Midterm PSTAT 134/234 (Spring 2024)

PART I: Multiple choice questions

For the next questions, fill in the parentheses with an (X) for the correct answer.

1. What does each eigenvalue represent in the context of PCA?

- The amount of variance explained by each principal component.(X)
- The total variance of the original data.()
- The correlation between the original variables.()
- The scaling factor applied to the principal components.()

2. Suppose you are writing a python simulation program in which you want to represent a scalar random variable, $X \sim N(\mu, \sigma^2)$. Matplotlib histograms are created from variable `x` that consist of simulated observations from the random variable X. Using `x`, two histograms are created as follows:

fone, bins, patches = plt.hist(x, bins=bins, density=True) ftwo, bins, patches = plt.hist(x, bins=bins, density=False)

Assume all bins defined by `bins` are of width one. Which of the following statements are true about `fone` and `ftwo`? Mark all that apply.

- `fone` approximates a continuous distribution (X)
- `ftwo` approximates a continuous distribution ()
- `fone` approximates a discrete distribution ()
- `ftwo` approximates a discrete distribution (X)
- `fone` is equal to `ftwo/sum(ftwo)` (X)
- `ftwo` is equal to `fone/sum(fone)` ()

3. Suppose we perform PCA on the centered data set Y. This involves the eigen decomposition of:

- The original data matrix Y. ()

- The covariance matrix $Y^T Y$. (X)
- The singular vectors of Y. ()
- The singular values of $Y^T Y$. ()

PART II: Python coding

Data description: Insurance Claims

The data given in the file *Insurance.csv* consists of the number of car insurance claims made by policyholders in the third quarter of 2020 and 2021.

Read Data into Python

Numpy and Pandas is used to read in the csv file into python.

In [1]:
```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
Claims = pd.read_csv("Insurance.csv")
Claims.head()
```

Out[1]:

	claims	year
0	38	2020
1	35	2020
2	20	2020
3	156	2020
4	63	2020

Question 1a: Subset Data

- Filter the data to only include rows in which the `year` is 2020. **From now on, all questions must be answered based on this filtered data set**.
- Assume that the number of claims is a random variable X with unknown distribution $F(X)$. Suppose we want to estimate the probability that the number of claims exceeds 50 ($\theta = P(X > 50)$).

Based on the given data calculate $\hat{\theta}$ as an estimate of this probability:

In [2]:
```
# Fill-in ...
claims_2020 = Claims[Claims["year"] == 2020]
claims_subset = claims_2020[claims_2020["claims"] > 50]
claims_count = claims_subset['claims'].count()
claims_mean = claims_subset["claims"].mean()
print(claims_count)
print(claims_mean)

theta_hat = claims_count / len(claims_2020)
print(theta_hat)
```

```
17
98.94117647058823
0.27419354838709675
```

Question 1b: Model-free resampling

Use non-parametric bootstrap to generate the epirical distribution of $\hat{\theta}$.

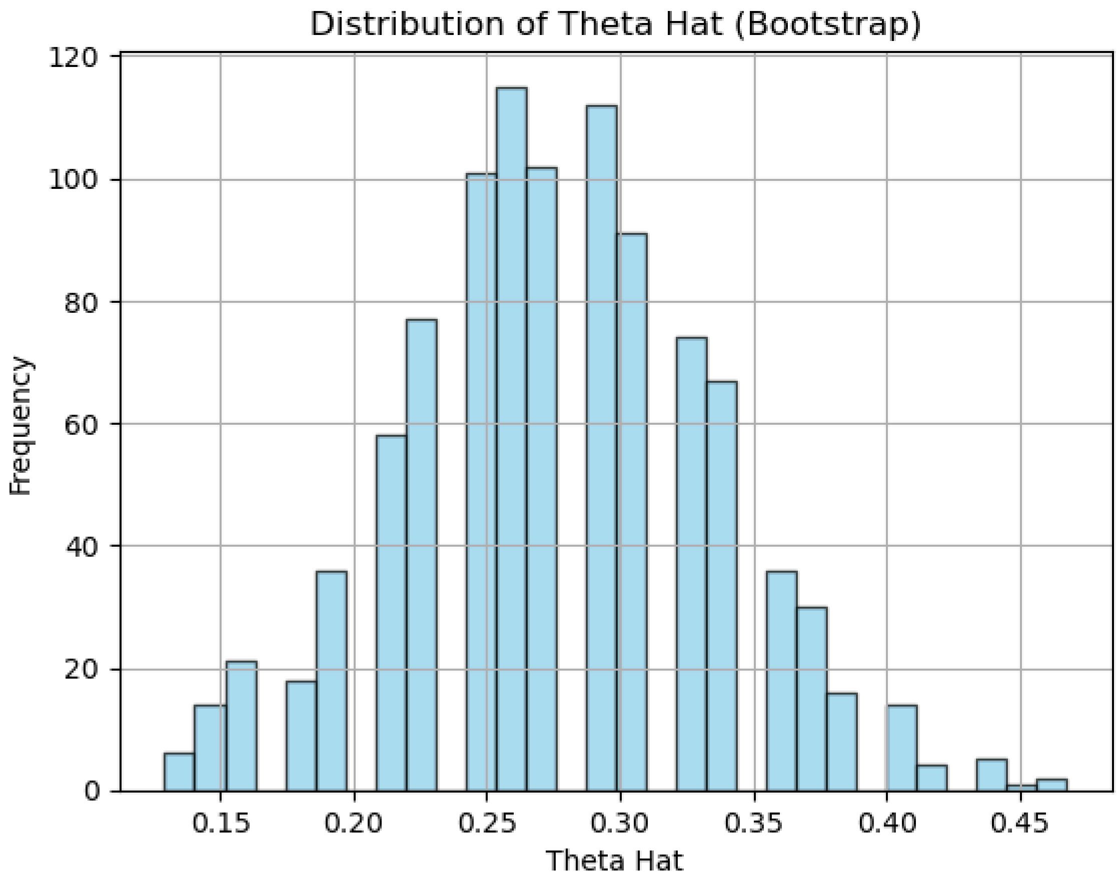
1. Write a function named `bootstrap_data_theta` that can take `data_in`, as input. Inside `bootstrap_data_theta` function, you will:

- Create a pseudo-data by using numpy.random.choice
- Calculate and return $\hat{\theta}$

2. Then, run the function `bootstrap_data_theta` function 1000 times, storing the resulting 1000 estimates of theta in a list.

3. Make a histogram showing the distribution of $\hat{\theta}$.

In [3]:
```python
# Fill-in ...
def choose_from_data(n=1, data_in=None):
    from numpy.random import choice

    return choice(data_in, n, replace=True)

def bootstrap_data_theta(data_in):
    n = len(data_in)

    # randomly sample with replacement
    pseudo_data = choose_from_data(n, data_in)

    # compute probability theta
    m = np.sum(pseudo_data > 50)
    bootstrap_theta = m / n

    return bootstrap_theta

repeat_resampling = 1000

# Run bootstrap_data_theta function `repeat_resampling` times
theta_hat_list = [bootstrap_data_theta(claims_2020["claims"]) for _ in range(repeat_resampling)]

## Histogram
plt.hist(theta_hat_list, bins=30, color='skyblue', edgecolor='black', alpha=0.7)
plt.xlabel('Theta Hat')
plt.ylabel('Frequency')
plt.title('Distribution of Theta Hat (Bootstrap)')
plt.grid(True)
plt.show()
```



Type your answer here, replacing this text.

Question 1c: Bootstrap Confidence Interval

Construct a 95% confidence interval for θ based on the boostrap samples of $\hat{\theta}$. You can use any method (normal interval, pivotal interval or percentile interval).

Hint: In order to calculate quantiles for any variable you can refer to numpy.quantile

In [4]:
```
# Fill-in ...
lower = np.quantile(theta_hat_list, 0.025)
```

```
upper = np.quantile(theta_hat_list, 0.975)
print(lower, upper)
```

```
0.16129032258064516 0.4032258064516129
```

Question 1d: Bias

Calculate the Bootstrap bias estimate.

In [5]:
```
# Fill-in ...
bias = np.mean(theta_hat_list) - theta_hat
print(bias)
```

```
0.0017419354838709 71
```

(PSTAT 234) Question 1e: Model-based bootstrap

Now suppose we want to estimate $\mu = E(X)$. Use parametric bootstrap to generate the empirical distribution of $\hat{\mu}$. For this strategy we assume a population distribution $f(x \mid \mu)$; pick an addequate distribution (Normal, poisson or Binomial) and follow the steps:

1. Estimate $\hat{\mu}$ based on the data.
2. Sample from $f(x \mid \hat{\mu})$
3. repeat steps 1 and 2, 1000 times.

- Make a histogram showing the distribution of $\hat{\mu}$.

In [6]:
```
# Fill in...

def bootstrap_data_mu(data_in):

    n = len(data_in)

    # randomly sample with replacement
    pseudo_data =  ...
    # compute mu hat
    bootstrap_mu = ...

    return(bootstrap_mu)
```

```
repeat_resampling = ...

# Histogram
...
```

Out[6]: Ellipsis

(PSTAT 234) Question 1f: Expected value and Variance

Based on the bootstrap resampling calculate estimates for $E(\hat{\mu})$ and $Var(\hat{\mu})$. How these values compare to the theoretical ones? (*Theoretical*: $E(\hat{\mu}) = \mu$, $Var(\hat{\mu}) = \frac{Var(X)}{n}$)

In [7]:
```
## Fill in:

E_theta_hat= ...
Var_theta_hat = ...
print(E_theta_hat,Var_theta_hat)
```

```
Ellipsis Ellipsis
```

Type your answer here, replacing this text.

Intentionally Blank

Question 2a:

Suppose $X \sim F$, with $X \in \{-5, -4, -3, -2, -1, 0, 1, 2\}$.

By using np.random.choice, generate 100 `samples` of the random variable X, according to the given `probabilities`.

In [8]:
```
## Fill in...
import numpy as np

probabilities = [0.1, 0.1, 0.1, 0.2, 0.2, 0.1, 0.1, 0.1]
values = [-5, -4, -3, -2, -1, 0, 1, 2]

samples = np.random.choice(values, size=100, p=probabilities)
```

Question 2b:

- Explain line by line what the following snippet is doing:

In [9]:
```
def Ecdf(data):
    sorted_sample = np.sort(data)
    n = sorted_sample.size
    unique_values, counts = np.unique(sorted_sample, return_counts=True)
    cumulative_counts = np.cumsum(counts)

    return pd.DataFrame({'X': unique_values, 'F(X)': cumulative_counts / n})
```

1: define an empirical cdf function

2: sort the data

3: define sample size of the sorted sample

4: define the unique_values of the distribution and the correponding counts of each unique value

5: calculate the cumulatie counts by summing up all the counts

6: return a pandas dataframe where column 'X' contains the unique values and 'F(x)' contains the average counts of each unique value

Question 2c:

- By using function Ecdf and the sample that you generated, calculate $P(X = 0)$:

In [10]:
```
## Fill in..

def Ecdf(data):
    sorted_sample = np.sort(data)
    n = sorted_sample.size
    unique_values, counts = np.unique(sorted_sample, return_counts=True)
    cumulative_counts = np.cumsum(counts)
```

```
    return pd.DataFrame({'X': unique_values, 'F(X)': cumulative_counts / n})

ecdf_df = Ecdf(samples)
prob_0 = ecdf_df[ecdf_df['X'] == 0]['F(X)'].values[0]
print(prob_0)
```

```
0.8
```

Submission Checklist

1. Save file to confirm all changes are on disk
2. Run *Kernel* > *Restart & Run All* to execute all code from top to bottom
3. Save file again to write any new output to disk
4. Select *File* > *Save and export Notebook as/* > *HTML*.
5. Open in Google Chrome and print to PDF.
6. Submit to Gradescope