Sampling Hypothetical Player Statistics Based on Minutes Played

In [1]:
```
import pandas as pd
import numpy as np
```

Importing Data

In [2]:
```
team_df=pd.read_csv('NBA_Team_Data_1999-2020.csv',index_col=0)
```

In [3]:
```
mapping=pd.read_csv('mapping.csv',index_col=0)
```

In [4]:
```
mapping
player_df=pd.read_csv('NBA_Player_Data_1999-2020.csv',index_col=0)
player_df.head(5)
```

Out[4]:

	PLAYER	YEAR	POS	GP	MIN	PTS	FGM	FGA	FG%	3PM	...	OREB%	DREB%	REB%	F
0	Shaquille O'Neal	1999-00	C	79	40.0	29.7	12.1	21.1	57.4	0.0	...	10.5	21.9	16.4	
1	Allen Iverson	1999-00	SG	70	40.8	28.4	10.4	24.8	42.1	1.3	...	2.4	7.0	4.6	
2	Grant Hill	1999-00	SF	74	37.5	25.8	9.4	19.2	48.9	0.5	...	3.6	14.6	9.1	
3	Vince Carter	1999-00	G	82	38.1	25.7	9.6	20.7	46.5	1.2	...	4.7	10.7	7.6	
4	Karl Malone	1999-00	PF	82	35.9	25.5	9.2	18.0	50.9	0.0	...	6.4	22.3	14.5	

5 rows × 43 columns

In [5]:
```
def add_mapping(df,mapping):
    return pd.merge(df ,mapping,on='TEAM')
```

In [6]:
```
player_df=add_mapping(player_df,mapping)
```

In [7]:
```
player_df
```

Out[7]:

	PLAYER	YEAR	POS	GP	MIN	PTS	FGM	FGA	FG%	3PM	...	DREB%	REB%	TO RATIO
0	Shaquille O'Neal	1999-00	C	79	40.0	29.7	12.1	21.1	57.4	0.0	...	21.9	16.4	8.8
1	Kobe Bryant	1999-00	SF	66	38.2	22.5	8.4	17.9	46.8	0.7	...	11.4	7.9	9.8
2	Glen Rice	1999-00	SF	80	31.6	15.9	5.3	12.3	43.0	1.1	...	10.0	6.2	7.9
3	Ron Harper	1999-00	G	80	25.5	7.0	2.7	6.6	39.9	0.4	...	11.3	7.9	13.2
4	Rick Fox	1999-00	SF	82	18.0	6.5	2.5	6.1	41.4	0.7	...	8.7	6.3	11.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8978	Frank Jackson	2019-20	PG	59	13.5	6.3	2.3	5.6	40.5	0.8	...	7.0	4.8	10.4
8979	Nickeil Alexander-Walker	2019-20	SG	47	12.6	5.7	2.1	5.7	36.8	1.0	...	12.2	6.8	12.7
8980	Kenrich Williams	2019-20	SF	39	21.3	3.5	1.3	3.8	34.7	0.6	...	15.7	10.4	9.3
8981	Zylan Cheatham	2019-20	SF	4	12.8	3.0	1.5	2.3	66.7	0.0	...	12.8	9.5	25.0
8982	Josh Gray	2019-20	G	2	11.5	1.0	0.5	1.0	50.0	0.0	...	4.0	4.5	55.6

8983 rows × 44 columns

Getting NBA Rosters

In [8]:
```
# function to get roster

def getRoster(player_df,team,year):
    return player_df[(player_df['TEAM NAME']==team) & (player_df['YEAR']==y
```

In [9]:
```
# testing function

getRoster(player_df,'Miami Heat','2012-13')
```

Out[9]:

	PLAYER	YEAR	POS	GP	MIN	PTS	FGM	FGA	FG%	3PM	...	DREB%	REB%	TO RATIO
3483	LeBron James	2012-13	SF	76	37.9	26.8	10.1	17.8	56.5	1.4	...	18.0	11.5	9.6
3484	Dwyane Wade	2012-13	G	69	34.7	21.2	8.2	15.8	52.1	0.2	...	11.2	7.9	10.6
3485	Chris Bosh	2012-13	PF	74	33.2	16.6	6.6	12.3	53.5	0.3	...	15.3	11.3	9.9
3487	Mario Chalmers	2012-13	G	77	26.9	8.6	2.9	6.9	42.9	1.6	...	7.6	4.6	12.3
3486	Ray Allen	2012-13	SG	79	25.8	10.9	3.7	8.2	44.9	1.8	...	9.0	5.7	10.9
3488	Shane Battier	2012-13	SF	72	24.8	6.6	2.1	5.0	42.0	1.9	...	7.1	4.9	7.0
3489	Norris Cole	2012-13	F	80	19.9	5.6	2.2	5.3	42.1	0.4	...	7.1	4.4	14.2
3493	Udonis Haslem	2012-13	PF	75	18.9	3.9	1.7	3.3	51.4	0.0	...	22.9	16.0	12.7
3492	Mike Miller	2012-13	SG	59	15.3	4.8	1.7	3.9	43.3	1.2	...	15.3	9.3	9.5
3491	Chris Andersen	2012-13	PF	42	14.9	4.9	1.7	2.9	57.7	0.0	...	18.0	14.4	11.7
3490	Rashard Lewis	2012-13	PF	55	14.4	5.2	1.9	4.5	41.4	0.9	...	13.4	8.3	9.9
3497	Joel Anthony	2012-13	C	62	9.1	1.4	0.6	1.1	51.5	0.0	...	12.3	10.6	19.1
3494	Juwan Howard	2012-13	PF	7	7.3	3.0	1.4	2.7	52.6	0.0	...	16.3	9.0	13.8
3496	James Jones	2012-13	SG	38	5.8	1.6	0.6	1.6	34.4	0.4	...	8.8	4.9	3.8
3495	Josh Harrellson	2012-13	C	6	5.2	1.7	0.7	1.5	44.4	0.2	...	10.0	10.6	23.1
3498	Jarvis Varnado	2012-13	PF	13	4.5	0.6	0.2	0.5	42.9	0.0	...	11.3	7.9	33.3

16 rows × 44 columns

In [10]:
```
# getting all rosters with minimum 8 players

rosters=[]

for team in player_df['TEAM NAME'].unique():
    for year in player_df[player_df['TEAM NAME']==team]['YEAR'].unique():
        if (len(getRoster(player_df,team,year)))>=8:
            rosters.append(getRoster(player_df,team,year).head(8))
```

In [11]:
```
rosters[0]
```

Out[11]:

	PLAYER	YEAR	POS	GP	MIN	PTS	FGM	FGA	FG%	3PM	...	DREB%	REB%	TO RATIO	EF
0	Shaquille O'Neal	1999-00	C	79	40.0	29.7	12.1	21.1	57.4	0.0	...	21.9	16.4	8.8	5
1	Kobe Bryant	1999-00	SF	66	38.2	22.5	8.4	17.9	46.8	0.7	...	11.4	7.9	9.8	4
2	Glen Rice	1999-00	SF	80	31.6	15.9	5.3	12.3	43.0	1.1	...	10.0	6.2	7.9	4
3	Ron Harper	1999-00	G	80	25.5	7.0	2.7	6.6	39.9	0.4	...	11.3	7.9	13.2	4
7	A.C. Green	1999-00	F	82	23.5	5.0	2.1	4.7	44.7	0.0	...	16.0	12.0	9.4	4
5	Derek Fisher	1999-00	PG	78	23.1	6.3	2.1	6.2	34.6	0.7	...	6.2	3.8	8.9	4
6	Robert Horry	1999-00	PF	76	22.2	5.7	2.1	4.8	43.8	0.4	...	12.4	10.2	12.1	4
4	Rick Fox	1999-00	SF	82	18.0	6.5	2.5	6.1	41.4	0.7	...	8.7	6.3	11.5	4

8 rows × 44 columns

Analyzing Minutes Played by NBA Rosters

In [12]:
```
minutes_wide={
    'player1':[],
    'player2':[],
    'player3':[],
    'player4':[],
    'player5':[],
    'player6':[],
    'player7':[],
    'player8':[]
}

minutes_long={
    'player':[],
    'index':[],
    'minutes':[],
    'team':[],
    'year':[],
    'pie':[],
    'total_mins':[]

}


team_mins=[]
team_score=[]



for team in rosters:
    for i in range(len(team)):
        minutes_wide['player{}'.format(i+1)].append((team.iloc[i]['MIN']))

        minutes_long['player'].append('player{}'.format(i+1))
        minutes_long['index'].append(i+1)
        minutes_long['year'].append(team.iloc[i]['YEAR'])

        minutes_long['minutes'].append((team.iloc[i]['MIN']))
        minutes_long['pie'].append((team.iloc[i]['PIE']))
        minutes_long['team'].append((team.iloc[i]['TEAM']+" "+team.iloc[i][
        minutes_long['total_mins'].append(np.sum(team['MIN']))





    team_mins.append(np.sum(team['MIN']))
    team_score.append(np.sum(team['PTS']))
```

In [13]:
```
# wide format
min_dist=pd.DataFrame.from_dict(minutes_wide)
min_dist
```

Out[13]:

	player1	player2	player3	player4	player5	player6	player7	player8
0	40.0	38.2	31.6	25.5	23.5	23.1	22.2	18.0
1	40.9	39.5	35.5	31.0	27.9	24.2	22.9	20.1
2	38.3	36.1	28.2	27.9	26.4	24.0	21.5	19.7
3	41.5	37.8	34.5	29.3	28.7	22.7	18.6	14.5
4	37.6	36.8	34.5	32.7	23.8	22.3	21.6	21.2
...	...	...	...	...	...	...	...	...
573	35.5	32.9	30.6	30.4	28.2	27.6	26.6	24.6
574	36.1	34.2	32.7	29.7	25.0	24.9	23.5	23.3
575	36.4	36.2	36.1	31.5	27.2	26.2	23.7	19.6
576	35.9	33.0	30.6	29.8	27.6	25.5	23.5	20.0
577	34.7	33.9	32.1	27.8	27.0	26.4	24.4	21.3

578 rows × 8 columns

In [14]:
```
# long format

min_long=pd.DataFrame.from_dict(minutes_long)
```

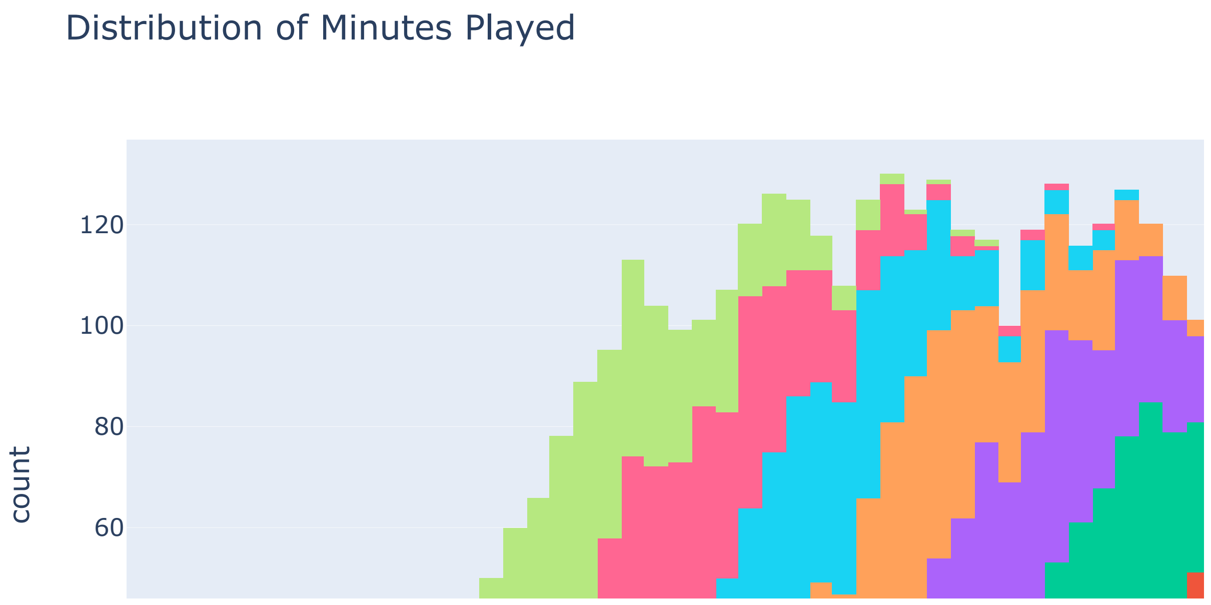
In [15]:
```
# save minutes data

min_long.drop(['team','year','pie'],axis=1).to_csv('minutes_played.csv')
```

In [16]:
```
# analyzing mean minutes played by player 1 to player 8

[np.mean(min_long[min_long['player']==i]['minutes']) for i in min_long['pla
```

Out[16]:
```
[36.595674740484455,
 34.399653979238764,
 31.93408304498267,
 29.616608996539757,
 27.221626297577856,
 24.622837370242213,
 22.17058823529412,
 19.97058823529412]
```

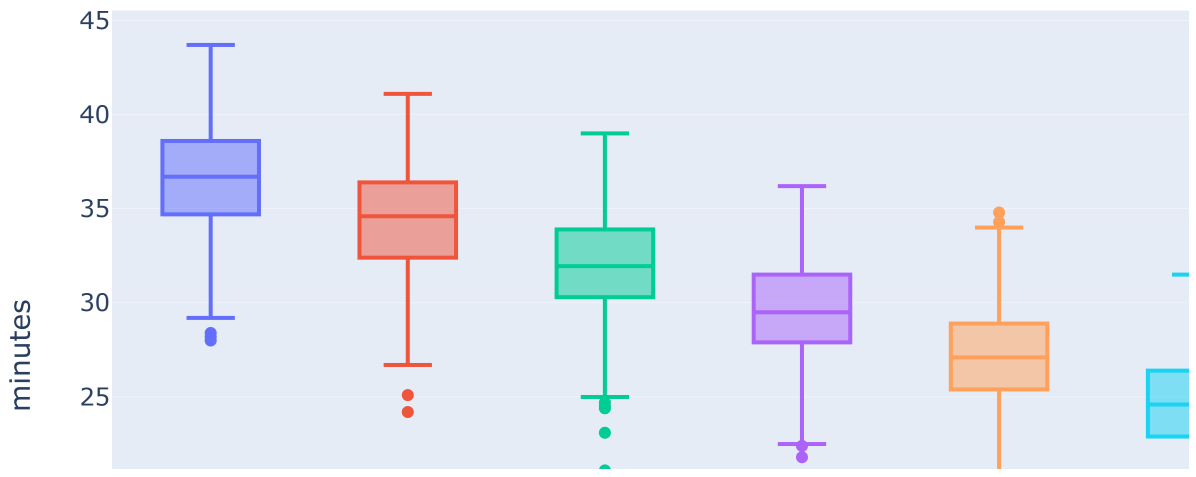
Data Visualization for Minutes Played

In [17]:
```
import plotly.express as px

# histogram
fig = px.histogram(min_long, x="minutes",color='player',title='Distribution
fig.show()
```



In [19]:
```
# boxplot

fig = px.box(min_long, x="player", y="minutes",color='player',title='Histog
fig.show()
```

Histogram of Minutes Played

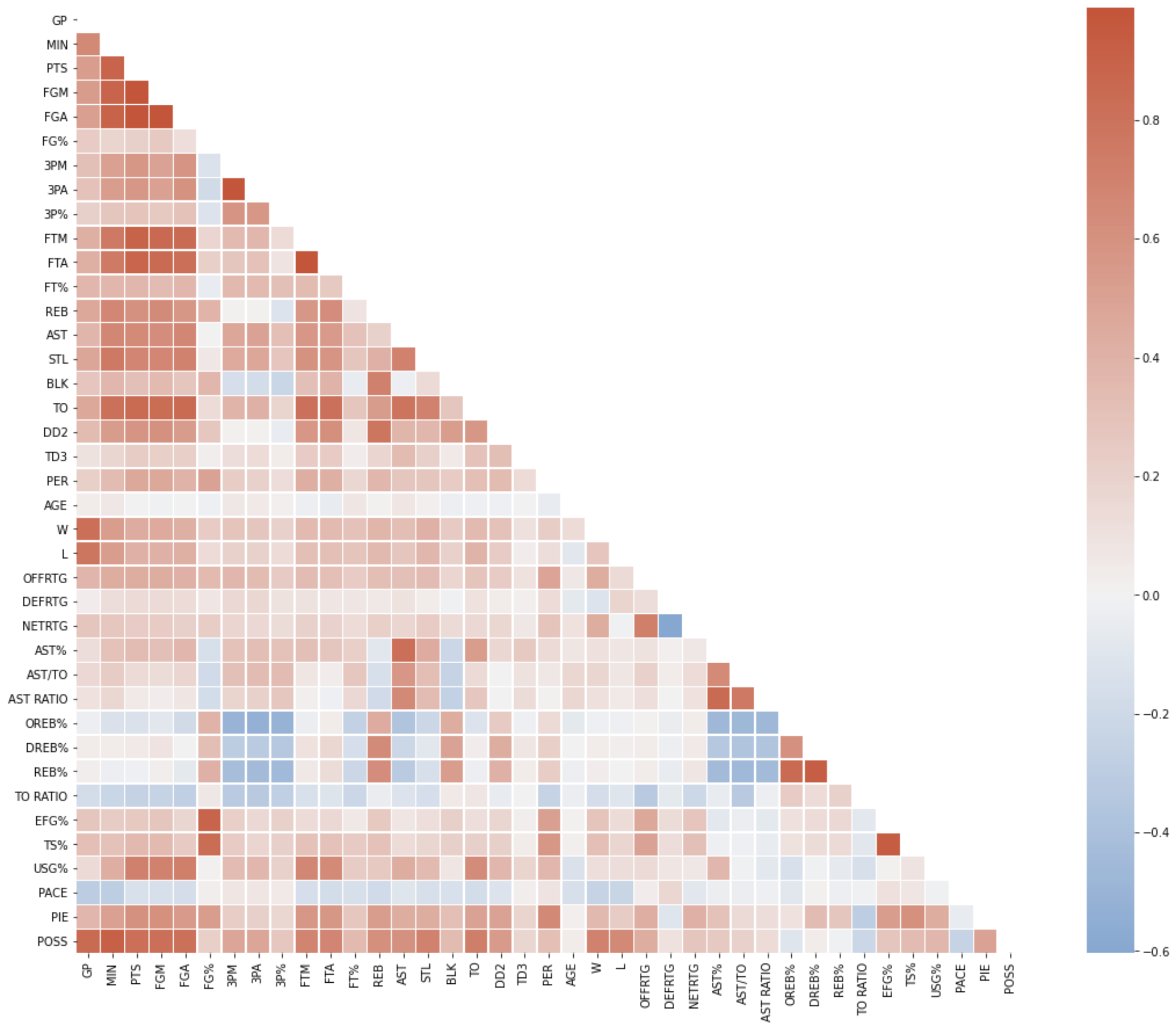


In [ ]:
```
player_df.drop(['PLAYER','YEAR','TEAM NAME','TEAM'],axis=1).dtypes
```

Metrics that Correlate with Minutes Played

In [23]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Generate a heatmap with annotations on and the colorbar off

corr= player_df.drop(['PLAYER','YEAR','TEAM NAME','TEAM'],axis=1).corr()

mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(20, 15))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(250, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap,  center=0,
            square=True, linewidths=.5)
```

Out[23]:
```
<AxesSubplot:>
```



In [ ]:
```
corr[(corr['MIN']>0.5) & (corr['MIN']!=1)][['MIN']].drop(['W','L','GP'])
```

Kernel Density Estimator for Distribution of Minutes Played

In [ ]:
```
from sklearn.neighbors import KernelDensity
```

In [ ]:
```
from sklearn.neighbors import KernelDensity

minutes=[]

for i in min_long['team'].unique():
    team=min_long[min_long['team']==i]
    minutes.append(list(team['minutes']))
```

In [ ]:
```
X=np.array(minutes)
```

In [ ]:
```
# Training kernel density estimator
kde = KernelDensity(kernel='gaussian', bandwidth=1.12883).fit(X)
```

In [ ]:
```
import pickle
```

In [ ]:
```
# Saving KDE
pickle.dump(kde, open( "kde.pkl", "wb" ))
test=pickle.loads(pickle.dumps(kde))
```

In [ ]:
```
# testing KDE minutes sampling
test.sample(1)
```

Adjusting Player Stats to Minutes Played

In [ ]:
```
metrics=['POS', 'MIN', 'PTS', 'FGM', 'FGA',
         'FG%', '3PM', '3PA', '3P%', 'FTM', 'FTA', 'FT%', 'REB', 'AST', 'STL'
         'BLK', 'TO', 'DD2', 'TD3', 'PER', 'AGE', 'OFFRTG', 'DEFRTG', 'NETRTG
         'AST%', 'AST/TO', 'AST RATIO', 'OREB%', 'DREB%', 'REB%', 'TO RATIO',
         'EFG%', 'TS%', 'USG%', 'PACE', 'PIE', 'POSS']


def adjust_to_minutes(minutes:float, player_stats:pd.DataFrame):
    adjusted_player_stat=[]

    for i in metrics:
        if i=='POS':

            adjusted_player_stat.append(max(set(list(player_stats[i])), key
        elif i=='MIN':
            adjusted_player_stat.append(minutes)


        elif (i in corr[(corr['MIN']>0.5) & (corr['MIN']!=1)][['MIN']].inde
            adjusted_player_stat.append(np.mean((player_stats[i])/np.mean(p

        else:
            adjusted_player_stat.append(np.mean((player_stats[i])))

    return adjusted_player_stat



def player_stats_sampling(player: str, minutes: float,stats_selection_metho

    if player not in player_df['PLAYER'].unique():
        raise Exception("Invalid player: '{}' - Please select from player l

    player_stats=player_df[player_df['PLAYER']==player]

    if stats_selection_method=='best':
        return [player]+adjust_to_minutes(minutes,player_stats.sort_values(

    elif stats_selection_method=='prime':
        if prime_window==None:
            years=5
        else:
            years=prime_window

        return [player]+adjust_to_minutes(minutes,player_stats.sort_values(

    else:
        return [player]+adjust_to_minutes(minutes,player_stats[metrics])




def team_stats_sampling(team:list, minutes_selection_method:str ='sample',

    if len(team)==8:
        raise Exception("Team must contain 8 players, contains '{}' players
    if minutes_selection_method not in ['sample','average']:
        raise Exception("Invalid minutes_selection_method: '{}' - Please se

    if stats_selection_method not in ['prime','average','best']:
         raise Exception("stats_selection_method: '{}' - Please select from

    if stats_selection_method!='prime':
        if prime_window!=None:
            raise Exception("prime_window requires stats_selection_method='


    if minutes_selection_method=='sample':
        minutes=kde.sample(1)[0]
    else:
        minutes=[np.mean(min_long[min_long['player']==i]['minutes']) for i


    team_stats=[]

    for player,minute in zip(team,minutes):

        team_stats.append(player_stats_sampling(player,minutes=minute,stats


    return team_stats
```

In [ ]:
```
kde.sample(1)[0][0]
```

Code above was finnished, implemented & tested in Main.ipynb file