stdv = []

n = df_Scaled_chd1.shape[0]
print("CHD == 1: Gaussian Likelihood with Unknown Mean and Unknown Precision")
for col, mu in zip(lst_cols, lst_means_chd1):
    print("-"*30)
    # get the column values
    x = df_Scaled_chd1[col]
    tau_prior = 2
    mu_prior = 0
    alpha_prior = 0.001
    beta_prior = 0.001

    mu_prime = ((tau_prior * mu_prior) + (n * np.mean(x))) / (tau_prior + n)
    alpha_prime = alpha_prior + n

    beta_prime_part2 = np.sum((x - np.mean(x))**2)
    beta_prime_part3 = (tau_prior * n * (np.mean(x) - mu_prior)**2) / (tau_prior + n)
    beta_prime = beta_prior + beta_prime_part2 + beta_prime_part3

    tau_prime = tau_prior + n

    scale_prime = np.sqrt(beta_prime / (alpha_prime * tau_prime))

    # store post mean and stdv for extra credit
```

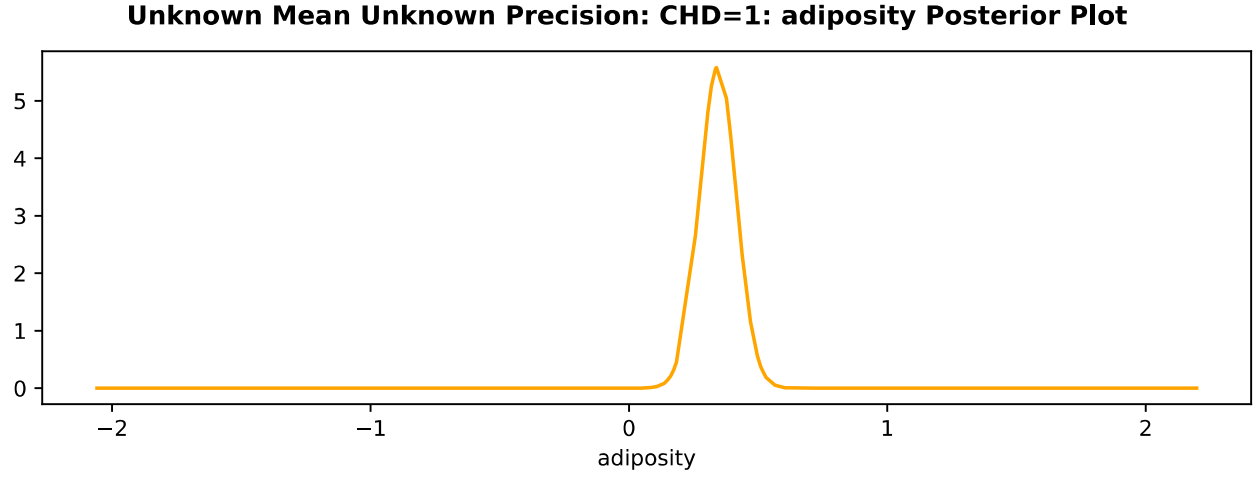
```
    lst_chd1_post_mean.append(mu_prime)
    lst_chd1_post_stdv.append(scale_prime)

    Posterior = t.pdf(x=x, df=alpha_prime, loc=mu_prime, scale=scale_prime)
    print("Confidence Interval for CHD = 1 {}: posterior loc: {:0.4f}, posterior scale:
    plus_minus = 1.96 * (scale_prime / np.sqrt(n))
    print('The probability is 0.95 that the value for p in the interval [{:0.3f},{:0.3f

    printPosteriorPlots(x, Posterior, 'Unknown Mean Unknown Precision: CHD=1: ' + col)
```

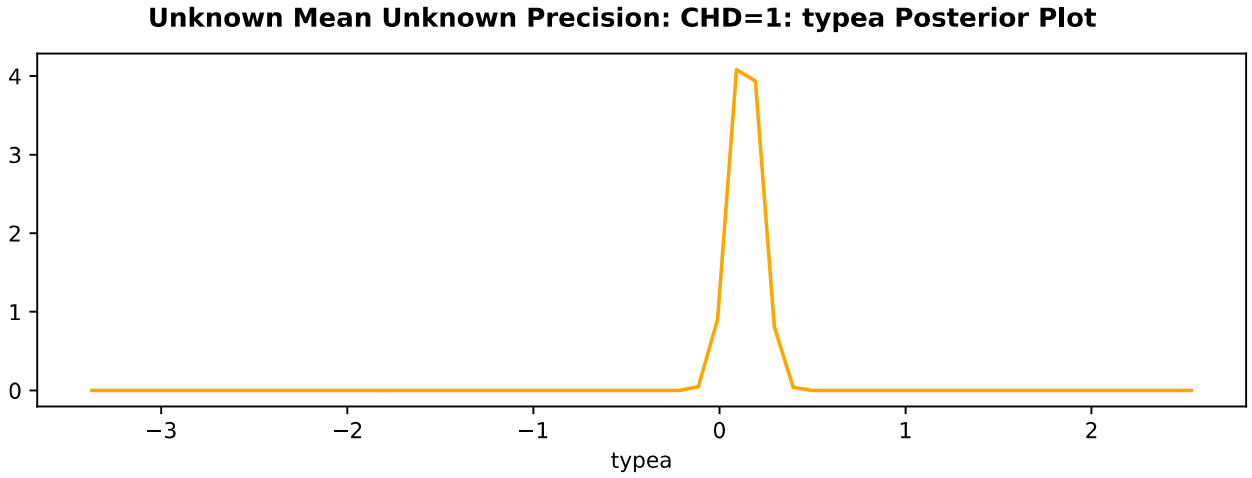
```
CHD == 1: Gaussian Likelihood with Unknown Mean and Unknown Precision
------------------------------
Confidence Interval for CHD = 1 sbp: posterior loc: 0.2607, posterior scale: 0.0905
The probability is 0.95 that the value for p in the interval [0.247,0.275].
------------------------------
Confidence Interval for CHD = 1 tobacco: posterior loc: 0.4062, posterior scale: 0.0950
The probability is 0.95 that the value for p in the interval [0.392,0.421].
------------------------------
Confidence Interval for CHD = 1 ldl: posterior loc: 0.3566, posterior scale: 0.0842
The probability is 0.95 that the value for p in the interval [0.344,0.370].
------------------------------
Confidence Interval for CHD = 1 adiposity: posterior loc: 0.3444, posterior scale: 0.071
1
The probability is 0.95 that the value for p in the interval [0.333,0.355].
------------------------------
Confidence Interval for CHD = 1 typea: posterior loc: 0.1398, posterior scale: 0.0818
The probability is 0.95 that the value for p in the interval [0.127,0.152].
------------------------------
Confidence Interval for CHD = 1 obesity: posterior loc: 0.1357, posterior scale: 0.0816
The probability is 0.95 that the value for p in the interval [0.123,0.148].
------------------------------
Confidence Interval for CHD = 1 alcohol: posterior loc: 0.0848, posterior scale: 0.0838
The probability is 0.95 that the value for p in the interval [0.072,0.098].
------------------------------
Confidence Interval for CHD = 1 age: posterior loc: 0.5055, posterior scale: 0.0573
The probability is 0.95 that the value for p in the interval [0.497,0.514].
```

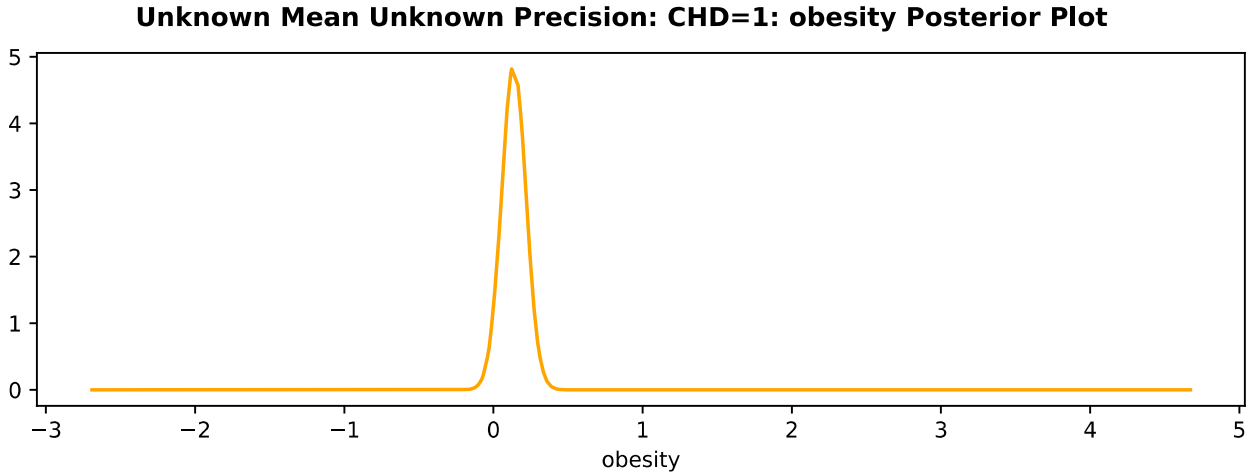


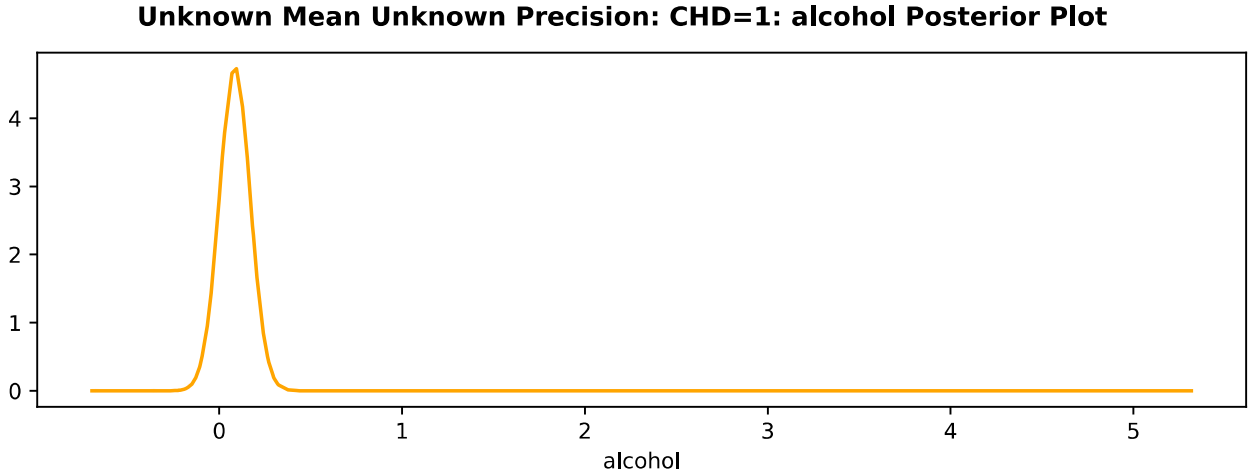

Unknown Mean Unknown Precision: CHD=1: tobacco Posterior Plot
4
3
2
1
0
−1
0
1
2
3
4
5
6
tobacco

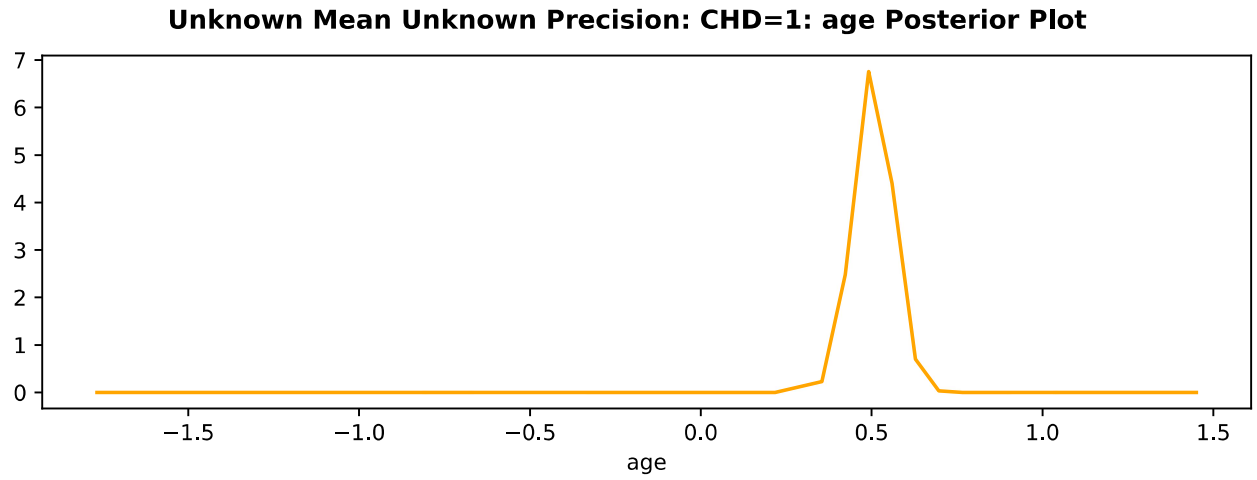

Unknown Mean Unknown Precision: CHD=1: ldl Posterior Plot
4
3
2
1
0
−1
0
1
2
3
4
ldl


Unknown Mean Unknown Precision: CHD=1: adiposity Posterior Plot
5
4
3
2
1
0
−2
−1
0
1
2
adiposity


Unknown Mean Unknown Precision: CHD=1: typea Posterior Plot
4
3
2
1
0
−3
−2
−1
0
1
2
typea


Unknown Mean Unknown Precision: CHD=1: obesity Posterior Plot
5
4
3
2
1
0
−3
−2
−1
0
1
2
3
4
5
obesity


Unknown Mean Unknown Precision: CHD=1: alcohol Posterior Plot
4
3
2
1
0
0
1
2
3
4
5
alcohol



The results from Gaussian Unknown Mean Unknown Precision are significantly different from the previous distribution.

I am confident this second analysis with unknown precision is correct (because we have so few samples in our sub populations so I was very unsure of my data), but the differences are interesting for me to see:

Gaussian Unknown Mean Unknown Precision Means for Predictors in CHD = 0 & CHD = 1

CHD predictor	**CHD = 0 posterior mu CI (and width)**	**CHD = 1 posterior mu CI (and width)**	Contains 0 Y/N
sbp	-0.145 - -0.133 (0.012)	0.247 - 0.275(0.028)	CHD=0: N, CHD=1: N
tobacco	-0.222 - -0.211 (0.011)	0.392 - 0.421 (0.029)	CHD=0: N, CHD=1: N
ldl	-0.196 - -0.184 (0.012)	0.344 - 0.37 (0.026)	CHD=0: N, CHD=1: N
adiposity	-0.19 - -0.177 (0.013)	0.333 - 0.355 (.022)	CHD=0: N, CHD=1: N
typea	-0.081 - 0.068 (0.013)	0.127 - 0.152 (0.025)	CHD=0: N, CHD=1: N
obesity	-0.079 - -0.066 (0.013)	0.123 - 0.148 (0.025)	CHD=0: N, CHD=1: N
alcohol	-0.051 - -0.039 (0.012)	0.072 - 0.098 (0.026)	CHD=0: N, CHD=1: N
age	-0.276 - -0.263 (0.013)	0.497 - 0.514 (0.017)	CHD=0: N, CHD=1: N

Analysis of CI's from Gaussian Unknown Mean Unknown Precision:

The ranges for the CIs using the correct distribution are much smaller, and all means in the two sub populations appear significantly different; **although typea, obesity, and alcohol are not far from 0 for either the CHD=0 or CHD=1 cohorts.**

For 5 extra credit points

compute the probability of observing a point at least as extreme as the posterior mean of patients without coronary heart disease under the posterior distribution for the patients with coronary heart disease.

Then compute the probability of observing a point at least as extreme as the posterior mean of patients with coronary heart disease under the posterior distribution for the patients without coronary heart disease.

Answer: Using my univariate analysis with the posterior means and stdv's from the Guassian Unknown Mean Unknown Precision:

In [36]:
```
# check posterior means of CHD = 0 against posterior distribution for CHD = 1
print("Probability of posterior mean for CHD=0 occuring under posterior distribution fo
for col, chd1_post_mean, chd1_post_stdv, chd0_post_mean,  in zip(lst_cols, lst_chd1_pos
    # Try sbp of
    test_prob = norm(loc=chd1_post_mean, scale=chd1_post_stdv).cdf(chd0_post_mean)
    print("For {}: {:1.4f}".format(col, test_prob))
```

```
Probability of posterior mean for CHD=0 occuring under posterior distribution for CHD=1
For sbp: 0.0000
For tobacco: 0.0000
For ldl: 0.0000
For adiposity: 0.0000
For typea: 0.0044
For obesity: 0.0054
For alcohol: 0.0604
For age: 0.0000
```

In [37]:
```
# check posterior means of CHD = 1 against posterior distribution for CHD = 0
print("Probability of posterior mean for CHD=1 occuring under posterior distribution fo
for col, chd0_post_mean, chd0_post_stdv, chd1_post_mean,  in zip(lst_cols, lst_chd0_pos
    # Try sbp of
    test_prob = 1 - norm(loc=chd0_post_mean, scale=chd0_post_stdv).cdf(chd1_post_mean)
    print("For {}: {:1.4f}".format(col, test_prob))
```

```
Probability of posterior mean for CHD=1 occuring under posterior distribution for CHD=0
For sbp: 0.0000
For tobacco: 0.0000
For ldl: 0.0000
For adiposity: 0.0000
For typea: 0.0001
For obesity: 0.0001
For alcohol: 0.0090
For age: 0.0000
```

It seems that **alcohol** could have values as extreme as the CHD=0 means when using the CHD=1 distributions - meaning that the two subpopulations (chd=0 and chd=1) may have similar rates alcohol use, which does agree with the initial historgrams I plotted for alcohol in chd=0, chd=1, and all.

Question 6 (10 points)

See PowerPoint Attached to Assignment

Question 7 (15 points)

Using the Python Notebook https://www.kaggle.com/billbasener/pt2-probabilities-likelihoods-and-bayes-theorem, complete the challenge question from Section 6: Modify the code from Section 5 to and add the ability to use the posterior from conjugate prior function to output the posterior probability parameters given parameters and for a Gaussian Likelihood with known variance σ2, and use your modified function to create the Prior, Likelihood, Posterior plots as in Section 5 of the notebook.

In [38]:

```
from scipy.stats import binom
from scipy.stats import beta
from scipy.stats import norm
from scipy.stats import gamma
import math

def posterior_from_conjugate_prior(**kwargs):
    if kwargs['Likelihood_Dist_Type'] == 'Binomial':
        # Get the parameters for the likelihood and prior distribution from the key wor
        x = kwargs['x'] # This is state space of possible values for p = 'probability o
        n = kwargs['n'] # This is the number of Bernoili trials.
        k = kwargs['k'] # This is the number of 'successes'.
        a = kwargs['a'] # This is the parameter alpha for the prior Beta distribution
        b = kwargs['b'] # This is the parameter beta for the prior Beta distribution

        print(f'a_prime = {k + a}.')
        print(f'b_prime = {n - k + b}.')
        Likelihood = binom.pmf(p=x, n=n, k=k)
        Prior = beta.pdf(x=x, a=a, b=b)
        Posterior = beta.pdf(x=x, a=k+a, b=n-k+b)

        return [Prior, Likelihood, Posterior]

    elif kwargs['Likelihood_Dist_Type'] == 'Gaussian_Known_Variance':
        # Get the parameters for the likelihood and prior distribution from the key wor
        x = kwargs['x'] # This is state space of possible values for x in (-infinity,in
        mu = kwargs['mu'] # This is the mean from the data
        var = kwargs['var'] # This is the variance from the data
        prior_mu = kwargs['prior_mu'] # This is the mean for the prior on mu
        prior_var = kwargs['prior_var'] # This is the variance for the prior on mu
        n = kwargs['n']
        #print(kwargs)

        # To answer the challenge question, modify this section with the correct formul
        xsum = n * mu
        print(f'x sum = {xsum}.') # n * mu

        var_prime = 1 / ((1/prior_var) + (n/var))

        mu_part1 = var_prime
        mu_part2a = prior_mu / prior_var
        mu_part2b = xsum / var
        mu_part2 = (mu_part2a + mu_part2b)
        mu_prime = mu_part1 * mu_part2

        print(f'mu_prime = {mu_prime}.')
        print(f'var_prime = {var_prime}.')
        Likelihood = norm.pdf(x=x, loc=mu, scale=math.sqrt(var))
        Prior = norm.pdf(x=x, loc=prior_mu, scale=math.sqrt(prior_var))
```

```
        Posterior = norm.pdf(x=x, loc=mu_prime, scale=math.sqrt(var_prime))
        #print(Posterior)

        return [Prior, Likelihood, Posterior]
    elif kwargs['Likelihood_Dist_Type'] == 'Gamma_Known_Shape':
        # Get the parameters for the likelihood and prior distribution from the key wor
        x = kwargs['x'] # This is state space of possible values for x in (-infinity,in
        alpha = kwargs['alpha'] # alpha for data
        prior_alpha = kwargs['prior_alpha'] # alpha for prior
        n = kwargs['n']
        #print(kwargs)

        alpha_prime = prior_alpha + (n * alpha)

        print(f'alpha_prime = {alpha_prime}.')
        # gamma.pdf(x, a, loc=0, scale=1)
        Likelihood = gamma.pdf(x=x, a=alpha) # assuming N(0,1)
        Prior = gamma.pdf(x=x, alpha=prior_alpha) # assuming N(0,1)
        Posterior = norm.pdf(x=x, a=alpha_prime) # assuming N(0,1)
        #print(Posterior)

        return [Prior, Likelihood, Posterior]

    else:
        print('Distribution type not supported.')
        return -1, -1, -1
```

In [39]:

```
x = np.arange(0, 100, 1)
n = 500
Prior, Likelihood, Posterior = posterior_from_conjugate_prior(Likelihood_Dist_Type='Gau
printProbPlots(x, Prior, Likelihood, Posterior, "Q7 Plot")
```

```
x sum = 20000.
mu_prime = 40.006995103427606.
var_prime = 0.04197062056560408.
```