به نام خدا

**گزارش کار تمرین 6 فصل 4 کتاب**

**آمار و احتمال مهندسی ترم بهار 1404**

**ماهان بانشی**

## صورت سوال:

**6.** Let's explore the impact of a single outlier on the mean vs. median, for large and small sample sizes. The purpose is to demonstrate the robustness of the median vs. the sensitivity of the mean to a single outlier, and whether that depends on the sample size.

Create a dataset of 50 random numbers drawn from a normal distribution. Compute the mean and median of that distribution, and visualize those two statistics as vertical lines drawn on top of a histogram of the data. Your plot will look something like the top-left panel in Figure 4.32. Next, create one outlier in the dataset by replacing the largest value in the data with itself raised to the $4^{th}$ power (that is, set $x_{max} = x_{max}^4$). Recompute the mean and median and the histogram plot (top-right panel). Then repeat these steps using a dataset of 5000 samples. Your final result will look like Figure 4.32.



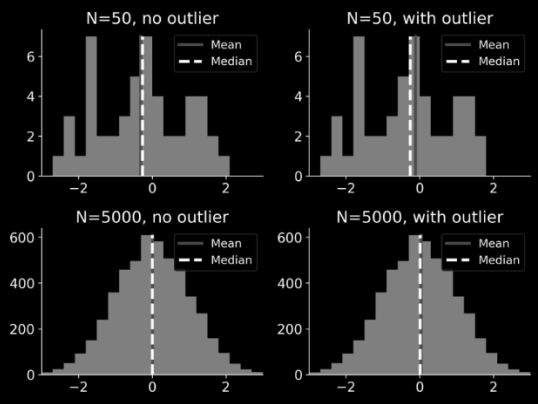



**Figure 4.32: Visualization for Exercise 6.**

Finally, print the shift in the mean and median when adding the outlier. My results are below (obviously the exact numbers will change each time you re-run the code).

the outlier. My results are below (obviously the exact numbers will change each time you re-run the code).

```
With N = 50, the mean increased by 0.62
With N = 50, the median increased by 0.00

With N = 5000, the mean increased by 0.03
With N = 5000, the median increased by 0.00
```

Some observations: The median was completely unaffected by the outlier. That's not surprising because we replaced the largest value with an even-larger value; in other words, we changed a numerical value but did not change its position relative to the midpoint of the data. It's not surprising that the mean was pulled up by the outlier, but it is interesting to see that the impact was much smaller when the sample size was

148

larger. That should be an intuitive result, but is still worth contemplating both why that happened and what it implies for datasets with small vs. large sample sizes.

Another interesting observation is that the $N = 50$ dataset appears to be trimodally distributed. In fact, the data were generated from a unimodal distribution; this apparent triple-peak is simply due to random variability. If you saw this in your data without knowing the underlying generative process, you might come to the conclusion that there are multiple clusters in the data. That is a reasonable conclusion, even though we know that it is false in this case. This highlights a difficulty with interpreting distributional features in small sample sizes that contain noise.

Final thought for this exercise: Now that you have code for this simulation, I encourage you to continue exploring it! Try different sample sizes, different ways of computing outliers, different distributions of data to sample from, and so on.

**7.** The purpose of this exercise is to use code to compute the

## خواسته های سوال:

در این سوال میخواهیم تاثیرات داده های پرت (outliers) را روی میانه و میانگین یک مجموعه داده بررسی کنیم. همچنین این کار را رو دو مجموعه داده با تعداد 50 و 5000 انجام می دهیم ببینیم تعداد داده ها هم تاثیری بر مقدار تغییری که outlier ایجاد میکند دارد یا نه.

برای این کارها 4 تا نمودار ایجاد می کنیم. دو تا 50 تایی و دوتا 5000 تایی. از هرکدام یکی بصورت عادی و دیگری با وجود outlier. نحوه ی تولید outlier این است که داده ی max نمودار را به توان 4 می رسانیم و جایگزین می کنیم.

کدی که برای سوال زدیم:

```
import numpy as np
import matplotlib.pyplot as plt

def add_outlier(data):
    max_value_index = np.argmax(data)
    outlier = data[max_value_index] ** 4
    data[max_value_index] = outlier
    return data

def plot_histogram(data, title, ax, x_limit=None):
    mean_value = np.mean(data)
    median_value = np.median(data)

    if x_limit:
        filtered_data = np.copy(data)
        filtered_data[filtered_data > x_limit[1]] =
x_limit[1]
    else:
        filtered_data = data
```

```
    ax.hist(filtered_data, bins=30, alpha=0.7, color='gray',
edgecolor='black')
    ax.axvline(mean_value, color='red', linestyle='dashed',
linewidth=2, label='Mean')
    ax.axvline(median_value, color='blue',
linestyle='dashed', linewidth=2, label='Median')
    ax.set_title(title)
    ax.legend()

    if x_limit:
        ax.set_xlim(x_limit)

data_50 = np.random.normal(loc=0, scale=1, size=50)
data_50_outlier = add_outlier(np.copy(data_50))
data_5000 = np.random.normal(loc=0, scale=1, size=5000)
data_5000_outlier = add_outlier(np.copy(data_5000))

x_limit_50 = (-4, 4)
x_limit_5000 = (-6, 6)

fig, axes = plt.subplots(1, 2, figsize=(12, 5))
plot_histogram(data_50, "N=50, No Outlier", axes[0],
x_limit_50)
plot_histogram(data_50_outlier, "N=50, With Outlier",
axes[1], x_limit_50)
plt.tight_layout()
plt.show()

fig, axes = plt.subplots(1, 2, figsize=(12, 5))
plot_histogram(data_5000, "N=5000, No Outlier", axes[0],
x_limit_5000)
plot_histogram(data_5000_outlier, "N=5000, With Outlier",
axes[1], x_limit_5000)
plt.tight_layout()
plt.show()

print(f"Mean shift for N=50: {np.mean(data_50_outlier) -
np.mean(data_50):.2f}")
```

```
print(f"Median shift for N=50: {np.median(data_50_outlier) -
np.median(data_50):.2f}")
print(f"Mean shift for N=5000: {np.mean(data_5000_outlier) -
np.mean(data_5000):.2f}")
print(f"Median shift for N=5000:
{np.median(data_5000_outlier) - np.median(data_5000):.2f}")
```

## توضیح کلی کد:

ابتدا یک تابع تعریف کرده ایم که یک مجموعه داده میگیرد و به آن outlier اضافه میکند و برمیگرداند. نحوه ی اضافه کردن outlier هم در بالا توضیح دادیم.

سپس یک تابع دیگر داریم که هیستوگرام ها را تولید می کند. در این نمایش ها میانگین و میانه داده ها هم مشخص است.

در آخر هم دو مجموعه داده رندوم تولید می کنیم. ابتدا خودشان را نمایش می دهیم . سپس خودشان را به همراه داده ی پرتی که از متد بالا اضافه شده نشان می دهیم. هدفمان بررسی تفاوت های میانه و میانگین این دو نمودار است. یکی از نمودار ها 50 عضو دارد و دیگری 5000 عضو.

## قسمت های خاص کد:

```
    outlier = data[max_value_index] ** 4
    data[max_value_index] = outlier
```

داده max را به توان 4 می رسانیم.

```
    mean_value = np.mean(data)
    median_value = np.median(data)
```

میانگین و میانه را خود نامپای به ما می دهد.

```
    if x_limit:
        filtered_data = np.copy(data)
        filtered_data[filtered_data > x_limit[1]] =
x_limit[1]
    else:
        filtered_data = data
```

این بخش از کد فیلتر پایین گذر را اعمال می کند.

```
    ax.axvline(mean_value, color='red', linestyle='dashed',
linewidth=2, label='Mean')

    ax.axvline(median_value, color='blue',
linestyle='dashed', linewidth=2, label='Median')
```

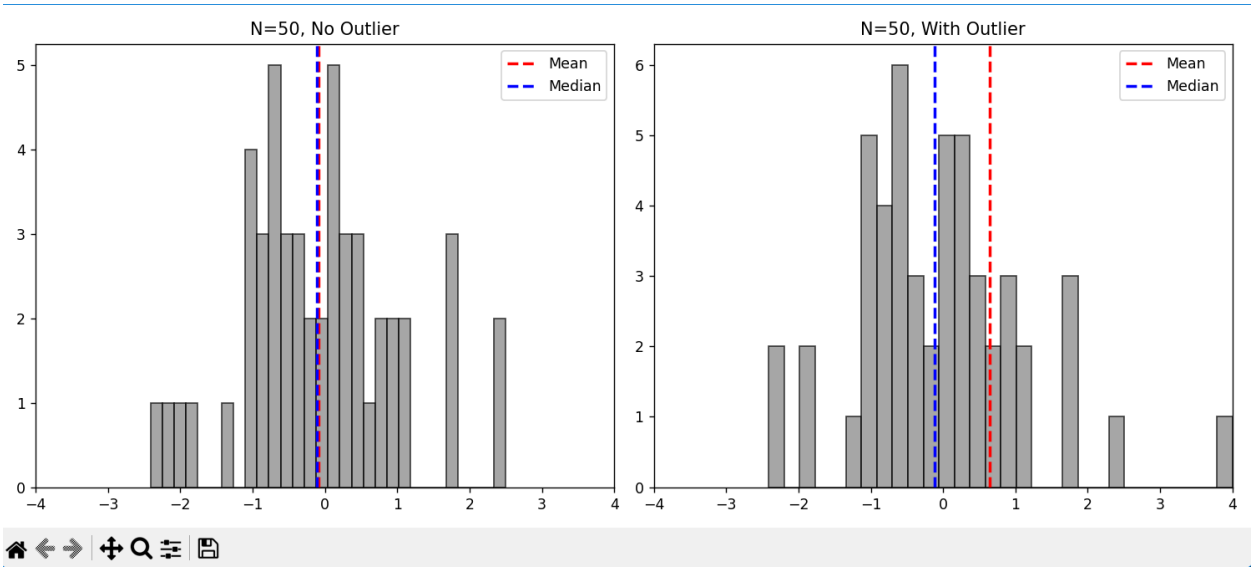
در اینجا خطوط میانه و میانگین را نمایش می دهیم.

```
data_50 = np.random.normal(loc=0, scale=1, size=50)
```

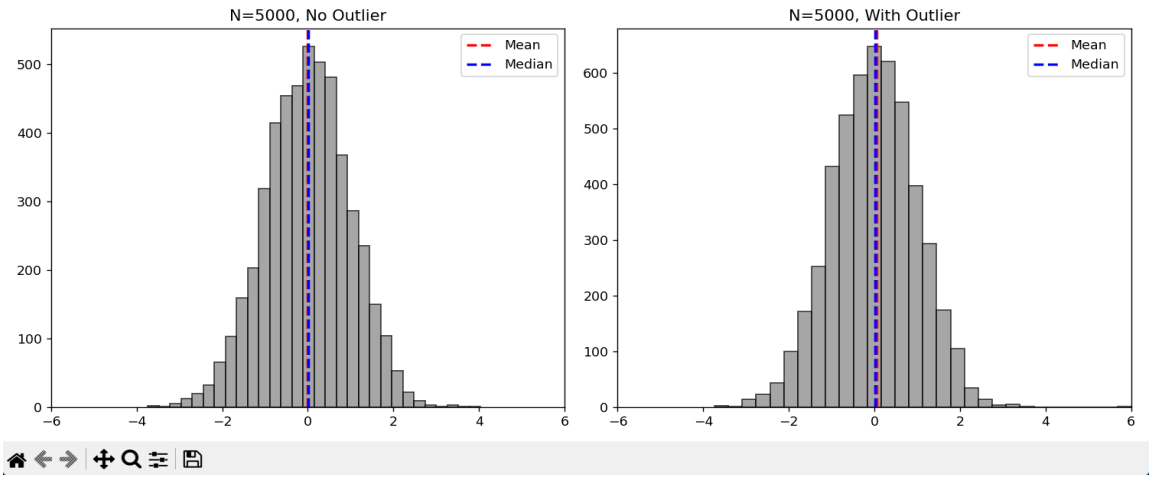
آرگومان اول مرکز پخش است، آرگومان دوم انحراف معیار است و آرگومان سوم تعداد داده ها.

## خروجی ها:

نمونه ی 50 تایی:




نمونه ی 5000 تایی:



نکته در مورد نمودار ها: در این نمودار ها ما مقدار دقیق outlier را نمیبینیم و outlier در راست نمودار ها است. دلیل این موضوع این است که میخواهیم نمودار های زیباتر بیفتند و اگر عدد outlier را روی محور x نمایش دهیم شکل نمودار ها جالب نمی شود.

خروجی کنسول:

Mean shift for N=50: 0.30

Median shift for N=50: 0.00

Mean shift for N=5000: 0.05

Median shift for N=5000: 0.00

## نتیجه گیری:

از شکل نمودار ها نکات زیر برآورد می شود:

1: میانه و میانگین با هم تفاوت دارند و یکسان نیستند.

2: مقدار میانه ی ما در outlier اضافه کردن تغییری نمی کند. زیرا این outlier ترتیب داده ها را تغییر نمی دهد و خودش همواره داده ی ماکسیمم است. ولی outlier مقدار میانگین را تغییر می دهد.

3: وقتی تعداد داده ها زیاد میشود تاثیری که outlier روی میانگین و اختلاف آن از میانه می گذارد خیلی کمتر می شود. (از شکل های خروجی و خروجی کنسول هم کاملا واضح است.) دلیل این موضوع فرمول ریاضی میانگین است که در آخر جمع اعداد بر تعداد کل تقسیم می شود و هر چه این تعداد بیشتر باشد مقدار تغییر میانگین کمتر میشود. برای مثال یک داده ی outlier در برابر 5000 داده ی دیگر نقش ناچیزی دارد.