```
import pandas as pd
import numpy as np
```

(a)

```
def empirical_dist(data, x):
    f = np.sum(data == x) / data.shape[0]
    return f

def plug_in_mean(data):
    return sum([val * empirical_dist(data, val) for val in data])

def plug_in_var(data):
    return sum([(val ** 2) * empirical_dist(data, val) for val in data]) - (plug_
```

```
df = pd.read_csv("ex5.csv")
lsat_mean = df['LSAT'].mean()
gpa_mean = df['GPA'].mean()
```

```
# calculate plug-in estimator using formula from tutorial
def corr_estimator(df):
    plug_in_corr = ((df['LSAT'] - lsat_mean) * (df['GPA'] - gpa_mean)).sum() / \
                np.sqrt(((df['LSAT'] - lsat_mean) ** 2).sum() * ((df['GPA'] - gpa
    return plug_in_corr
```

```
corr = corr_estimator(df)
print(f"The plug-in estimate for the correlation coefficient between LSAT score a
```

```
The plug-in estimate for the correlation coefficient between LSAT score and GPA
```

(b)

```
B = 1000
corr_df = np.zeros(B)
for i in range(B):
    boot = df.sample(n=15, replace=True)
    corr_df[i] = corr_estimator(boot)

se_boot = np.std(corr_df)
print(f"Std: {se_boot}")
```

```
Std: 0.13374187005828042
```

(c)

```
# Gaussian approximation
print(f"Confidence interval for correlation coefficient under Gaussian assumption
print("[{corr - 2*se_boot}, {corr + 2*se_boot}]")
# Pivotal approximation
low = 2*corr - np.quantile(corr_df, 0.975)
high = 2*corr - np.quantile(corr_df, 0.025)
print(f"Pivotal confidence interval for correlation coefficient: ")
print(f"[{low}, {high}]")
# Quantile based approximation
print(f"Quantile-based confidence interval for correlation coefficient: ")
print(f"[{2*corr - high}, {2*corr - low}]")
```

```
Confidence interval for correlation coefficient under Gaussian assumption:
[{corr - 2*se_boot}, {corr + 2*se_boot}]
Pivotal confidence interval for correlation coefficient:
[0.5954261057122939, 1.0970028227440025]
Quantile-based confidence interval for correlation coefficient:
```

```
[0.4557461598348116, 0.9573228768665202]
```