MCEN 3047 - Homework 3

Jack Reilly Goldrick

November 2, 2024

0.1 Problem 1

0.1.1 Part A & B

- Mean:
  - Indoor: 9.842105
  - Outdoor: 7.773684
- Std:
  - Indoor: 1.844305
  - Outdoor: 0.983103
- Median:
  - Indoor: 9.4
  - Outdoor: 7.6
- Mode:
  - Indoor: 9.2
  - Outdoor: 7.2

0.1.2 Part C

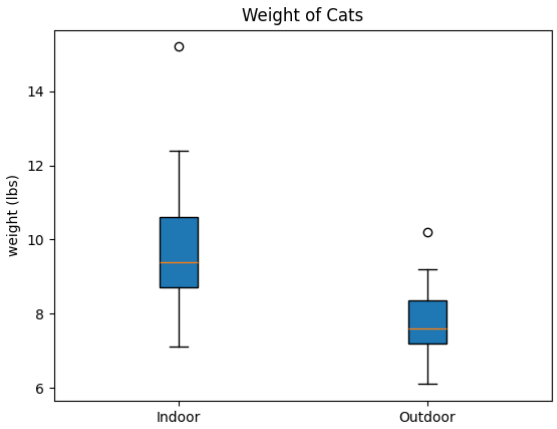

Figure 1: Box Plot where Each Category has 1 outlier defined by the Open Circle

0.1.3 Part D

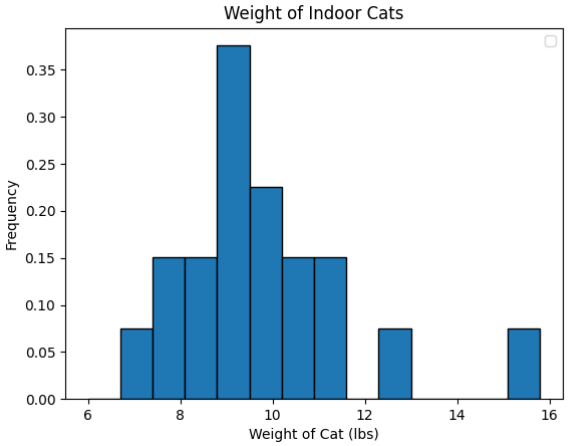

Figure 2: Indoor Histogram

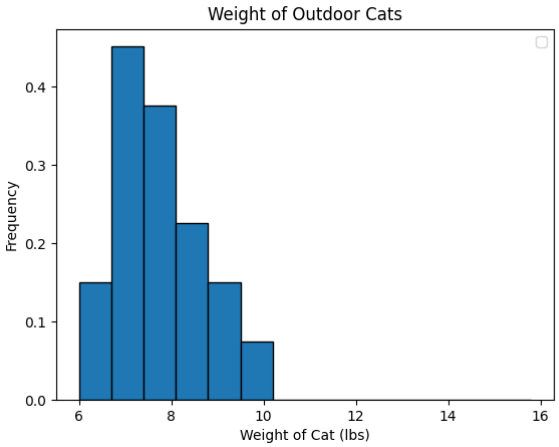

Figure 3: Outdoor Histogram

0.1.4 Part E

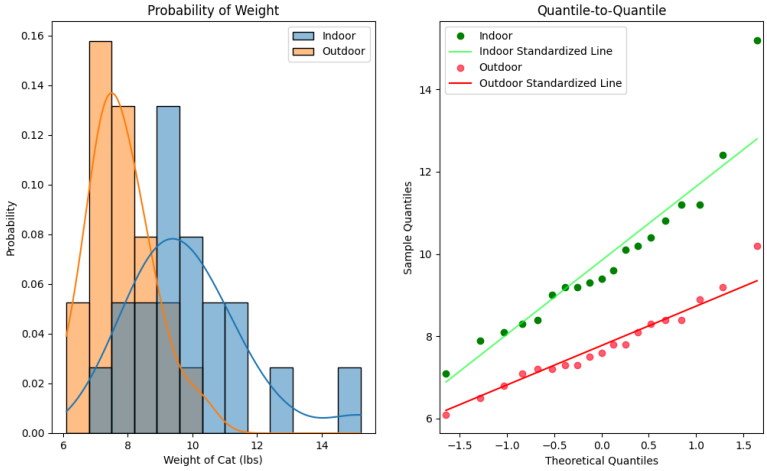

Figure 4: Norm Plot with a QQ-Plot that clearly shows the distribution models the data well

0.1.5 Part F

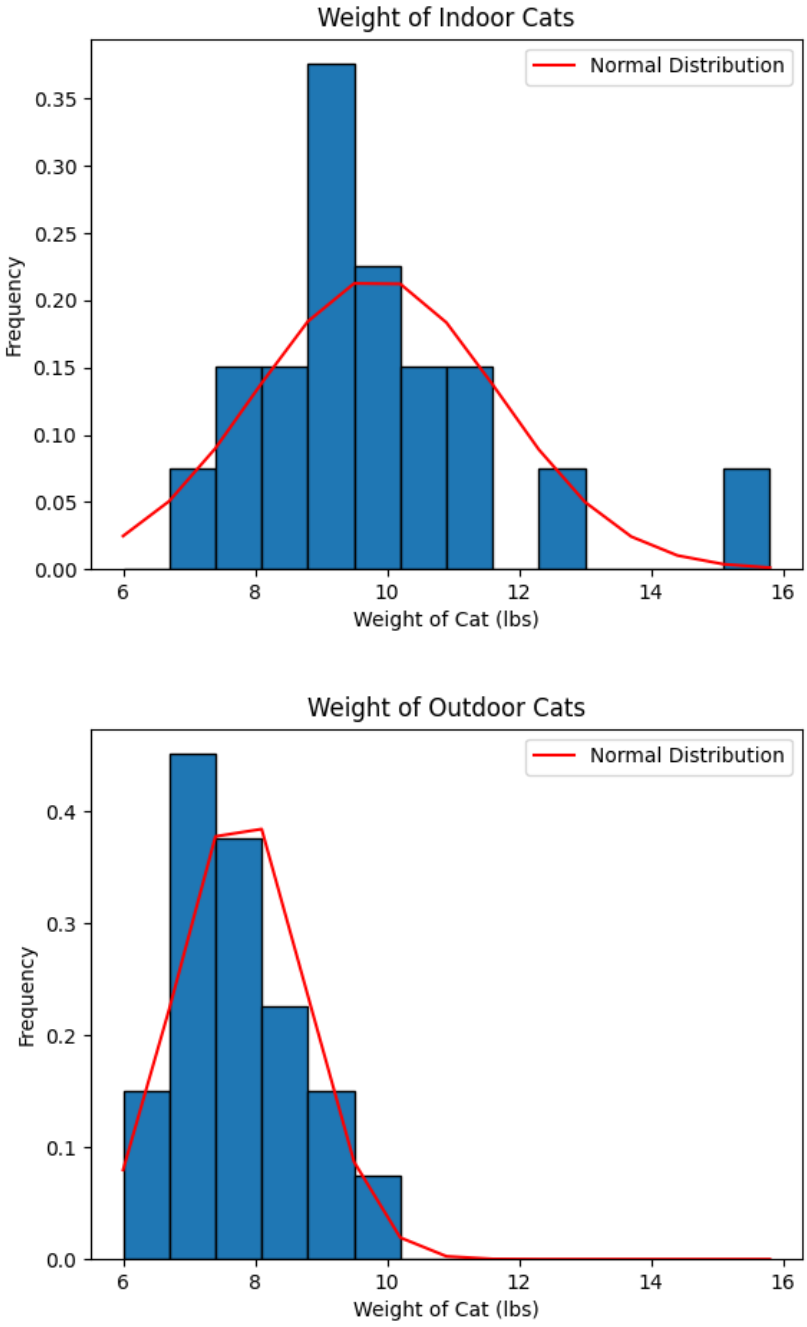

Figure 5: The Figure Shows the Distribution is an accurate Representation of the Data due to the high similarity IV

0.2 Problem 2

0.2.1 Part A

- Let $I = [0, 120] \subset \mathbb{R}$
- Let $(I,\ \mathcal{P}(I),\ \mu)$ be a Probability Space such that:
  - $\mu(I) = 1$
  - $\mathbf{X} : \mathcal{P}(I) \to \mathbb{R}$ is an Independent Variable with
    - $f_x : \mathbb{R} \to [0, \infty]$ such that

$$f_x = \frac{x - 20}{5000}$$

  - Thus by definition, Since the measure sends a variable to the interval $[0, 1]$ we have:
    - $\mu(\mathcal{P}(I)) = P_x(\mathcal{P}(I)) = \int_{\mathcal{P}(I)} f_x(x)\, dx$
- Thus the portion of students who take (60, 120) minutes is defined as:
  - $P_x([60, 120]) = \int_{[60,120]} f_x(x)\, dx = \int_{60}^{120} \frac{x-20}{5000}\, dx$
    - Therefore the portion is 84% of the students

0.2.2 Part B

- Measuring the Expected Value of the Independent Variable X, $\mathbf{EX}$, with $f_x : \mathbb{R} \to [0, \infty]$ we have:

$$\mathbf{EX} = \int_I x f_x(x)\, dx = \int_{20}^{120} \frac{x^2 - 20x}{5000}\, dx$$

$$\mathbf{EX} = 86\ \frac{2}{3}\ \text{Minutes}$$

0.2.3 Part C

- Measuring the Standard Deviation of the Independent Variable X, σ, with $f_x : \mathbb{R} \to [0, \infty]$ we have:

$$\sigma = \left(\int_I (x - \mathbf{EX})^2 f_x(x)\, dx \right)^{\frac{1}{2}} = \left(\int_{20}^{120} (x - \frac{260}{3}) \frac{x - 20}{5000}\, dx \right)^{\frac{1}{2}}$$

$$\sigma = 23.570\ \text{Minutes}$$

0.3 Problem 3

0.3.1 Part A

- Let $I = [0, \infty] \subset \mathbb{R}$
- Let $(I,\ \mathcal{P}(I),\ \mu)$ be a Probability Space such that:
  - $\mu(I) = 1$
  - $\mathbf{X} : \mathcal{P}(I) \to \mathbb{R}$ is an Independent Variable with
    - $f_x : \mathbb{R} \to [0, \infty]$ such that

$$f_x = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

  - Thus by definition, Since the measure sends a variable to the interval $[0, 1]$ we have:
    - $\mu(\mathcal{P}(I)) = P_x(\mathcal{P}(I)) = \int_{\mathcal{P}(I)} f_x(x)\, dx$
- Thus the Probability of $\mathcal{P}(I) = \{[24000, \infty]\}$ is defined as:
  - $P_x([24000, \infty]) = \int_{[24000,\infty]} f_x(x)\, dx = \int_{24000}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}\, dx$
    - Therefore the probability a light lasts more than 24000 hrs is

$$74.7507\%$$

0.3.2 Part B

- Using The Cumaltive Distribution Function we have:
  $P_x(\mathbf{X} \leq 26012) = \int_{[0,26012]} f_x(x)\, dx = \int_0^{26012} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}\, dx$

$$P_x(\mathbf{X} \leq 26012) \approx 75.00\%$$

0.3.3 Part C

- Using The Cumaltive Distribution Function we have:
  $P_x(\mathbf{X} \leq 27500) = \int_{[0,27500]} f_x(x)\, dx = \int_0^{27500} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}\, dx$

$$P_x(\mathbf{X} \leq 27500) \approx 95.22 \text{ Percentile}$$

0.3.4 Part D

- Using The Measure of the Probability Density Function we have:
  $P_x(20000 < \mathbf{X} < 30000) = \int_{[20000,30000]} f_x(x)\,dx = \int_{20000}^{30000} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}\,dx$

$$P_x(20000 < \mathbf{X} < 30000) \approx 99.22 \text{ Percentile}$$

0.4 Problem 4

0.4.1 Part A

- Mean: 86.67018074199994
- Variance: 555.912639

0.4.2 Part B

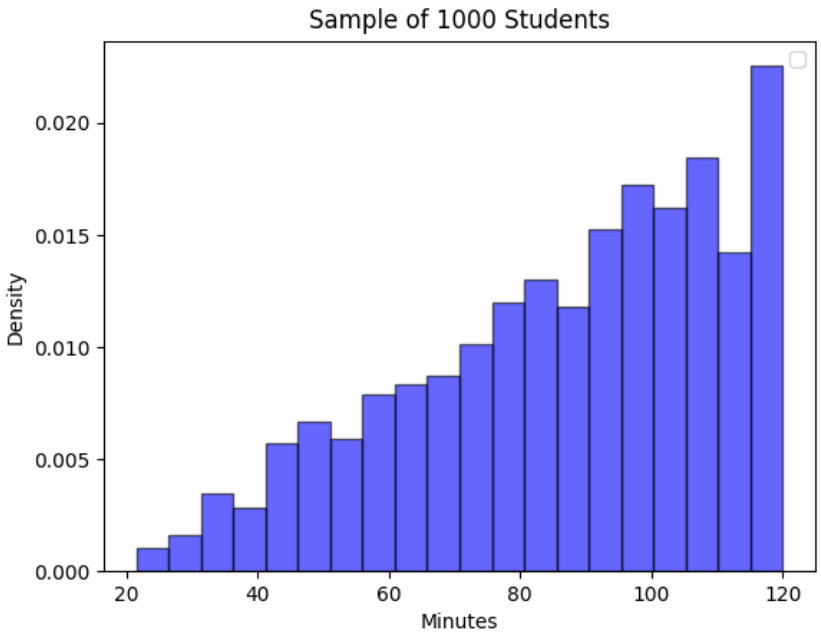

Figure 6:

0.4.3 Part C

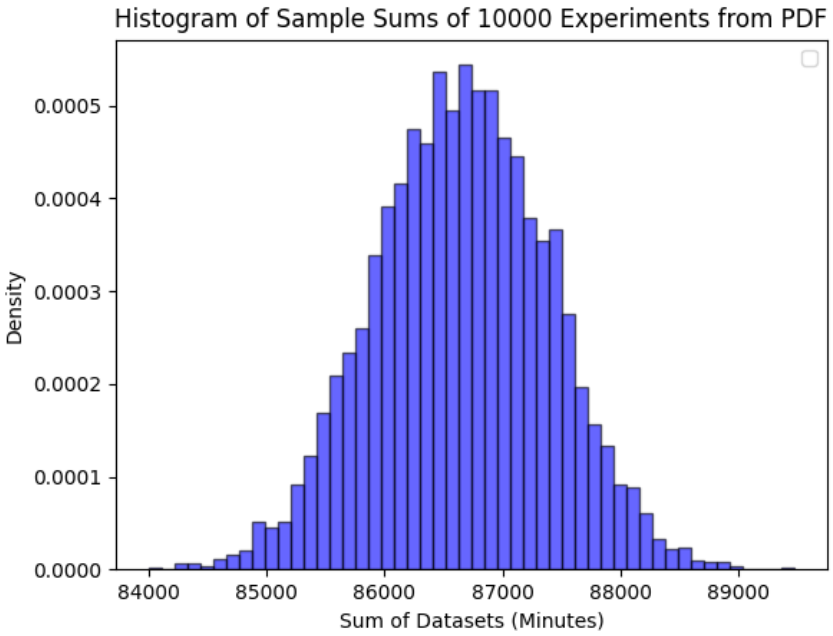

Figure 7:

0.4.4 Part D

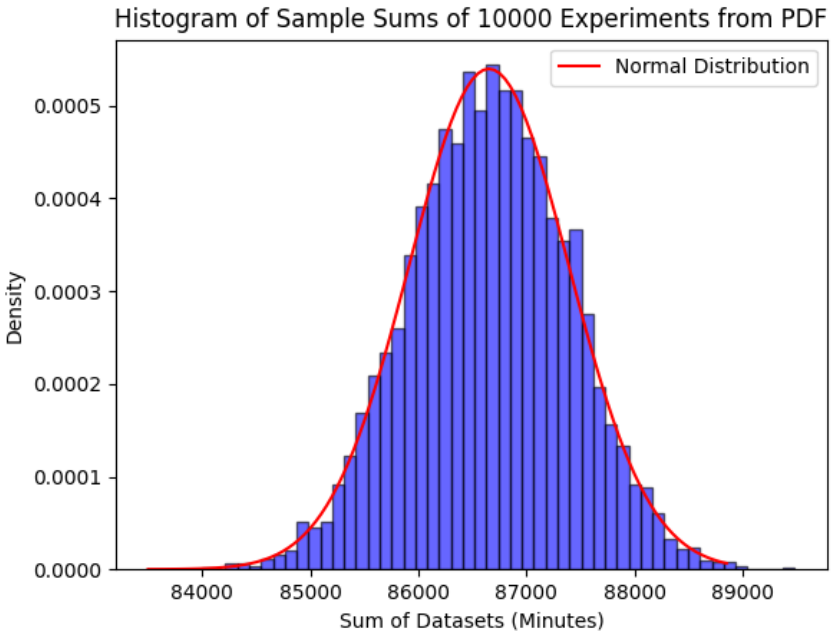

Figure 8:

- Mean: 86670.180742
- Variance: 558185.236937
- Both the Mean and Variance are approximately scaled proportionally by the number of summed samples.

0.5 Code

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from random import sample
import torch as tc
from scipy.stats import norm
import statistics
import seaborn as sns
import scipy.stats as sc
import statsmodels.graphics.gofplots as sm

class set_4:

    class p1:

        def __init__(self):
            self.frame = pd.read_csv("../data/cat.csv")
            self.frame.columns = ["Indoor", "Outdoor"]
            self.data = self.frame.to_numpy()
            print(f"Mean: \n {self.frame.mean()}")
            print(f"Std: \n {self.frame.std()}")
            print(f"Median: \n {self.frame.median()}")
            print(f"Mode: \n {self.frame.mode(0)}")


        def box_plot(self, title_name='Weight of Cats',labels=["
            ↪ Indoor", "Outdoor"], ylabel="weight (lbs)"):
            fig, ax = plt.subplots()
            ax.boxplot(self.data, tick_labels=labels, patch_artist
                ↪ =True)
            ax.set_title('Weight of Cats')
            ax.set_ylabel(ylabel)
            plt.show()


        def quick_hist(self, xlabel="Weight of Cat (lbs)", ylabel=
            ↪ "Frequency", width=0.7, indoor=True, Line=None):
            if indoor:
                data = self.data[:, 0]
                title="Weight of Indoor Cats"
            else:
                data = self.data[:, 1]
                title="Weight of Outdoor Cats"
```

```
        bins = np.arange(6, 16, width)
        norm_data, _, _ = plt.hist(data, density=True, bins=
            ↪ bins, edgecolor = "black")
        if Line:
            mu, sigma = Line
            plt.plot(bins, norm.pdf(bins, mu, sigma), color='r'
                ↪ , label='Normal Distribution')

        plt.title(title)
        plt.xlabel("Weight of Cat (lbs)")
        plt.ylabel(ylabel)
        plt.legend()
        plt.show()
        return norm_data

    def plot_normal_dist(self, h_title="Probability of Weight"
        ↪ , xlabel="Weight of Cat (lbs)", width=0.7):
        fig, ax = plt.subplots(1, 2, figsize=(12, 7))
        sns.histplot(self.frame, binwidth=width, stat='
            ↪ probability', kde=True,ax=ax[0], cbar=True)
        pp_i = sm.ProbPlot(self.frame['Indoor'])
        pp_iQ = pp_i.qqplot(line='s', ax=ax[1]).set_label("
            ↪ Indoor")
        pp_o = sm.ProbPlot(self.frame['Outdoor'])
        pp_o.qqplot(line='s', ax=ax[1]).set_label("Outdoor")
        points = ax[1].get_lines()
        points[0].set_markerfacecolor('green')
        points[0].set_markeredgecolor('green')
        points[0].set_label("Indoor")
        points[1].set_color((.1, 1, .2, .7))
        points[1].set_label("Indoor Standardized Line")
        points[2].set_markerfacecolor((1, .1, .2, .7))
        points[2].set_markeredgecolor((1, .1, .2, .7))
        points[2].set_label("Outdoor")
        points[3].set_label("Outdoor Standardized Line")
        ax[1].legend()
        ax[1].set_title("Quantile-to-Quantile")
        ax[0].set_title(h_title)
        ax[0].set_xlabel(xlabel)
        plt.show()

    def check_normal_dist(self):
        params_i = sc.anderson(self.frame['Indoor']).
            ↪ fit_result
        mean_i = params_i.params[0]
```

```
        stdev_i = params_i.params[1]
        print(f"Indoor Anderson-Darling Test: {params_i.
            ↪ message}")
        print(f" Mean: {mean_i}")
        print(f" Standard Deviation: {stdev_i}")

        params_o = sc.anderson(self.frame['Outdoor']).
            ↪ fit_result
        mean_o = params_o.params[0]
        stdev_o = params_o.params[1]
        print(f"Outdoor Anderson-Darling Test: {params_o.
            ↪ message}")
        print(f" Mean: {mean_o}")
        print(f" Standard Deviation: {stdev_o}")

        self.quick_hist(indoor=True, Line=[mean_i, stdev_i])
        self.quick_hist(indoor=False, Line=[mean_o, stdev_o])



    def run(self):
        self.box_plot(self.data, ylabel="weight (lbs)")
        self.quick_hist(self.data, indoor=True)
        self.quick_hist(self.data, indoor=False)
        self.plot_normal_dist()
        self.check_normal_dist()


class p4:
    def __init__(self):
        self.frame = pd.read_csv("../data/dist.csv")
        self.data = self.frame.to_numpy()


    def quick_hist(self,data=None, size=1000, bins=20, x_label
        ↪ ="Minutes", y_label="Density", width=1, Line=None,
        ↪ title=None):
        if data is None:
            data = self.get_random_sample(size=size)

        if title is None:
            title = f"Histogram of Sample of {size} Students"

        if Line is not None:
            mu, sigma = Line
            buck = np.arange(83500, 88900, bins)
```

```
        plt.plot(buck, norm.pdf(buck, mu, sigma), color='r'
            ↪ , label='Normal Distribution')
    plt.hist(data, bins=bins, density=True, alpha=0.6,
        ↪ color='blue', edgecolor='black')
    plt.title(title)
    plt.xlabel(x_label)
    plt.ylabel(y_label)
    plt.legend()
    plt.show()

def get_random_sample(self, size=1000):
    df = self.frame
    df.columns = df.columns.str.strip()

    minutes = df['Minutes']
    pdf = df['PDF']

    pdf_normalized = pdf / pdf.sum()

    return np.random.choice(minutes, size=size, p=
        ↪ pdf_normalized)

def conduct_n_sample_experiments(self, n=10000, size=1000)
    ↪ :
    samples = np.concatenate([self.get_random_sample(size=
        ↪ size) for _ in range(n)])
    sums = np.sum(samples.reshape(n, size), axis=1)
    return samples, sums


def check_normal_dist(self, data, bins=50, Line=None):
    params_i = sc.anderson(data).fit_result
    mean_i = params_i.params[0]
    stdev_i = params_i.params[1]
    print(f"Anderson-Darling Test: {params_i.message}")
    print(f" Mean: {mean_i}")
    print(f" Standard Deviation: {stdev_i}")


    self.quick_hist(Line=[mean_i, stdev_i], data=data,
        ↪ title="Histogram of Sample Sums of 10000
        ↪ Experiments from PDF", x_label="Sum of Datasets
        ↪ (Minutes)", bins=bins)
```

```
        def run(self):
            # part a
            self.quick_hist(size=1000, title="Sample of 1000
                ↪ Students")
            # part b
            samps, sums = self.conduct_n_sample_experiments(n
                ↪ =10000, size=1000)
            self.quick_hist(data=sums, title="Histogram of Sample
                ↪ Sums of 10000 Experiments from PDF", x_label="
                ↪ Sum of Datasets (Minutes)", bins=50)
            self.check_normal_dist(sums)



def run():
    s = set_4()
    s.p1().run()
    #
    s.p4().run()


def main():
    run()

if __name__ == "__main__":
    main()
```