In [1]:

```
from matplotlib import font_manager, rc
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import platform

if platform.system() == 'Windows':

    font_name = font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf"
                                            ).get_name()
    rc('font', family=font_name)
else:
    pass

matplotlib.rcParams['axes.unicode_minus'] = False
```

In [2]:

```
preseason_df = pd.read_csv("./Pre_Season_Batter.csv")
regular_season_df = pd.read_csv("./Regular_Season_Batter.csv")

print(preseason_df.shape)

display(preseason_df.head())
```

(1393, 29)

	batter_id	batter_name	year	team	avg	G	AB	R	H	2B	...	GDP	SLG	OBP	E	height
0	0	가르시아	2018	LG	0.350	7	20	1	7	1	...	1	0.550	0.409	1	177
1	1	강경학	2011	한화	0.000	4	2	2	0	0	...	0	0.000	0.500	0	180
2	1	강경학	2014	한화	-	4	0	2	0	0	...	0	NaN	NaN	0	180
3	1	강경학	2015	한화	0.130	10	23	3	3	0	...	0	0.130	0.286	2	180
4	1	강경학	2016	한화	0.188	14	32	4	6	1	...	0	0.281	0.212	0	180

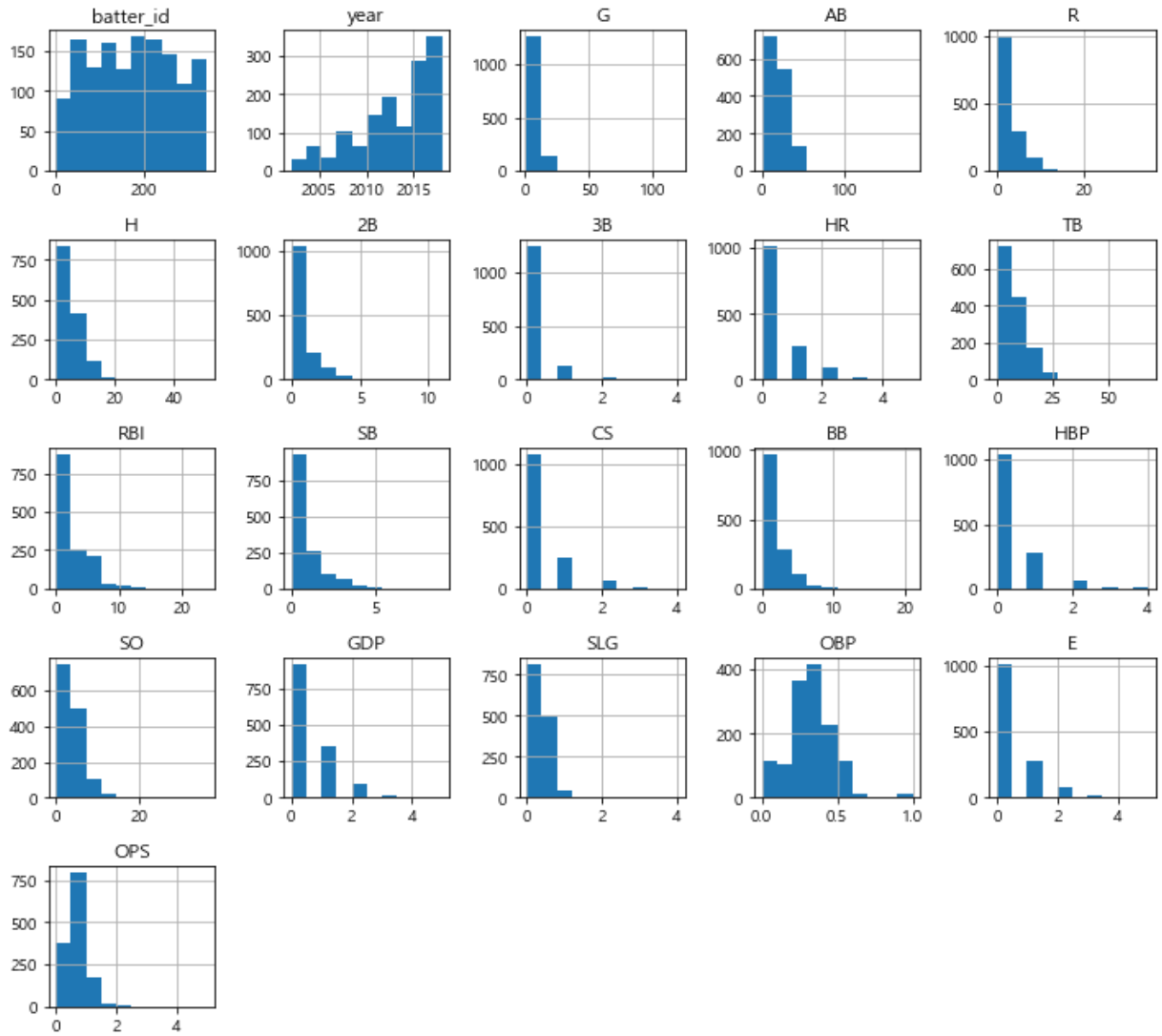
5 rows × 29 columns

In [3]:
```
display(preseason_df.describe())
```

	batter_id	year	G	AB	R	H	2B
count	1393.000000	1393.000000	1393.000000	1393.000000	1393.000000	1393.000000	1393.000000
mean	173.434314	2013.014358	8.705671	19.201723	2.679828	5.021536	0.954774
std	94.716851	4.166757	5.562686	13.395946	2.637212	4.232584	1.196904
min	0.000000	2002.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	99.000000	2010.000000	6.000000	9.000000	1.000000	2.000000	0.000000
50%	178.000000	2014.000000	9.000000	18.000000	2.000000	4.000000	1.000000
75%	254.000000	2017.000000	11.000000	28.000000	4.000000	8.000000	2.000000
max	344.000000	2018.000000	119.000000	183.000000	35.000000	51.000000	11.000000

8 rows × 21 columns

In [4]:
```
preseason_df.hist(figsize=(10,9))
plt.tight_layout()
plt.show()
```



In [5]:
```
# 정규시즌 데이터에서 2002년 이후의 연도별 기록된 선수의 수
regular_count = regular_season_df.groupby('year')['batter_id'].count().rename('regula
```

```
# 프리시즌 데이터에서 연도별 기록된 선수의 수
preseason_count = preseason_df.groupby('year')['batter_id'].count().rename('preseason
pd.concat([regular_count,preseason_count, np.round(preseason_count/regular_count,2).r
        'ratio')], axis = 1).transpose().loc[:,2002:] # 2002년부터 봅니다.
```

Out[5]:

year	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	20
regular	43.00	54.00	68.00	73.00	85.00	98.00	115.00	124.00	130.00	151.0	174.0	194.00	186
preseason	12.00	19.00	28.00	37.00	36.00	43.00	61.00	66.00	72.00	75.0	87.0	104.00	117
ratio	0.28	0.35	0.41	0.51	0.42	0.44	0.53	0.53	0.55	0.5	0.5	0.54	0

In [6]:

```
# 타자의 이름과 연도를 이용해 새로운 인덱스를 생성
regular_season_df['new_idx'] = regular_season_df['batter_name'] + ₩
                               regular_season_df['year'].apply(str)
preseason_df['new_idx'] = preseason_df['batter_name'] + preseason_df['year'].apply(st

# 새로운 인덱스의 교집합
intersection_idx = list(set(regular_season_df['new_idx']). ₩
                        intersection(preseason_df['new_idx']))

# 교집합에 존재하는 데이터만 불러오기
regular_season_new = regular_season_df.loc[
    regular_season_df['new_idx'].apply(lambda x: x in intersection_idx)].copy()
regular_season_new = regular_season_new.sort_values(by = 'new_idx').reset_index(drop=

# 비교를 위해 인덱스로 정렬
preseason_new = preseason_df.loc[preseason_df['new_idx'].apply(
    lambda x: x in intersection_idx)].copy()
preseason_new = preseason_new.sort_values(by = 'new_idx').reset_index(drop=True)

# 검정 코드
print(regular_season_new.shape, preseason_new.shape)
sum(regular_season_new['new_idx'] == preseason_new['new_idx'])
```

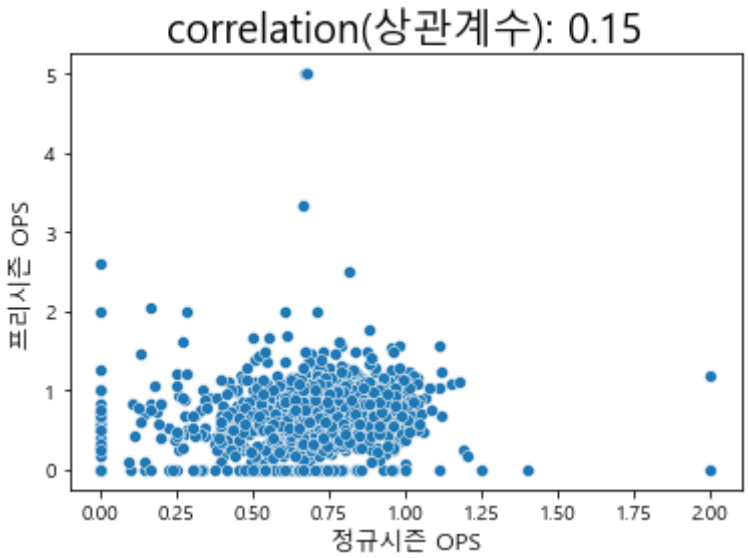
```
(1358, 30) (1358, 30)
```

Out[6]:

```
1358
```

In [7]:

```
# 정규시즌과 프리시즌의 상관관계 계산
correlation = regular_season_new['OPS'].corr(preseason_new['OPS'])
sns.scatterplot(regular_season_new['OPS'], preseason_new['OPS'])
plt.title('correlation(상관계수): '+str(np.round(correlation,2)), fontsize=20)
plt.xlabel("정규시즌 OPS",fontsize=12)
plt.ylabel("프리시즌 OPS",fontsize=12)
plt.show()
```



In [8]:
```
regular_season_df = pd.read_csv("./Regular_Season_Batter.csv")
display(regular_season_df.shape, regular_season_df.head(),regular_season_df.describe(
```

(2454, 29)

	batter_id	batter_name	year	team	avg	G	AB	R	H	2B	...	GDP	SLG	OBP	E	he
0	0	가르시아	2018	LG	0.339	50	183	27	62	9	...	3	0.519	0.383	9	
1	1	강경학	2011	한화	0.000	2	1	0	0	0	...	0	0.000	0.000	1	
2	1	강경학	2014	한화	0.221	41	86	11	19	2	...	1	0.349	0.337	6	
3	1	강경학	2015	한화	0.257	120	311	50	80	7	...	3	0.325	0.348	15	
4	1	강경학	2016	한화	0.158	46	101	16	16	3	...	5	0.257	0.232	7	

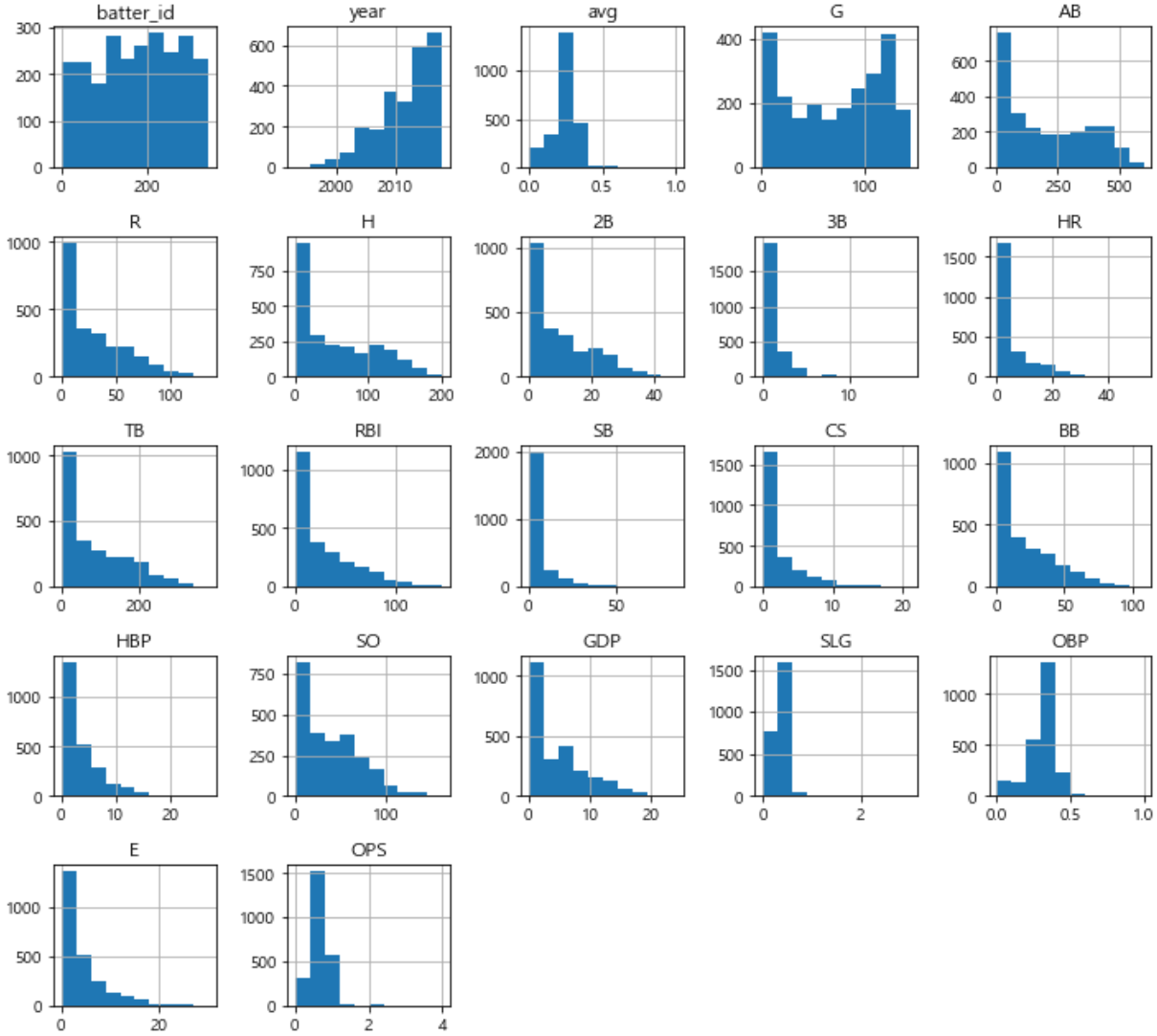
5 rows × 29 columns

	batter_id	year	avg	G	AB	R	H
count	2454.000000	2454.000000	2428.000000	2454.000000	2454.000000	2454.000000	2454.000000
mean	178.079462	2011.614507	0.237559	72.535045	201.514670	29.912388	55.988183

	batter_id	year	avg	G	AB	R	H
std	97.557947	4.992833	0.098440	45.093871	169.537029	28.778759	52.253844
min	0.000000	1993.000000	0.000000	1.000000	0.000000	0.000000	0.000000
25%	101.250000	2008.000000	0.203000	28.000000	38.250000	5.000000	8.000000
50%	183.000000	2013.000000	0.255000	79.000000	163.000000	21.000000	40.000000
75%	265.000000	2016.000000	0.291000	115.000000	357.500000	49.000000	100.000000
max	344.000000	2018.000000	1.000000	144.000000	600.000000	135.000000	201.000000

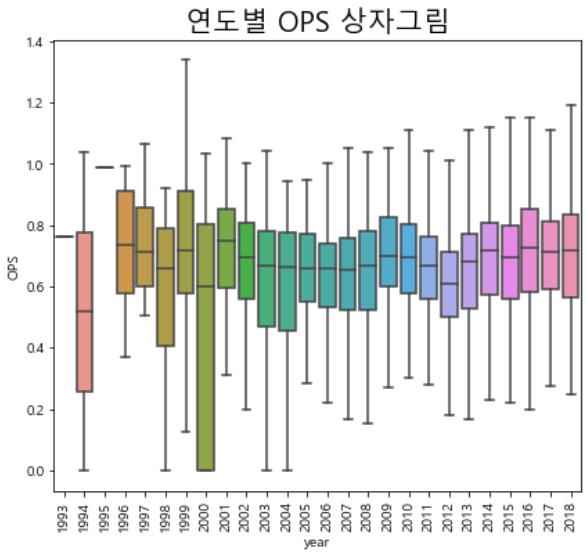
8 rows × 22 columns

In [9]:
```
regular_season_df.hist(figsize=(10,9))
plt.tight_layout()
plt.show()
```



In [10]:
```
plt.figure(figsize=(15,6))
plt.subplot(1,2,1)
g = sns.boxplot(x="year", y="OPS", data=regular_season_df, showfliers=False)
g.set_title('연도별 OPS 상자그림', size = 20)
g.set_xticklabels(g.get_xticklabels(),rotation=90)
plt.subplot(1,2,2)
plt.plot(regular_season_df.groupby('year')['OPS'].median())
```

```
plt.title('연도별 OPS 중앙값', size = 20)
plt.show()
```



In [11]:
```
pd.crosstab(regular_season_df['year'],'count').T
```

Out[11]:

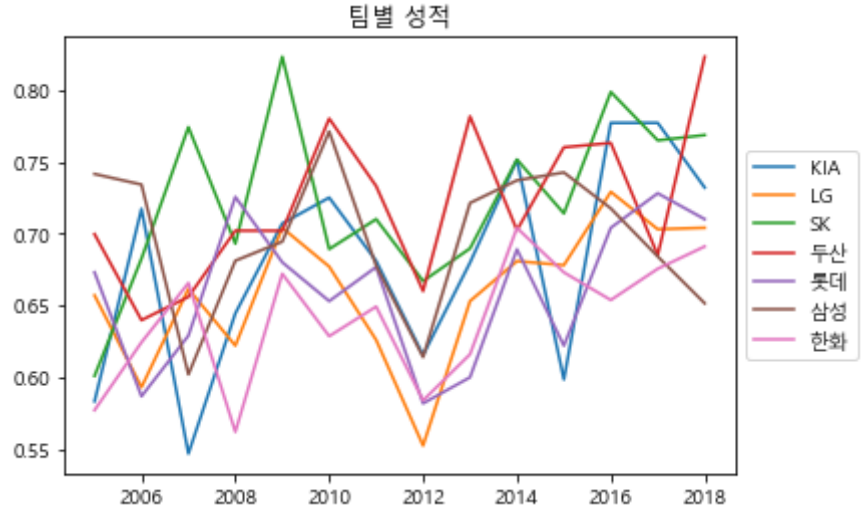
year	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	...	2009	2010	2011	201
col_0															
count	1	2	1	7	8	10	14	20	32	43	...	124	130	151	17

1 rows × 26 columns

In [12]:
```
AVG_OPS_team = regular_season_df.pivot_table(index=['team'], columns='year',
                                             values='OPS', aggfunc='median')
```

In [13]:
```
team_idx = AVG_OPS_team.loc[:,2005:].isna().sum(axis=1) <= 0
```

In [14]:
```
plt.plot(AVG_OPS_team.loc[team_idx,2005:].T)
plt.legend(AVG_OPS_team.loc[team_idx,2005:].T.columns,
           loc='center left', bbox_to_anchor=(1, 0.5))
plt.title('팀별 성적')
plt.show()
```



In [15]:
```
import re
```

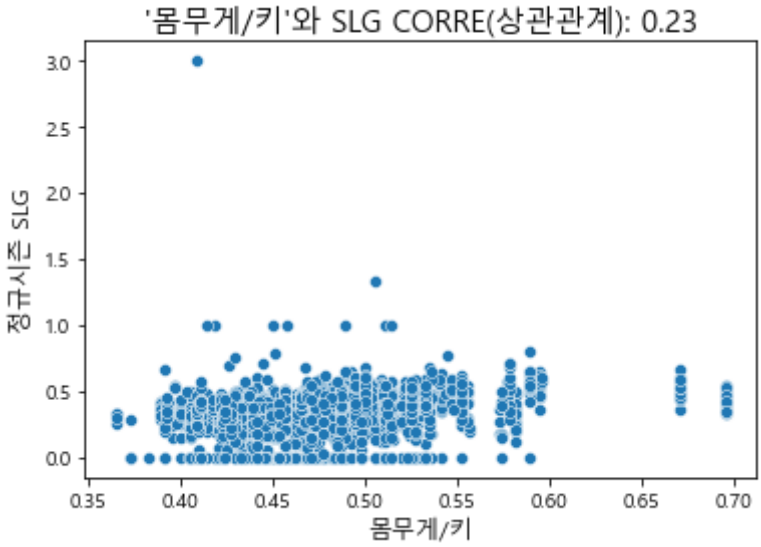
In [16]:
```
regular_season_df['weight'] = regular_season_df['height/weight'].apply(
    lambda x: int(re.findall('Wd+',x.split('/')[1])[0]) if pd.notnull(x) else x)
```

In [17]:
```
regular_season_df['height'] = regular_season_df['height/weight'].apply(
    lambda x: int(re.findall('Wd+',x.split('/')[0])[0]) if pd.notnull(x) else x)
```

In [18]:
```
print(regular_season_df['height/weight'][0], regular_season_df['height'][0],
      regular_season_df['weight'][0])
```

```
177cm/93kg 177.0 93.0
```

In [21]:
```
CORRE = regular_season_df['weight_per_height'].corr(regular_season_df['SLG'])
sns.scatterplot(regular_season_df['weight_per_height'], regular_season_df['SLG'])
plt.title("'몸무게/키'와 SLG CORRE(상관관계): " + str(np.round(CORRE, 2)), W
          fontsize=15)
plt.ylabel('정규시즌 SLG', fontsize=12)
plt.xlabel('몸무게/키', fontsize=12)
plt.show()
```



In [22]:
```
regular_season_df['position'].value_counts()
```

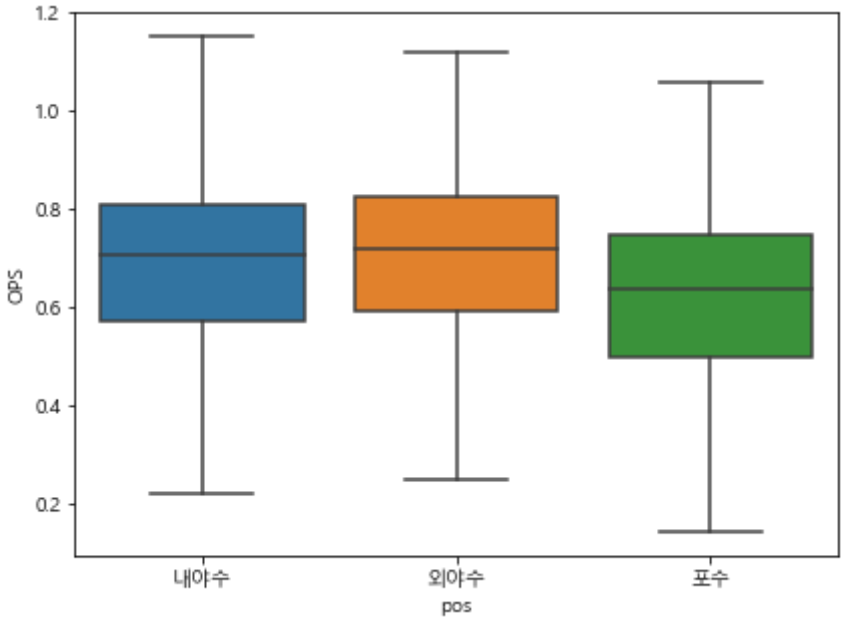
Out[22]:
```
내야수(우투우타)    643
외야수(우투우타)    230
외야수(좌투좌타)    201
포수(우투우타)     189
외야수(우투좌타)    184
내야수(우투좌타)    141
내야수(좌투좌타)     36
포수(우투좌타)      14
내야수(우투양타)      7
외야수(우투양타)      7
Name: position, dtype: int64
```

In [23]:
```
regular_season_df['pos']=regular_season_df['position'].apply(
    lambda x: x.split('(')[0] if pd.notnull(x) else x)
```

In [24]:
```
# 우타, 좌타, 양타
regular_season_df['hit_way'] = regular_season_df['position'].apply(
    lambda x: x[-3:-1] if pd.notnull(x) else x)
print(regular_season_df['position'][0], regular_season_df['pos'][0],
      regular_season_df['hit_way'][0])
```

```
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
ax = sns.boxplot(x='pos', y='OPS', data = regular_season_df, showfliers=False)
```

내야수(우투우타) 내야수 우타



In [25]:
```
avg = regular_season_df.groupby(['pos'])['OPS'].median().to_dict()
```

In [26]:
```
nobs = regular_season_df['pos'].value_counts().to_dict()
```

In [27]:
```
for key in nobs: nobs[key] = "n: " + str(nobs[key])
```

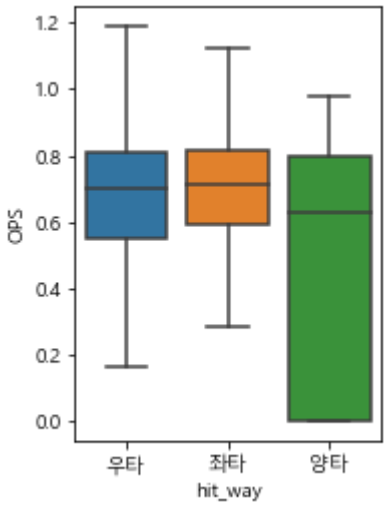
In [28]:
```
Xtick_labels = [item.get_text() for item in ax.get_xticklabels()]
```

In [29]:
```
for label in ax.get_xticklabels():
    ax.text(Xtick_labels.index(label.get_text()),
            avg[label.get_text()] + 0.03, nobs[label.get_text()],
            horizontalalignment='center', size='large', color='w', weight='semibold')
```

In [30]:
```
ax.set_title('포지션별 OPS')
```

Out[30]:
```
Text(0.5, 1.0, '포지션별 OPS')
```

In [31]:
```
plt.subplot(1,2,2)
ax = sns.boxplot(x='hit_way', y='OPS', data = regular_season_df, showfliers=False)
```



In [32]:
```
avg = regular_season_df.groupby(['hit_way'])['OPS'].median().to_dict()
```

In [33]:
```
nobs = regular_season_df['hit_way'].value_counts().to_dict()
```

In [34]:
```
for key in nobs: nobs[key] = "n: " + str(nobs[key])
Xtick_labels = [item.get_text() for item in ax.get_xticklabels()]
```

In [35]:
```
# tick은 tick의 위치
for label in ax.get_xticklabels():
    ax.text(Xtick_labels.index(label.get_text()), avg[label.get_text()] + 0.03,
            nobs[label.get_text()], horizontalalignment='center', size='large',
            color='w', weight='semibold')
ax.set_title('타석방향별 OPS')

plt.show()

regular_season_df['career'].head()
```

Out[35]:
```
0    쿠바 Ciego de Avila Maximo Gomez Baez(대)
1                        광주대성초-광주동성중-광주동성고
2                        광주대성초-광주동성중-광주동성고
3                        광주대성초-광주동성중-광주동성고
4                        광주대성초-광주동성중-광주동성고
Name: career, dtype: object
```

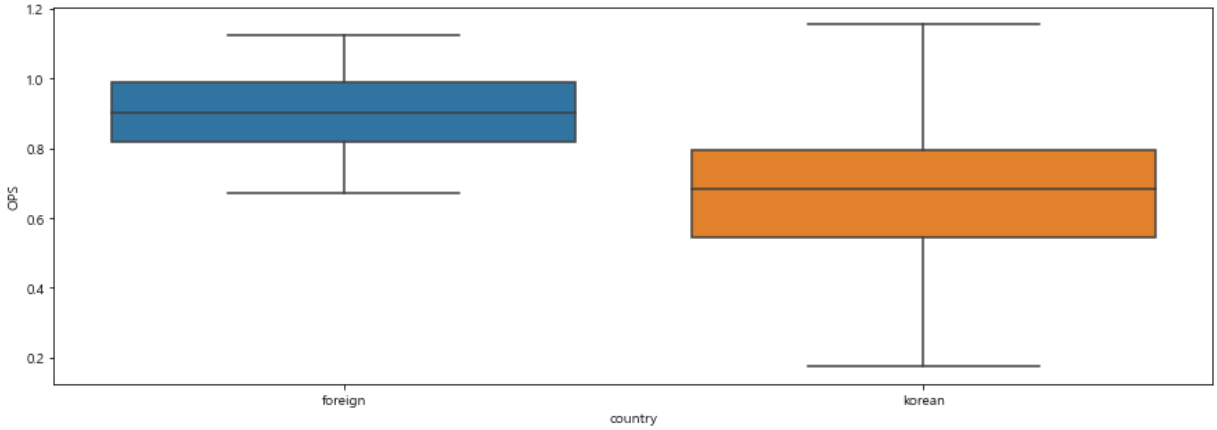
In [36]:
```
# career를 split
outside_country = regular_season_df['career'].apply(
    lambda x: x.replace('-', ' ').split(' ')[0])
```

In [37]:
```
# 외국만 추출
outside_country_list = list(set(outside_country.apply(
    lambda x: np.nan if '초' in x else x)))
```

In [38]:
```
# 결측치 처리
outside_country_list = [x for x in outside_country_list if str(x) != 'nan']
outside_country_list

regular_season_df['country'] = outside_country
regular_season_df['country'] = regular_season_df['country'].apply(
    lambda x: x if pd.isnull(x)
                else ('foreign' if x in outside_country_list else 'korean'))
regular_season_df[['country']].head()

plt.figure(figsize=(15,5))
ax = sns.boxplot(x='country', y='OPS', data = regular_season_df, showfliers=False)
```



In [39]:
```
# 내외국인 별 OPS 중앙값 dict
avg = regular_season_df.groupby(['country'])['OPS'].median().to_dict()
# 내외국인 관측치 수 dict
nobs = regular_season_df['country'].value_counts().to_dict()
# 키 값을 'n: 값' 형식으로 변환
for key in nobs: nobs[key] = "n: " + str(nobs[key])
```

In [40]:
```
# 그래프의 Xticks text 값 얻기
Xtick_labels = [item.get_text() for item in ax.get_xticklabels()]

for label in ax.get_xticklabels(): # tick은 tick의 위치, label은 그에 해당하는 text 값
    ax.text(Xtick_labels.index(label.get_text()), avg[label.get_text()] + 0.03, W
            nobs[label.get_text()], # x 좌표, y 좌표, 해당 text
            horizontalalignment='center', size='large', color='w', weight='semibold')
ax.set_title('국적별 OPS')
plt.show()

regular_season_df['starting_salary'].value_counts()
```

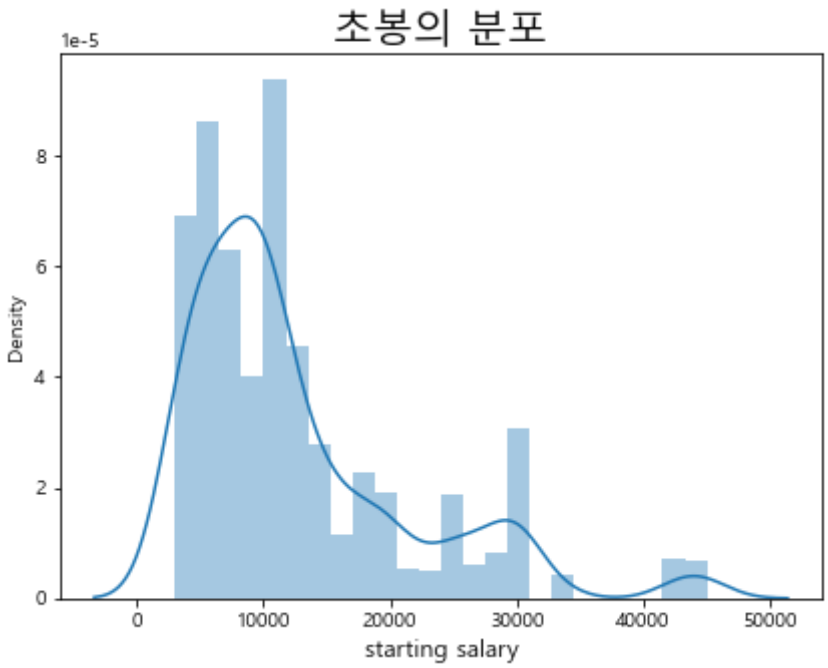
Out[40]:
```
10000만원      177
6000만원       117
3000만원       105
9000만원        97
5000만원        91
8000만원        89
30000만원       74
4000만원        62
12000만원       62
18000만원       54
7000만원        53
11000만원       49
13000만원       48
20000만원       46
25000만원       45
15000만원       41
16000만원       28
14000만원       26
28000만원       20
43000만원       17
45000만원       16
27000만원       15
21000만원       13
23000만원       12
6500만원        10
33000만원       10
100000달러       4
300000달러       3
50000달러        2
17000만원        1
Name: starting_salary, dtype: int64
```

In [42]:
```
regular_season_df['starting_salary'] = regular_season_df['starting_salary'].apply(
    lambda x: x if pd.isnull(x)
                else(int(re.findall('Wd+',x)[0]) if '만원' in x else np.nan))
```

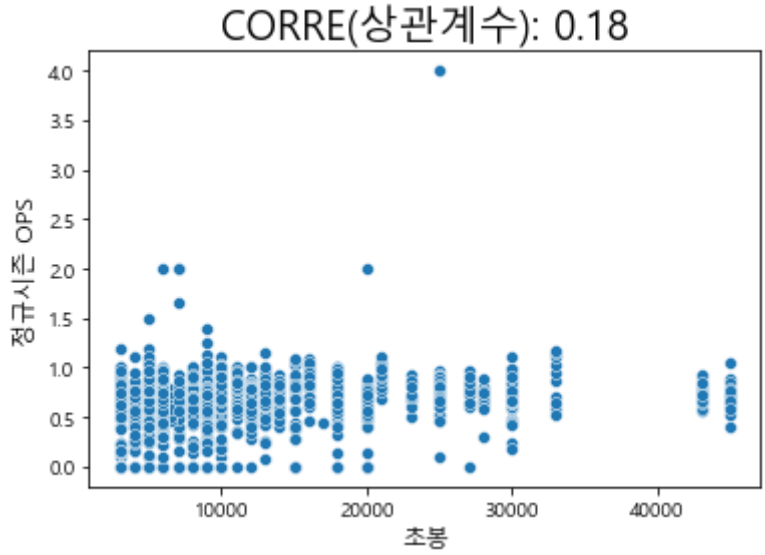
In [48]:
```
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
b=sns.distplot(regular_season_df['starting_salary']. W
               loc[regular_season_df['starting_salary'].notnull()], hist=True)
b.set_xlabel("starting salary",fontsize=12)
b.set_title('초봉의 분포', fontsize=20)

#plt.subplot(1,2,2)
```

Out[48]:



In [44]:
```
# 정규시즌과 프리시즌의 계산
CORRE = regular_season_df['starting_salary'].corr(regular_season_df['OPS'])
b = sns.scatterplot(regular_season_df['starting_salary'], regular_season_df['OPS'])
b.axes.set_title('CORRE(상관계수): '+str(np.round(CORRE,2)), fontsize=20)
b.set_ylabel("정규시즌 OPS",fontsize=12)
b.set_xlabel("초봉",fontsize=12)
plt.show()
```


CORRE(상관계수): 0.18
정규시즌 OPS
초봉
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
10000
20000
30000
40000

In [ ]:

In [ ]:

In [ ]: