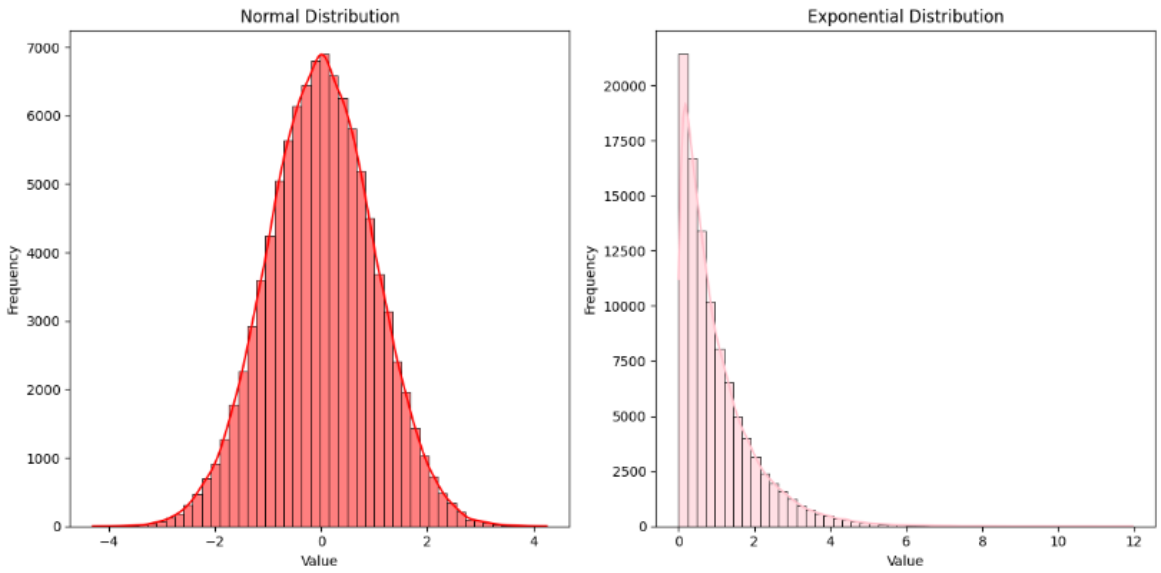
Python에서 서로 다른 두 분포로 부터 100000개의 sample data를 generation 하고 distribution을 확 인 할 수 있는 plot을 각각 그린다. (10) •

정규분포와 지수분포로 100000개의 sample data를 만들고 plot그림



각 sample data로부터 아래의 통계량을 계산하는 코드를 내부 함수 or vectorize 를 사용하여 / 사용 하지 않고 각각 작성하고 실행 소요 시간을 비교 (20) – Mean / Median / Variance / MAD / Skewness / Kurtosis / IQR

```
vectorized_normal = calculate_vectorized(normal_samples)
vectorized_exponential = calculate_vectorized(exponential_samples)

print("벡터화로 구한 정규의 통계량:", vectorized_normal)

print("벡터화로 구한 지수의 통계량:", vectorized_exponential)
```

executed in 44ms, finished 21:07:20 2024-10-19

벡터화로 구한 정규의 통계량: (-0.006183143374538243, -0.006509914082430952, 0.9972874772331161, 0.7974252944351315, -0.0095728122789598
67, 0.0001355925844417349 4, 1.3548164841507329)

벡터화로 구한 지수의 통계량: (0.9992670879816408, 0.6932205250845452, 0.9915514696427213, 0.7353625378888233, 1.9591725973401473, 5.694
23067921325, 1.099642529492349)

```
loop_normal = calculate_loops(normal_samples)
loop_exponential = calculate_loops(exponential_samples)

print("벡터화 X ,정규:", loop_normal)
print("벡터화 X, 지수:", loop_exponential)
```

executed in 712ms, finished 21:10:05 2024-10-19

벡터화 X ,정규: (-0.006183143374538357, -0.006509914082430952, 0.9972874772331226, 0.7974252944351362, -0.009572812278959392, 0.0001355
925843862238, 1.3548331117479433)

벡터화 X, 지수: (0.9992670879816431, 0.6932205250845452, 0.9915514696427257, 0.7353625378888164, 1.9591725973400995, 5.694230679213224,
1.0996791243757478)

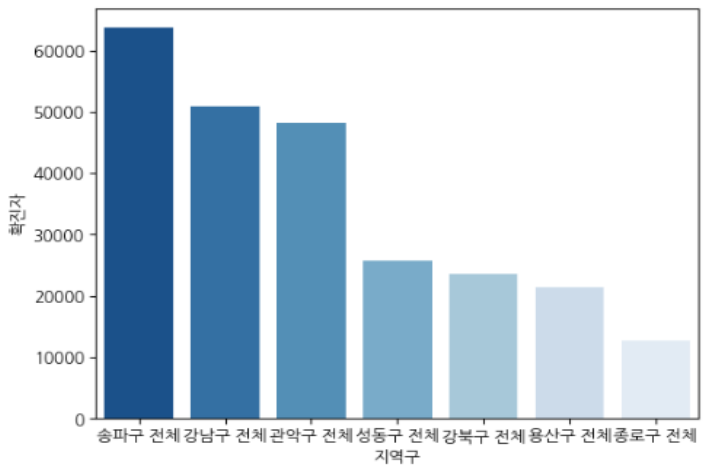
벡터화 경우 = 44ms 그렇지 않은 경우: 712ms 로 예상대로 벡터화가 더 빠름

gpt쓴점

내부함수를 사용하지말라는 조건이 어디까지 인지 몰라서 그럼 sum()과 같은 경우도 말아야 하는것인가? 하는 의문점이 들어 gpt를통해 내부함수 없이 통계량을 들 구하는 함수를 알아보고 같은 답이 나오는지 확인

- **시각화 실습2 강의자료의 21, 28 page 완성하여 코드와 결과 첨부 (20**

```
[58]: sns.barplot(x="지역구", y="확진자", data=data_barplot_220608, palette="Blues_r",
                  order=data_barplot_220608.sort_values(by='확진자', ascending=False)['지역구'])

C:\Users\DESKTOP\AppData\Local\Temp\ipykernel_11160\116782786.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` vari
egend=False` for the same effect.

  sns.barplot(x="지역구", y="확진자", data=data_barplot_220608, palette="Blues_r",
[58]: <Axes: xlabel='지역구', ylabel='확진자'>
```



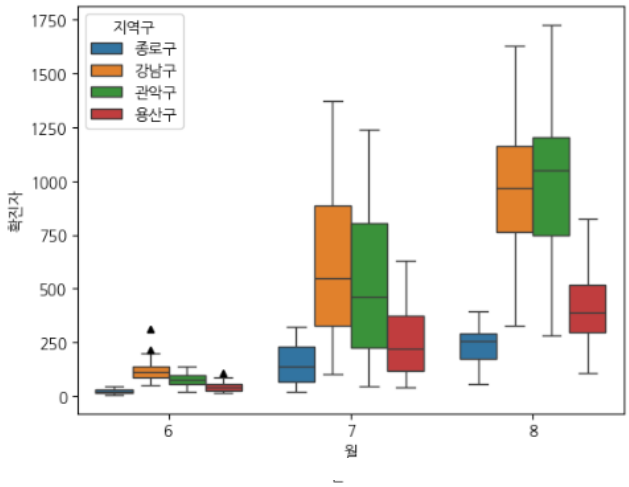
gpt를쓴점

사진과 같이 legend가 이상하게나왔음 gpt로해결방법찾음

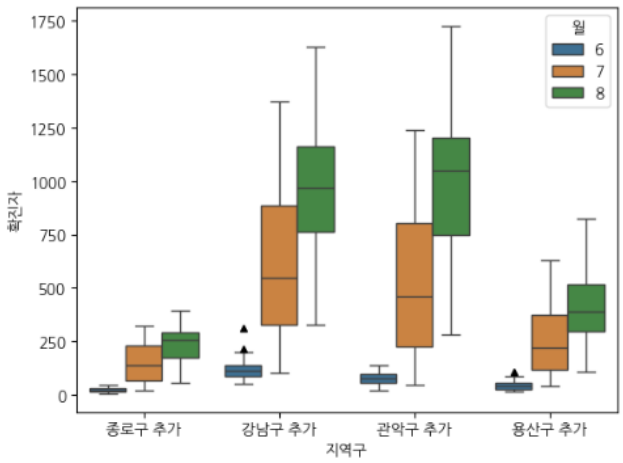
해결- # 현재 범례의 핸들과 레이블 가져오기

handles, _ = ax.get_legend_handles_labels() 가져와서 새로운레이블로 legend생성

In [30]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# 데이터프레임 변환 (기존 변환 코드)
data_transformed = pd.melt(data_boxplot_220608, id_vars=["월"],
                           value_vars=["종로구 추가", "강남구 추가", "관악구 추가", "용산구 추가"],
                           var_name="지역구", value_name="확진자")
flierprops = dict(marker='^', color="black", markersize=5, markerfacecolor="black", markeredgecolor="black")

# boxplot 생성
ax = sns.boxplot(data=data_transformed, x="월", y="확진자", hue="지역구",flierprops=flierprops)

# 현재 범례의 핸들과 레이블 가져오기
handles, _ = ax.get_legend_handles_labels()

# 새로운 레이블로 범례 설정
ax.legend(handles, ["종로구", "강남구", "관악구", "용산구"], title='지역구')
plt.show()
```



In [31]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# 박스 플롯 그리기
month_plot=sns.boxplot(data=data_transformed, x="지역구", y="확진자", hue="월",
            palette={6: (0.19, 0.45, 0.63, 1.00), 7: (0.88, 0.51, 0.17, 1.00), 8:(0.23, 0.57, 0.23, 1.00)},flierprops=flierprops)


handles2, _ = month_plot.get_legend_handles_labels()
month_plot.legend(handles2,["6", "7", "8"],title='월')
plt.show()
```

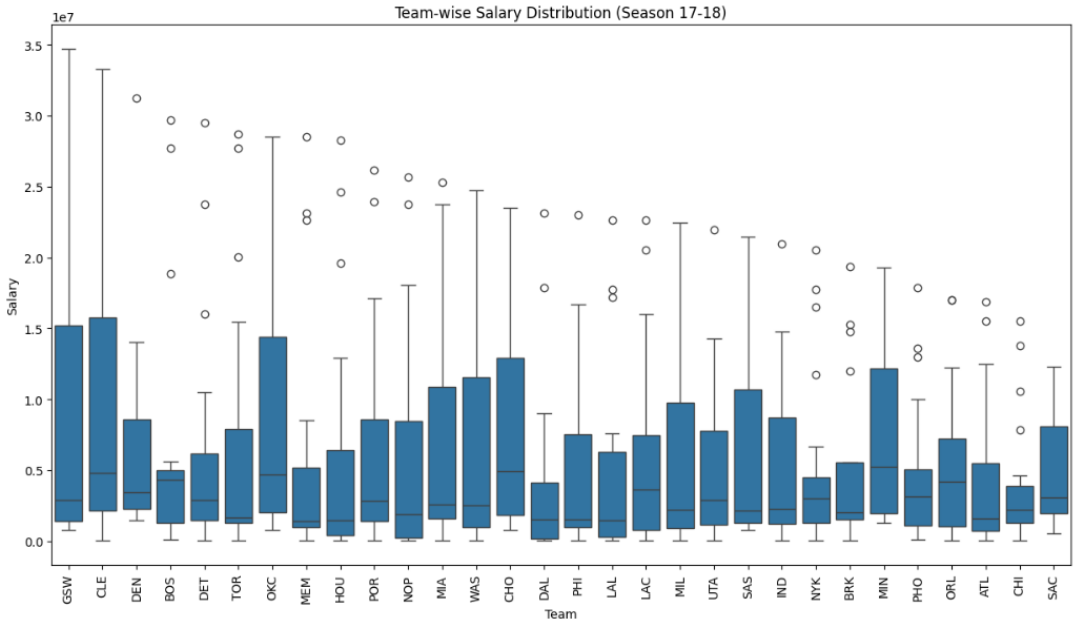


사진과 최대한 비슷하게하기위해 outlier부분도 바꿔봄

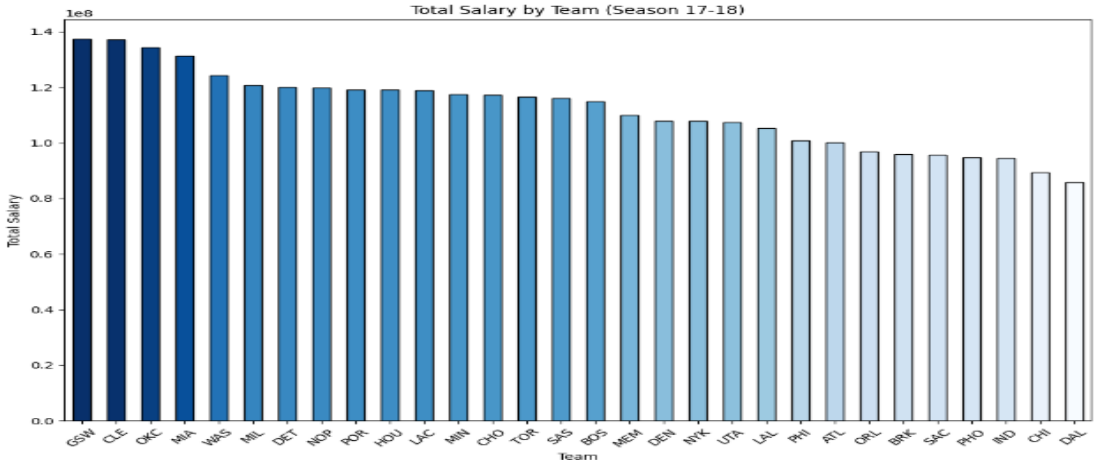
) • 데이터 전처리 강의자료 49page 실습에 있는 3가지 질문에 대한 답을 코드와 plot과 본인 의견 작성 (50)

팀별로 salary 분포를 찍어보기. – 아웃라이어도 상당히 존재하고 선수별 차이도 존재한다고 생각

```
[66]: import matplotlib.pyplot as plt
      import seaborn as sns
      plt.figure(figsize=(15, 8))
      sns.boxplot(x='Tm', y='season17_18', data=mba_sal)
      plt.xticks(rotation=90)
      plt.title('Team-wise Salary Distribution (Season 17-18)')
      plt.xlabel('Team')
      plt.ylabel('Salary')
      plt.show()
      #팀별로 연봉이 어떻게 분포되는지 확인
```



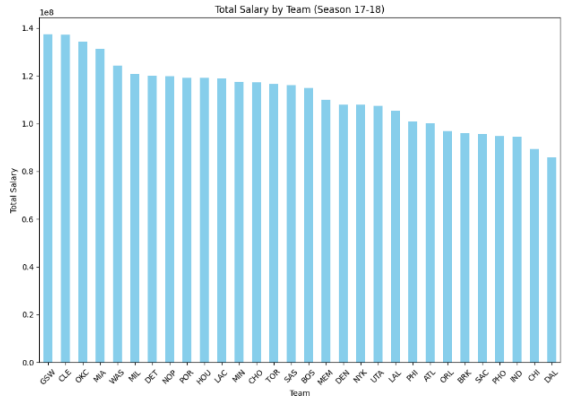

```
[ ]: import pandas as pd
     import matplotlib.pyplot as plt
     import matplotlib.cm as cm

     groupsal = mba_sal.groupby('Tm')['season17_18'].sum()

     groupsal = groupsal.sort_values(ascending=False)

     # 색상 정규화 및 컬러맵 설정
     norm = plt.Normalize(groupsal.min(), groupsal.max())
     colors = cm.Blues(norm(groupsal.values))

     # 막대그래프 시각화
     plt.figure(figsize=(12, 8))
     groupsal.plot(kind='bar', color=colors, edgecolor='black')
     plt.title('Total Salary by Team (Season 17-18)')
     plt.xlabel('Team')
     plt.ylabel('Total Salary')
     plt.xticks(rotation=45)
     plt.show()
     #팀별로 연봉 총액 확인
```



분포2개인 이유 - 팀별로 연봉이 어떻게 분포되는지와 팀별연봉순위 확인

gpt쓴점 barplot의 팀별연봉이 높을수록 색깔이 진해지도록 수정

```
[67]: import pandas as pd
      import matplotlib.pyplot as plt

      groupsal = mba_sal.groupby('Tm')['season17_18'].sum()

      groupsal = groupsal.sort_values(ascending=False)

      plt.figure(figsize=(12, 8))
      groupsal.plot(kind='bar', color='skyblue')
      plt.title('Total Salary by Team (Season 17-18)')
      plt.xlabel('Team')
      plt.ylabel('Total Salary')
      plt.xticks(rotation=45)
      plt.show()
```



- 포지션과 salary의 상관 관계가 있을까?

과정

1.연봉은 1718시즌이므로 그 해에 선수들 데이터를 모음

(mba_season1718=mba_season[mba_season["Year"].isin([2017])] # 선수들 모으기)

2.연봉데이터와 선수데이터를 선수이름과 팀이 같은 경우에 merge함

(연봉데이터를 그 팀에 존재할 때 선수가 받은 돈이기때문)

3.필요컬럼만 추출해서 확인

	Year	Player	Pos	Tm	season17_18
0	2017.0	Alex Abrines	SG	OKC	5725000.0
1	2017.0	Quincy Acy	PF	BRK	1709538.0
2	2017.0	Steven Adams	C	OKC	22471910.0
3	2017.0	Arron Afflalo	SG	SAC	1500000.0
4	2017.0	Alexis Ajinca	C	NOP	4961798.0
...	...	...	...	...	...
265	2017.0	Joe Young	PG	IND	1471382.0
266	2017.0	Thaddeus Young	PF	IND	14796348.0
267	2017.0	Cody Zeller	PF	CHO	12584270.0
268	2017.0	Paul Zipser	SF	CHI	1312611.0
269	2017.0	Ivica Zubac	C	LAL	1312611.0

아노바 분석 사용결과 - ANOVA F-statistic: 0.5387193894551657, p-value: 0.7074148091788368

아노바를 쓴 이유 – 독립변수는 집단을 나타낼 수 있는 범주형 변수 , 종속변수는 연속형 변수 + 각 집단의 평균값 차이가 통계적으로 유의한지 검증한다고 생각했기 때문

내린 결론 - **p-Value**: **0.7074**로, 일반적인 유의수준(0.05)보다 크다. 이는 포지션 간 연봉의 차이가 통계적으로 유의미하지 않음을 나타낸다. 즉, 포지션이 다르다고 해서 평균 연봉이 유의미하게 차이가 나는 것은 아니라는 결론을 내렸다.

• Player efficiency rating (PER)과 salary의 상관 관계가 있을까?– 2014~2017년의 개인 PER 평균

1. 2014~2017 데이터를 모아 PER에 결측치없는지 확인
2. 선수별 평균을 구함
3. 연봉데이터와 선수이름을 기준으로 합침

[55]:
```
mergedata2 = pd.merge(Perdata,mba_sal,on='Player', how='inner')
mergedata2
```

[55]:

	Player	PER	Unnamed: 0	Tm	season17_18
0	A.J. Hammons	8.400000	411	MIA	1312611.0
1	Aaron Brooks	12.166667	319	MIN	2116955.0
2	Aaron Gordon	14.266667	190	ORL	5504420.0
3	Aaron Gray	7.133333	492	DET	452059.0
4	Al Horford	20.125000	11	BOS	27734405.0
...	...	...	...	...	...
469	Wilson Chandler	13.500000	101	DEN	12016854.0
470	Yogi Ferrell	11.233333	429	DAL	1312611.0
471	Zach LaVine	13.400000	251	CHI	3202217.0
472	Zach Randolph	18.650000	98	SAC	12307692.0
473	Zaza Pachulia	15.475000	239	GSW	3477600.0

474 rows × 5 columns

[56]:
```
myanser = mergedata2[['PER', 'season17_18']].corr()
myanser
```

[56]:

	PER	season17_18
PER	1.000000	0.464479
season17_18	0.464479	1.000000

결론 - 약 0.46 라는 상관계수값을 통해 PER와 연봉이 양의 상관관계를 가진다는 결론을 내렸다.